JMLR: Workshop and Conference Proceedings 80:1–16, 2017 ACML 2017

Scale-Invariant Recognition by Weight-Shared CNNs in Parallel

Ryo Takahashi TAKAHASHI@AI.CS.KOBE-U.AC.JP
Takashi Matsubara MATSUBARA@PHOENIX.KOBE-U.AC.JP
Kuniaki Uehara UEHARA@KOBE-U.AC.JP
Graduate School of System Informatics, Kobe University, 1-1 Rokko-dai, Nada, Kobe, Hyogo 657-8501, Japan

Editors: Yung-Kyun Noh and Min-Ling Zhang

Abstract

Deep convolutional neural networks (CNNs) have become one of the most successful methods for image processing tasks in past few years. Recent studies on modern residual architectures, enabling CNNs to be much deeper, have achieved much better results thanks to their high expressive ability by numerous parameters. In general, CNNs are known to have the robustness to the small parallel shift of objects in images by their local receptive fields, weight parameters shared by each unit, and pooling layers sandwiching them. However, CNNs have a limited robustness to the other geometric transformations such as scaling and rotation, and this lack becomes an obstacle to performance improvement even now. This paper proposes a novel network architecture, the *weight-shared multi-stage network* (WSMS-Net), and focuses on acquiring the scale invariance by constructing of multiple stages of CNNs. The WSMS-Net is easily combined with existing deep CNNs, enables existing deep CNNs to acquire a robustness to the scaling, and therefore, achieves higher classification accuracy on CIFAR-10, CIFAR-100 and ImageNet datasets.

Keywords: Image Classification, Scale Invariance, Convolutional Neural Network, Shared Weights

1. Introduction

Convolutional neural networks (CNNs) (LeCun et al., 1989) have made great achievements in the tasks of image classification and image processing (Zeiler and Fergus, 2014; Sermanet et al., 2014). They have already been employed for practical applications in various situations. The CNNs are known to be robust to small parallel shift thanks to their architecture (LeCun et al., 1989): Units in a CNN have their own local receptive fields, share weight parameters with other units, and are sandwiched by pooling layers. A brief history of CNNs can be found in the results of the ImageNet Large Scale Visual Recognition Competition (ILSVRC) (Russakovsky et al., 2014), which is a competition for image processing algorithm using machine learning. After an 8-layer CNN, AlexNet (Krizhevsky et al., 2012), recorded the highest accuracy in ILSVRC 2012, the 19-layer CNN, VGG (Simonyan and Zisserman, 2015), and 22-layer CNN, GoogLeNet (Szegedy et al., 2015), updated the state-of-the-art results in 2014 by deepening the network and increasing the number of internal weight parameters to expand the expressive ability of the network. A further deep neural network model, ResNet (He et al., 2016), recorded the highest accuracy in ILSVRC 2015. This new

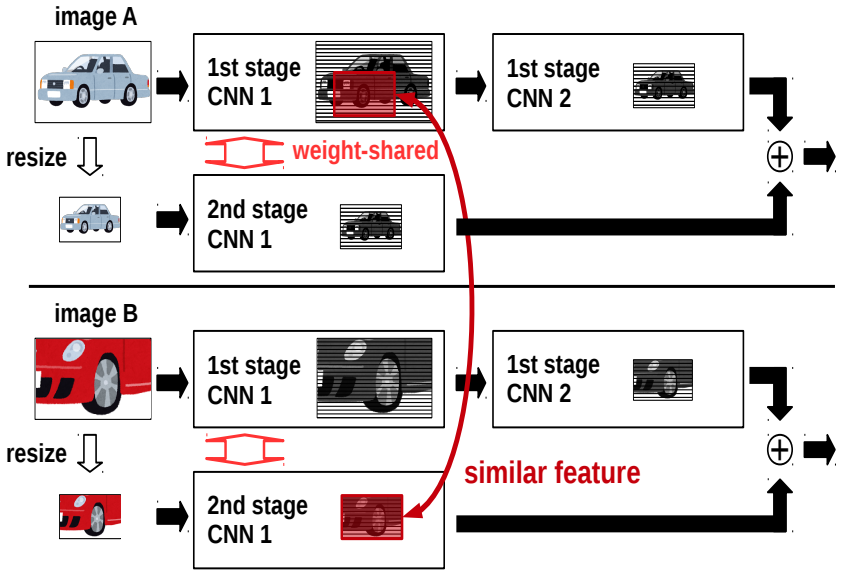


Figure 1: Conceptual explanation of scale invariance in the weight-shared multi-stage network (WSMS-Net). Image A provides the local features of an automobile such as the shapes of the tires, windows, headlights, and license plate in the first stage. Similar features are detected in the resized version of image B, a part of a sports car, in the second stage thanks to the shared weight parameters.

model has more than 100 convolution layers and new shortcut connections that do not perform convolution. This architecture propagates the gradient through the shortcut connections and overcomes the gradient vanishing problem (Schmidhuber, 2015; Bengio, 1994; Glorot and Bengio, 2010), which limits the performance of the deep neural networks. In 2016, a model based on ResNet also broke the record (Zagoruyko and Komodakis, 2016). Many studies have focused on devising the structure of CNNs to prevent the gradient vanishing problem (e.g., DenseNet by Huang et al. (2016a)).

In contrast, CNNs still have a limited robustness to geometric transformations other than a parallel shift, and this problem has no established solution. In addition to parallel shift, the main geometric transformations of objects in images are scaling and rotation (Le et al., 2010). Scaled and rotated objects in images are recognized incorrectly by CNNs or at least require parameters in addition to those for recognizing the original objects, which limits the expressive ability of CNNs. Several previous studies tried to acquire the invariance by combining with geometric transformation layers (Max et al., 2015; Gregor et al., 2015). These layers require the localization of an object, repeated computations for a single image, or computations between all-to-all pairs of pixels. Therefore, they require extensive computation time and are not suitable for combination with deep CNNs.

In this paper, we propose a novel network architecture called the *weight-shared multi-stage network* (WSMS-Net). In contrast to ordinary CNNs, which are built by stacking some convolution layers, a WSMS-Net has a multi-stage architecture consisting of multiple

ordinary CNNs arranged in parallel. A conceptual diagram of the WSMS-Net is depicted in Fig. 1. Each stage of the WSMS-Net consists of all or part of the same CNN: The weight of each convolution layer in each stage is shared with those of the corresponding layer of the other stages. Each stage is given a scaled image of the original input image. The features extracted at all stages are integrated at the *integration layer* and used for image classification. Thanks to this architecture, similar features are extracted, even from objects of different scales at the various stages, and thereby, the same objects of different scales are classified into the same class. We apply the WSMS-Net to existing deep CNNs and evaluate them on the CIFAR-10, CIFAR-100 (Krizhevsky, 2009), and ImageNet (Russakovsky et al., 2014) datasets. The experimental results demonstrate that the use of WSMS-Net significantly improves the classification accuracy of the original deep CNNs by enabling them to acquire scale invariance or at least the robustness to the scaling of objects.

2. Related Works

2.1. Deep CNNs

The CNNs have been introduced with the shared weight parameters, the limited receptive fields, and the pooling layers (LeCun et al., 1989), inspired by the biological architecture of the mammalian visual cortex (Hubel and Wiesel, 1968; Fukushima, 1980). Thanks to this architecture, CNNs suppress increases in the number of weight parameters and are robust to the parallel shift of objects in images. In recent years, CNNs have made remarkable improvements in image classification with the increasingly deeper architectures. However, by stacking more and more convolution layers in CNNs, the gradient vanishing problem arises (Schmidhuber, 2015; Bengio, 1994; Glorot and Bengio, 2010). In general, CNNs use the back-propagation algorithm (LeCun et al., 1989). When the network becomes very deep, the gradient information from the output layer does not pass well to the input layer and other layers near the input, and learning does not progress. Against the vanishing gradient problem, ResNet (He et al., 2016) and DenseNet (Huang et al., 2016a) were proposed as network architectures that enable learning with even deep architecture. From 2015, following ResNet, many ideas and new architectures have been proposed (He and Zhang, 2016; Zagoruyko and Komodakis, 2016; Huang et al., 2016b; Targ et al., 2016; Han et al., 2016). The basic structure of ResNet is called a *residual block*, which is composed of two convolution layers and a shortcut connection that does not pass through the convolution layers: The shortcut connection simply performs identity mapping. The result obtained from the usual convolution layers is denoted by $F(x)$, and the shortcut connection output is denoted by x. The output obtained from the whole block is $F(x) + x$. In deep CNNs, the gradient information can vanish at the feature extraction point in the convolution layer. However, by adding an identity mapping here, the gradient information can be transmitted without any risk of vanishing. ResNet needs no extra parameters, and the gradient can be calculated in the same manner as the conventional CNNs.

DenseNet (Densely Connected Convolutional Network) (Huang et al., 2016a) is a network that improves the concept of ResNet (He et al., 2016). Its image classification accuracy is superior to that of ResNet. As its name implies, the structure of DenseNet connects layers densely. In addition, the output of the shortcut is not added to but is concatenated with the output of a subsequent convolution layer in the channel direction of the feature

map. The number of the channels of the feature map increases linearly as the network is deepened, and the number k of the increased channels per layer is called the *growth rate*. The basic architecture of DenseNet is called a *dense block*, and the whole network of a DenseNet is built by stacking the dense block. The size of the feature map in a single dense block is fixed, and the shortcuts are not connected across each dense block. Instead, a 1×1 convolution layer that does not change the size and the number of channels of the feature map is placed after a dense block for cushioning, followed by a 2×2 pooling layer that thins out the feature map. The main idea of these networks is an addition of extra connections, which propagate the gradient information successfully and enable robust learning even with 100 or more convolution layers.

Although the development of new these networks, CNNs still do not have an established solution to geometric transformations such as scaling and rotation. Lack of invariance to these transformations is an obstacle to the progress of deep CNNs.

2.2. Deep CNNs Robust Against Geometric Transformations

Several studies have tried to address geometrical transformations using neural networks (Max et al., 2015; Gregor et al., 2015). The Spatial Transformer Network (STN) and Deep Recurrent Attentive Writer (DRAW) infer parameters such as position and the angle of the geometric transformation of an object in an image and correct the shape of the object by using neural networks. They are general and powerful approaches to make a network robust against geometric transformations. However, the STN requires additional CNNs to localize and correct an object in an image. DRAW requires repeated computations between all-to-all pairs of pixels of the input and output images to adjust parameters gradually for each image. They need many additional parameters and computation time and therefore are not suitable for combining with deep CNNs.

3. Weight-Shared Multi-Stage Network

3.1. Weight-Shared Multi-Stage Network Architecture

We propose a novel network architecture called *weight-shared multi-stage network* (WSMS-Net) to acquire robustness to object scaling. The basic architecture of WSMS-Net is shown in Fig. 2. A WSMS-Net consists of S stages, and each stage is just all or part of an ordinary CNN that can be divided into t convolution blocks for $t \geq S$. Each convolution block consists of some convolution layers. After each convolution block, the feature map is resized to half at a *downsizing* layer. The downsizing layer is typically a max pooling (Riesenhuber and Poggio, 1999), average pooling (LeCun et al., 1989), or convolution layer with a stride of 2, but this depends on the original CNN: ResNet (He et al., 2016) employs a convolution layer with a stride of 2 and DenseNet (Huang et al., 2016a) employs trans layer consisting of a convolution layer with a stride of 1 followed by a 2×2 average pooling layer with a stride of 2, as mentioned above. The first stage consists of all t blocks while the second stage has the first $(t-1)$ blocks, the third stage has the first $(t-2)$ blocks, and so on. The convolution blocks at the same depth of all the stages completely share the weight parameters of the convolution layers. Note that when batch normalization (Ioffe and Szegedy, 2015) is employed, its parameters are not shared. The first stage

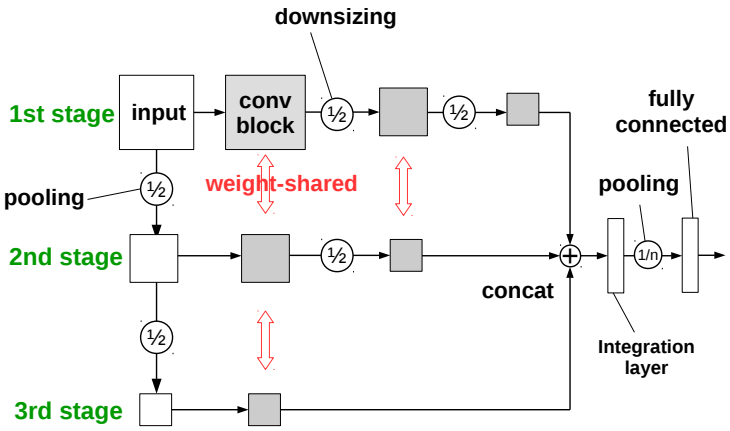


Figure 2: Basic WSMS-Net consists of 3 stages. Each *conv block* consists of some convolution layers and it is followed by a *downsizing* layer, at which the feature map sizes are reduced to half. The *weight-shared* label indicates that the weight parameters of all the convolution layers in the same block in each stage are shared with each other. At *concat*, the outputs of all the stages are concatenated in the channel direction.

is given the original image, while the second stage is given the image resized by half by a pooling layer. Similarly, the s-th stage is given the image resized to half of the size of the image given to the $(s-1)$-th stage. Therefore, the sizes of the output feature maps of all the stages are the same. Then, these feature maps are concatenated in the channel direction at the ends of all the stages. After concatenation, all of the feature maps are integrated at the integration layer and are used as input to the last fully connected layer for the classification process. Thanks to these processes, various features are extracted from the input image of multiple sizes in multiple stages and hence contribute to the classification.

3.2. Combination with Existing CNNs

By combining with the WSMS-Net, ResNet (He et al., 2016) and DenseNet (Huang et al., 2016a) are expected to classify the images that could not be dealt with by the original CNNs. In this paper, WSMS-Net combined with ResNet is called *WSMS-ResNet*, and WSMS-Net combined with DenseNet is called *WSMS-DenseNet*. The number of stages and the shape of the integration layer should be optimized to construct WSMS-ResNet and WSMS-DenseNet. The integration layer is an extra layer placed just after the concatenation layer that concatenates all the feature maps from all the stages, and integrates all the feature maps before the classification. For comparison, we also introduce a more trivial network called the *multi-stage network* (MS-Net). A MS-Net has multiple stages like a WSMS-Net, but each stage has weight parameters, similar to an ensemble of multiple CNNs (e.g., Zhang et al. (2016)). MS-Net has many more parameters than WSMS-Net but does not have the features of WSMS-Net described above.

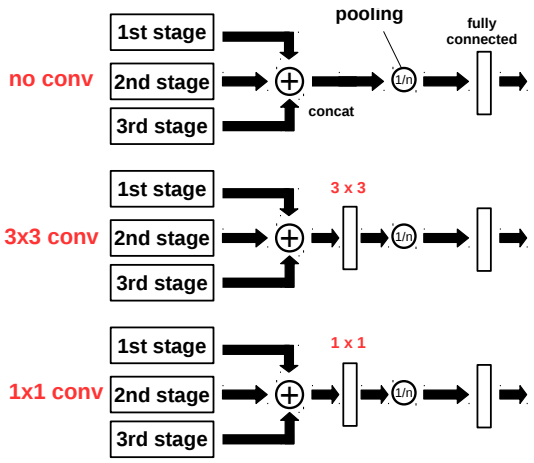


Figure 3: Three types of integration layers. With *no conv*, the concatenated feature map is directly given to the global pooling layer followed by the last fully connected layer. 3×3 *conv* and 1×1 *conv* are literally a 3×3 convolution layer and a 1×1 convolution layer, placed between the concatenated feature map and the global pooling layer.

3.3. Integration Layer

We have not yet described the integration layer in detail. In WSMS-Net, the feature maps obtained from all the stages are concatenated and are given to the integration layer before the global pooling layer and the last fully connected layer. We consider several candidates for an integration layer as shown in Fig. 3. In the simplest way, the integration layer does nothing, and the last fully connected layer is given a feature vector that is longer than that of the original network. This integration layer is called *no conv* hereafter. Otherwise, the integration layer can be a convolution layer with a $k \times k$ kernel and a c output channels. We evaluated a 3×3 convolution layer as the integration layer because both ResNet and DenseNet mainly employ 3×3 convolution layers. This integration layer is also called 3×3 *conv*. We also evaluated a 1×1 convolution layer as the integration layer called 1×1 *conv*.

4. Experiments

4.1. CIFAR-10 and CIFAR-100 Classification by WSMS-DenseNet

WSMS-DenseNet for CIFAR-10 and CIFAR-100 In this section, we combined a DenseNet (Huang et al., 2016a) with a growth rate $k = 24$ with WSMS-Net to constructed a WSMS-DenseNet, and evaluated WSMS-DenseNet for CIFAR-10 and CIFAR-100 datasets (Krizhevsky, 2009). CIFAR-10 and CIFAR-100 are datasets of 32×32 RGB images of natural scene objects: Each consists of 50,000 training images and 10,000 test images. Each image is manually given one of 10 class labels in CIFAR-10, while each image is given one of 100 class labels in CIFAR-100: The number of images per class is reduced in CIFAR-100. The DenseNet ($k = 24$) was reported as the model achieving the second

Table 1: Test Error Rates of WSMS-DenseNet on the CIFAR-10/CIFAR-100.

		C10+		C100+	
network	growth rate	#params	Error(%)	#params	Error(%)
DenseNet	24	27.2M	3.74	27.2M	19.25
DenseNet	26	31.9M	3.82	31.9M	18.94
MS-DenseNet (1×1 conv)	24	41.3M	4.18	41.3M	**18.70**
WSMS-DenseNet (no conv)	24	27.4M	4.54	27.8M	20.11
WSMS-DenseNet (3×3 conv)	24	32.7M	**3.54**	32.7M	**19.16**
WSMS-DenseNet (1×1 conv)	24	28.0M	**3.51**	28.0M	**18.45**

highest classification accuracy among the results of CIFAR-10 and CIFAR-100 datasets in the original study. The original DenseNet ($k = 24$) has three dense blocks, and the sizes of the feature map are 32×32, 16×16, and 8×8 in the first, second, and third dense blocks, respectively. There are 3 channels at the input, which is increased to 16 channels by the convolution layer placed before the first dense block. Moreover, the growth rate k is increased by 24 at every convolution layer. Each dense block consists of 32 convolution layers. Therefore, the total number of the channels is $16 + 24 \times 32 \times 3 = 2,320$. After the third dense block, global average pooling is performed, resulting in a 2,320-dimensional feature vector that is given to the last fully connected layer for classification.

We constructed WSMS-DenseNet of growth rate $k = 24$ by combining the DenseNet with WSMS-Net. An overall diagram of WSMS-DenseNet is depicted in Fig. 4. Each dense block of the DenseNet was treated as a convolution block of the WSMS-Net, and we set the number of stages to 3. The final numbers of the channels for the first, second, and third stages were $16 + 24 \times 32 \times 3 = 2,320$, $16 + 24 \times 32 \times 2 = 1,552$, and $16 + 24 \times 32 \times 1 = 784$, respectively. We set the number c of the channels of the final feature map to 128: The feature map was projected to 8×8 feature maps of 128 channels through the integration layer of 3×3 conv or 1×1 conv before the global pooling layer. The hyperparameters and the other structure parameters of the WSMS-DenseNet followed those of the original DenseNet ($k = 24$). All the images were normalized with the mean and variance of each filter, and 4 pixels padding on each side, 32×32 cropping and random flipping in the horizontal direction were employed as further data normalization and data augmentation (Lee et al., 2015; Romero et al., 2015; Springenberg et al., 2015). Batch normalization (Ioffe and Szegedy, 2015) and ReLU activation function (Nair and Hinton, 2010) were used. The weight parameters were initialized following the algorithm proposed in the reference (He et al., 2016). WSMS-DenseNet was trained using the momentum SGD algorithm with the momentum parameter of 0.9 and the weight decay of 10^{-4} over 300 epochs. The learning rate was initialized to 0.1, and then, it was reduced to 0.01 and 0.001 at the 150th and the 225th epochs, respectively. Note that, of the experimental condition, only the mini-batch size was changed (from 64 to 32) owing to the capacity of the computer we used: This change could potentially degrade the classification accuracy of the WSMS-DenseNet.

Classification Results Table 1 summarizes the results of the WSMS-DenseNet ($k = 24$) and the original DenseNets. For CIFAR-10, the error rate of our proposed WSMS-DenseNet with the 1×1 conv integration layer is 3.51 %, which is better than the error rate of 3.74 % obtained by the original DenseNet ($k = 24$). However, the number of parameters of

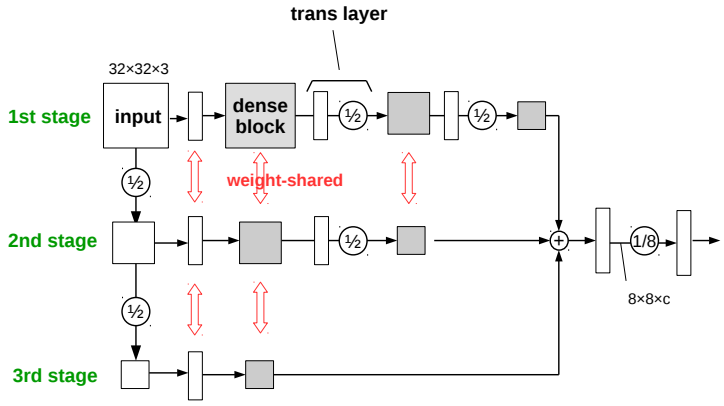
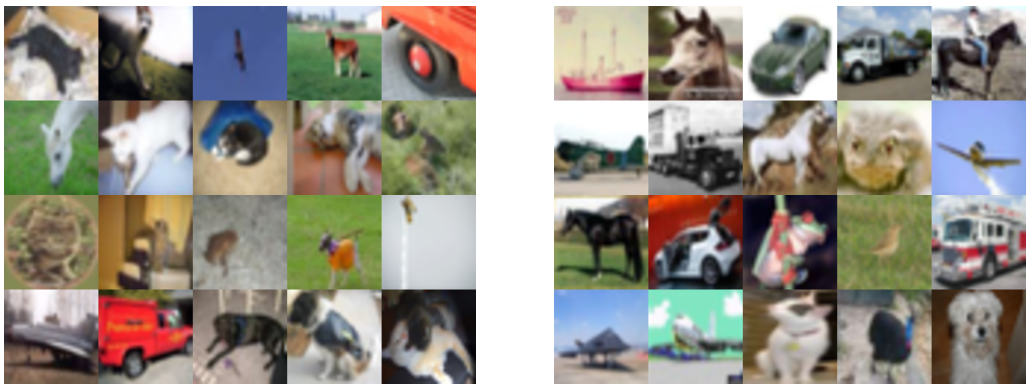


Figure 4: WSMS-DenseNet for CIFAR-10 and CIFAR-100 classification. The input is 32×32 RGB images ($32 \times 32 \times 3$). The *dense block* is dense architecture of DenseNet. The sizes of the feature maps are same in each dense blocks. The *trans layer* consists of an 1×1 convolution layer with a stride of 1 and a 2×2 average pooling layer for downsizing the feature maps. After integration layer, $8 \times 8 \times c$ feature maps are given to the fully connected layer for classification.



(a) (b)

Figure 5: (a) Examples of CIFAR-10 test images misclassified by the DenseNet ($k = 24$) and the DenseNet ($k = 26$) but classified by the WSMS-DenseNet ($k = 24$, 1×1 conv) correctly; (b) Examples of CIFAR-10 test images.

WSMS-DenseNet was increased from 27.2M to 28.0M. For fair comparison, we also evaluated DenseNet ($k = 26$). In spite of the larger increase in the number of parameters, DenseNet ($k = 26$) achieved a worse error rate of 3.82 %. These results demonstrate that an increase in the number of parameters has a limit in the improvement of classification accuracy. In contrast, our proposed WSMS-Net enables the original DenseNet to achieve better classification accuracy. The MS-DenseNet achieved an error rate that was worth than

those of the original DenseNet and WSMS-DenseNet in spite of a massive increase in the number of parameters, indicating that the improvement of the WSMS-DenseNet is thanks to the shared parameters rather than the network architecture. Even though the WSMS-DenseNet has a limited increase in the number of parameters thanks to shared weight parameters than the MS-DenseNet, both the WSMS-DenseNet and MS-DenseNet have an equal increase in the number of calculations owing to the second and third stages. Therefore, we evaluated the number of multiplications over all convolution layers in the original DenseNet and WSMS-DenseNet. The number of multiplications is about 6,889M for the original 100-layer DenseNet and 8,454M for the 101-layer WSMS-DenseNet with the 1×1 conv integration layer. Hence, our proposed WSMS-DenseNet incurs only a 20 % increase in the number of multiplications. Also for CIFAR-100, our proposed WSMS-DenseNet with the 1×1 conv integration layer achieved an accuracy that was 18.45 % than the original DenseNet. DenseNet ($k = 26$) and the MS-DenseNet also achieved accuracies better than that of the original DenseNet ($k = 24$) but worse than that of the WSMS-DenseNet.

To qualitatively evaluate our proposed WSMS-Net, we collected test images from CIFAR-10 that DenseNet ($k = 24$) and DenseNet ($k = 26$) misclassified but WSMS-DenseNet ($k = 24$) classified correctly. Random examples of such test images are shown in Fig. 5(a). Many images in Fig. 5(a) show close-up views of objects with large portions of object cropped, or they show objects taken against backgrounds in long shot. For comparison, example images were randomly chosen from all the test images, as shown in Fig. 5(b). A Comparison between the two groups demonstrates that the WSMS-Net additionally classifies images showing scaled objects. We conclude that the WSMS-Net has acquired the scale invariance or is at least more robust to object scaling, and therefore, it achieves better classification accuracy than the original CNNs in spite of the limited increase in the number of weight parameters.

4.2. CIFAR-10 and CIFAR-100 Classification by WSMS-ResNet

WSMS-ResNet for CIFAR-10 and CIFAR-100 This section evaluates ResNet (He et al., 2016) with WSMS-Net for CIFAR-10 and CIFAR-100 (Krizhevsky, 2009) in the same manner as WSMS-DenseNet. The 110-layer ResNet was reported as the model achieving the highest accuracy for CIFAR-10 when the reference (He et al., 2016) was published. The 110-layer ResNet has three compartments and each compartment consists of 18 residual blocks. Sizes of the feature maps are 32×32, 16×16, and 8×8 in the first, second, and third compartments, respectively. The number of the channels of the feature map is 3 at the input and increased to 16, 32, and 64 by the convolution layers placed before the first, second, and third compartments, respectively. After the third compartment, global average pooling is performed on the 8×8 feature map of 64 channels, resulting in an 1×1 feature map of 64 channels, i.e., a 64-dimensional feature vector. This feature vector is given to the final fully connected layer for classification.

We constructed WSMS-ResNet by combining the 110-layer ResNet with WSMS-Net and we set the number of stages 3. An overall diagram of WSMS-ResNet is depicted in Fig. 6 The final numbers of the channels for the first, second, and third stages were 64, 32, and 16, respectively. The integration layer was given the feature map of $64+32+16 = 112$ channels. With the integration layer of no conv, the feature maps became an 112-dimensional feature

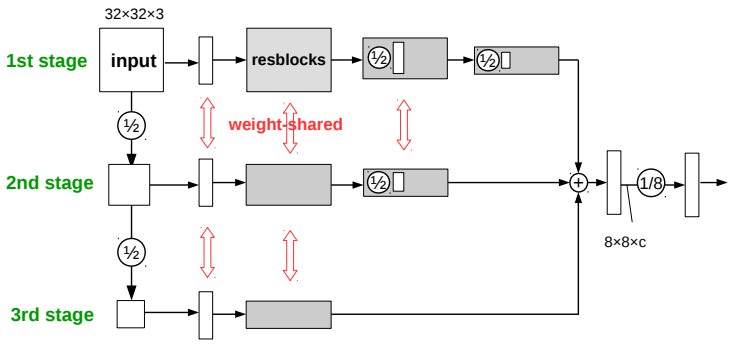


Figure 6: WSMS-ResNet for CIFAR-10 and CIFAR-100 classification. The *resblocks* consists of a mass of residual blocks (resblock means residual block). The sizes of feature maps are same in each mass of resblocks. In 2nd block and 3rd block of 1st stage, and in 2nd block of 2nd stage, convolution layers with strides of 2 are used for downsizing of feature maps. These types of a mass of resblock represented as rectangle having $\frac{1}{2}$ circle inside. After integration layer, $8 \times 8 \times 128$ feature maps are put in fully connected layer after global pooling layer for classification.

vector through the global average pooling layer. We set c, the number of channels of the final feature map, to 128. The hyperparameters and other conditions of WSMS-ResNet followed those used in the original study. The WSMS-ResNet was trained using the momentum SGD algorithm with the momentum parameter of 0.9, the mini-batch size of 128, and the weight decay of 10^{-4} over 164 epochs. The learning rate was set to 0.01, and then, it was changed to 0.1, 0.01, and 0.001 at the 2nd, the 82nd, and the 123rd epochs, respectively.

Classification Results Table 2 summarizes the results of the WSMS-ResNet and the original ResNet. Since the original study (He et al., 2016) did not report the results for CIFAR-100, we evaluated the 110-layer ResNet for CIFAR-100 in addition to the WSMS-ResNet for comparison. Following the original study (He et al., 2016), we obtained the mean test error rate of five trials. For CIFAR-10, the error rate of our proposed WSMS-ResNet with the 1×1 conv integration layer is 6.36 %: This accuracy is obviously superior to the error rate of 6.61 % obtained by the original ResNet. However, the number of parameters of WSMS-ResNet was increased from 1.73M to 1.75M. For fair comparison, we also evaluated the deeper ResNets of 116-layer and 122-layer. In spite of the increase in the number of parameters, the deeper ResNets achieved the accuracies at a similar level to the 110-layer ResNet. These results demonstrate that the increase in the number of parameters along with the increase in the depth has a limit in the improvement of classification accuracy. In contrast, our proposed WSMS-Net enables the original ResNet to achieve better classification accuracy. The MS-ResNet achieved an error rate better than those of the original ResNet but worse than those of the WSMS-ResNet in spite of a massive increase in the number of parameters, indicating that the improvement of WSMS-ResNet

Table 2: Test Error Rates of WSMS-ResNet on the CIFAR-10/CIFAR-100.

		C10+		C100+	
Network	depth	#params	Error (%)	#params	Error (%)
ResNet	110	1.73M	6.61	1.73M	28.77
ResNet	116	1.82M	6.60	1.83M	28.34
ResNet	122	1.92M	6.57	1.93M	28.39
MS-ResNet (1×1 conv)	111	2.23M	**6.41**	2.25M	**27.23**
WSMS-ResNet (no conv)	110	1.73M	7.23	1.74M	**28.72**
WSMS-ResNet (3×3 conv)	111	1.86M	**6.41**	1.87M	**27.16**
WSMS-ResNet (1×1 conv)	111	1.75M	**6.36**	1.76M	**27.45**

is thanks to the shared parameters rather than the network architecture. The number of multiplications is about 252M in the original 110-layer ResNet and 301M for the 111-layer WSMS-ResNet with the 1×1 conv integration layer. Hence, our proposed WSMS-ResNet incurs only a 20 % increase in the number of multiplications in the same case as WSMS-DenseNet.

In the case of CIFAR-100, the WSMS-ResNet with 1×1 conv integration layer also surpassed the original ResNet. Unlike the case of CIFAR-10, the deeper ResNets and the MS-ResNet achieved better results than those of the original ResNet, and the WSMS-ResNet with 3×3 conv integration layer achieved the best result. We consider that the difference between CIFAR-10 and CIFAR-100 is caused by the complexity of classification task. Classification of CIFAR-100 is more difficult because of the larger number of classes and the limited numbers of samples, and thus, requires much more weight parameters than the classification of CIFAR-10. Therefore, the WSMS-ResNet with 3×3 conv integration layer, having more weight parameters, is simply more advantageous than the WSMS-ResNet with 1×1 conv integration layer. The deeper ResNet and MS-ResNet are also better than the original ResNet thanks to their large numbers of weight parameters. Anyway, our proposed WSMS-Net enables the original ResNet to achieve a better classification accuracy in spite of the limited increase in the parameters even for CIFAR-100.

4.3. ImageNet classification by WSMS-ResNet

WSMS-ResNet for ImageNet This section evaluates ResNet (He et al., 2016) with WSMS-Net for the ImageNet 2012 classification dataset (Russakovsky et al., 2014). ImageNet consists of 1.28 million training images and 50,000 validation images. Each image is given one of 1,000 class labels. We used the 50-layer, 101-layer and 152-layer ResNets with bottleneck architecture. To downsize the feature map at the entrance of each compartment, a ResNet with the bottleneck architecture modifies the stride of the second of the three sequential convolution layers to 2, and replaces the shortcut with a 1×1 convolution layer with a stride of 2, while the original ResNet employs pooling layers.

The ResNet is given an $N \times M$ RGB input image. The input image becomes an $\frac{N}{4} \times \frac{M}{4}$ feature map of 64 channels through a 7×7 convolution layer with a stride of 2 and a subsequent 3×3 max pooling layer with a stride of 2. The remaining part of the ResNet has four compartments, each consisting of several residual blocks with bottleneck architecture. The size of the feature maps are $\frac{N}{4} \times \frac{M}{4}$, $\frac{N}{8} \times \frac{M}{8}$, $\frac{N}{16} \times \frac{M}{16}$, and $\frac{N}{32} \times \frac{M}{32}$, and the number of channels of the feature maps are 256, 512, 1,024, and 2,048 for the first, second, third,

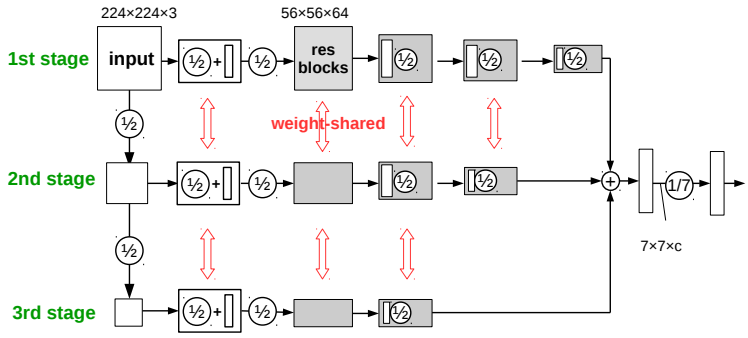


Figure 7: WSMS-ResNet for ImageNet classification. The input is a cropped 224×224 RGB image ($224 \times 224 \times 3$) in this experiment. At the first convolution block in each stage, a 7×7 convolution layer with a stride of 2 and a max pooling layer result in the feature map of $56 \times 56 \times 64$. The *resblocks* consists of many residual blocks. The sizes of the feature maps are fixed through the same compartments. At the entrances of the second and following blocks, convolution layers with strides of 1 and 2×2 average pooling layers are used for downsizing of the feature maps; They are represented as the rectangles having $\frac{1}{2}$ circles and convolution layers inside. After the integration layer, the $7 \times 7 \times c$ feature map is given to the global pooling layer followed by the fully connected layer for classification.

and fourth compartments, respectively. After the fourth compartment, global pooling is performed, resulting in 1×1 feature map of 2,048 channels. This feature vector is given to the final fully connected layer for 1,000-class classification. The number of residual blocks are 3, 4, 6, and 3 in the first, second, third and forth compartments of the 50-layer ResNet, 3, 4, 23, and 3 in the 101-layer ResNet, and 3, 8, 36, and 3 in the 152-layer ResNet.

We constructed the WSMS-ResNets by combining the WSMS-Net with 50-layer, 101-layer, and 152-layer ResNets (He et al., 2016) for the ImageNet dataset with WSMS-Net. An overall diagram of the WSMS-ResNet for the ImageNet classification task is depicted in Fig. 7. The number of stages was set to three. The first stage was just the same as the original ResNet before the global pooling layer. An $N \times M$ input image was downsized to $\frac{N}{2} \times \frac{M}{2}$ for the second stage, and was downsized to $\frac{N}{4} \times \frac{M}{4}$ for the third stage. The second stage was composed of the first three compartments of the original ResNet, in which the size of the feature map was $\frac{N}{8} \times \frac{M}{8}$, $\frac{N}{16} \times \frac{M}{16}$, and $\frac{N}{32} \times \frac{M}{32}$ in the first, second, and third blocks, respectively. In addition, the third stage was composed of the first two compartments, using the $\frac{N}{16} \times \frac{M}{16}$ and $\frac{N}{32} \times \frac{M}{32}$ feature maps. The integration layer was given a $\frac{N}{32} \times \frac{M}{32}$ feature map of $2,048 + 1024 + 512 = 3,584$ channels. We set c, the number of channels of the final feature map, to 2,048. Note that the WSMS-ResNets used convolution layers with strides of 1 followed by 2×2 average pooling layers instead of convolution layers with strides of 2 as depicted in Fig. 8. This experiment evaluated only the integration layer of 1×1 conv because of the results in the previous sections. The hyperparameters and other condi-

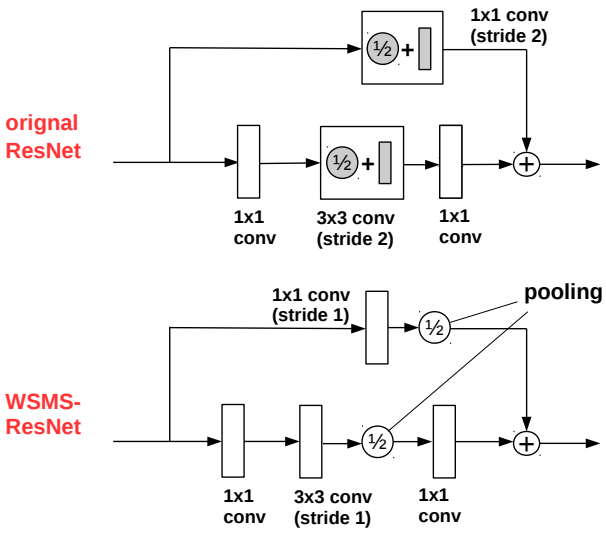


Figure 8: Difference in downsizing between the original ResNets and our proposed WSMS-ResNets. The original ResNets use 3×3 convolution layers with strides of 2 for downsizing the feature maps, i.e., 2×2 pooling layers followed by convolution layers with strides of 1. On the other hand, WSMS-ResNets use 3×3 convolution layers with strides of 1 followed by 2×2 average pooling layers.

tions of WSMS-ResNets followed those used in the original study and the reimplementation by Facebook AI Research posted on https://github.com/facebook/fb.resnet.torch. Batch normalization and the ReLU activation function were used. The weight parameters were initialized following the algorithm proposed in He et al. (2016). The WSMS-ResNets were trained using the momentum SGD algorithm with a momentum parameter of 0.9, mini-batch size of 256, and weight decay of 10^{-4} over 90 epochs. The learning rate was initialized to 0.1 and reduced to 0.01 and 0.001 at the 30th and 60th epochs, respectively.

Classification Results Table 3 summarizes the results of the WSMS-ResNets and original ResNets for ImageNet classification. The classification results of the original ResNets were cited not from the original study (He et al., 2016) but from the reimplementation by Facebook AI Research posted on https://github.com/facebook/fb.resnet.torch. Note that Facebook's results are better than those shown in the original study (He et al., 2016). According to Table 3, our proposed 51-layer, 102-layer, and 153-layer WSMS-ResNets achieved better results than original 50-layer, 101-layer, and 152-layer ResNets, respectively, with respect to both top-1 and top-5 error rates. The WSMS-ResNets for ImageNet classification have more parameters than the original ResNets. However, the 102-layer WSMS-ResNet with 52.0M parameters is superior to the original 152-layer ResNet with 60.2M parameters. This is an evidence that our proposed WSMS-Net improves the original CNN while maintaining a limited increase in the number of weight parameters.

Table 3: Single Crop Test Error Rates of WSMS-ResNets on the ImageNet.

Network	depth	#params	top-1 Error(%)	top-5 Error(%)
ResNet	50	25.6M	24.01	7.02
WSMS-ResNet (1×1 conv)	51	32.9M	**23.00**	**6.45**
ResNet	101	44.5M	22.44	6.21
WSMS-ResNet (1×1 conv)	102	52.0M	**21.91**	**6.03**
ResNet	152	60.2M	22.16	6.16
WSMS-ResNet (1×1 conv)	153	67.7M	**21.78**	**5.89**

All the results for the CIFAR-10, CIFAR-100, and ImageNet classification tasks demonstrate that a combination with our proposed WSMS-Net contributes to the various datasets and architectures of CNNs.

5. Conclusion

In this study, we proposed a novel network architecture for convolutional neural networks (CNNs) called the *weight-shared multi-stage network* (WSMS-Net) to improve classification accuracy by acquiring the robustness to object scaling. The WSMS-Net consists of multiple stages of CNNs given input images of different sizes. All the feature maps obtained from all the stages are concatenated and integrated at the ends of the stages. The increases in the number of weight parameters and computations were limited. The experimental results demonstrated that the WSMS-Net achieved better classification accuracy and had higher robustness to object scaling than existing CNN models, while deepening the network is a limited way to improve classification accuracy. Future works include a more detailed analysis and experiments how our method works; e.g., visualizing the feature maps in different stages to ensure that our method can acquire multi-scale features in network, evaluating our method on controlled datasets containing many scales of images to test the ability to handle multi-scale objects in detail.

Acknowledgments

This study was partially supported by the JSPS KAKENHI (16K12487), Kayamori Foundation of Information Science Advancement, and SEI Group CSR Foundation.

References

Y Bengio. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks 5(2)*, pages 157–166, 1994.

Kunihiko Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, 36(4): 193–202, 1980.

Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proc. of the 13th International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 9, pages 249–256, 2010.

K Gregor, I Danihelka, A Graves, and D Wierstra. DRAW: A Recurrent Neural Network For Image Generation. *arXiv*, pages 1–11, 2015.

Dongyoon Han, Jiwhan Kim, and Junmo Kim. Deep Pyramidal Residual Networks. *arXiv*, pages 1–9, 2016.

Kaiming He and Xiangyu Zhang. Identity mappings in deep residual networks. *Lecture Notes in Computer Science*, 9908(1):630–645, 2016.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *Proc. of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR2016)*, pages 770–778, 2016.

Gao Huang, Zhuang Liu, and Kilian Q. Weinberger. Densely Connected Convolutional Networks. *arXiv*, pages 1–12, 2016a.

Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Weinberger. Deep Networks with Stochastic Depth. In *Proc. of European Conference on Computer Vision (ECCV2016)*, volume 9905, pages 646–661, 2016b.

D H Hubel and T N Wiesel. Receptive fields and functional architecture of monkey striate cortex. *The Journal of Physiology*, 195(1):215–43, 1968.

Sergey Ioffe and Christian Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. *arXiv*, pages 1–11, 2015.

Alex Krizhevsky. Learning Multiple Layers of Features from Tiny Images. *Technical report, University of Toronto*, pages 1–60, 2009.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. ImageNet Classification with Deep Convolutional Neural Networks. In *Advances in Neural Information Processing Systems (NIPS2012)*, pages 1097–1105, 2012.

Qv Le, Jiquan Ngiam, Zhenghao Chen, Dj Hao Chia, and Pw Koh. Tiled convolutional neural networks. In *Advances in Neural Information Processing Systems (NIPS2010)*, pages 1279–1287, 2010.

Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation Applied to Handwritten Zip Code Recognition. *Neural Computation*, 1(4):541–551, 1989.

C Y Lee, S Xie, P Gallagher, Z Zhang, and Z Tu. Deeply-supervised nets. In *Proc. of the 18th International Conference on Artificial Intelligence and Statistics (AISTATS2015)*, volume 2, pages 562–570, 2015.

Jaderberg Max, Karen Simonyan, Andrew Zisserman, and Koray Kavukcuoglu. Spatial Transformer Networks. In *Advances in Neural Information Processing Systems (NIPS2015)*, pages 2017–2025, 2015.

Vinod Nair and Geoffrey E Hinton. Rectified Linear Units Improve Restricted Boltzmann Machines. In *Proc. of the 27th International Conference on Machine Learning (ICML2010)*, number 3, pages 807–814, 2010.

M. Riesenhuber and T. Poggio. Hierarchical models of object recognition in cortex. *Nature Neuroscience*, 2(11):1019–1025, 1999.

Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. Fitnets: Hints for Thin Deep Nets. In *Proc. of International Conference on Learning Representations (ICLR2015)*, pages 1–13, 2015.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, and C V Jan. ImageNet Large Scale Visual Recognition Challenge. *arXiv*, pages 1–43, 2014.

Juergen Schmidhuber. Deep Learning in neural networks: An overview. *Neural Networks*, 61:85–117, 2015.

Pierre Sermanet, David Eigen, Xiang Zhang, Michael Mathieu, Rob Fergus, and Yann LeCun. OverFeat: Integrated Recognition, Localization and Detection using Convolutional Networks. In *Proc. of International Conference on Learning Representations (ICLR2014)*, pages 1–16, 2014.

Karen Simonyan and Andrew Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. In *Proc. of International Conference on Learning Representations (ICLR2015)*, pages 1–14, 2015.

Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin Riedmiller. Striving for Simplicity: The All Convolutional Net. In *Proc. of International Conference on Learning Representations (ICLR2015)*, pages 1–14, 2015.

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proc. of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR2015)*, pages 1–9, 2015.

Sasha Targ, Diogo Almeida, and Kevin Lyman. ResNet In ResNet: Generalizing Residual Architectures. In *Workshop on International Conference on Learning Representations (ICLR2016)*, number 1, pages 1–4, 2016.

Sergey Zagoruyko and Nikos Komodakis. Wide Residual Networks. *arXiv*, pages 1–15, 2016.

Matthew D. Zeiler and Rob Fergus. Visualizing and Understanding Convolutional Networks. In *Proc. of European Conference on Computer Vision (ECCV2014)*, pages 818–833, 2014.

Yingying Zhang, Desen Zhou, Siqin Chen, Shenghua Gao, and Yi Ma. Single-Image Crowd Counting via Multi-Column Convolutional Neural Network. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR2016)*, pages 589–597, 2016.