Communications for Statistical Applications and Methods
2022, Vol. 29, No. 2, 161–176

https://doi.org/10.29220/CSAM.2022.29.2.161
Print ISSN 2287-7843 / Online ISSN 2383-4757

A review and comparison of convolution neural network models under a unified framework

Jimin Park[a], Yoonsuh Jung[1,b]

[a]Memory Business, Samsung Electronics, Korea;
[b]Department of Statistics, Korea University, Korea

Abstract

There has been active research in image classification using deep learning convolutional neural network (CNN) models. ImageNet large-scale visual recognition challenge (ILSVRC) (2010-2017) was one of the most important competitions that boosted the development of efficient deep learning algorithms. This paper introduces and compares six monumental models that achieved high prediction accuracy in ILSVRC. First, we provide a review of the models to illustrate their unique structure and characteristics of the models. We then compare those models under a unified framework. For this reason, additional devices that are not crucial to the structure are excluded. Four popular data sets with different characteristics are then considered to measure the prediction accuracy. By investigating the characteristics of the data sets and the models being compared, we provide some insight into the architectural features of the models.

Keywords: classification, convolutional neural network (CNN), ImageNet large-scale visual recognition challenge (ILSVRC), image data

1. Introduction

LeCun *et al.* (1989) first introduced convolutional neural network (CNN), which had one of the representative deep learning architectures widely used in image processing these days. Despite CNN's success, the artificial neural network was not suitable in practice due to the gradient vanishing problem. That is, the deeper the neural network, the slower the learning speed. Thus, model fitting was not satisfactory. This issue prevented researchers from learning by adding many hidden layers. For this reason, the area of the artificial neural network had been underdeveloped until the mid-2000s.

The ImageNet large scale visual recognition challenge (ILSVRC) is an annual competition for image classification held between 2010 and 2017. In this competition, participants are asked to categorize images of ImageNet (Deng *et al.*, 2009), a large number of visual data databases. A training data set with image labels is provided along with an unlabeled test data set. Competitors predict the label in the test data. An evaluation server then releases the classification accuracy at the end of the competition.

There have been various technical attempts to improve CNN. Using ReLU (Nair and Hinton, 2010) for an activation function of layers solved the problem of gradient vanishing. AlexNet (Iandola

Jung's work has been partially supported by National Research Foundation of Korea (NRF) grants funded by the Korea government(MIST) 2019R1A4A1028134 and 2021R1F1A1062347.

[1] Corresponding author: Department of Statistics, Korea University, 145 Anam-ro, Seongbuk-gu, Seoul 02841, South Korea. E-mail: yoons77@korea.ac.kr

Published 31 March 2022 / journal homepage: http://csam.or.kr

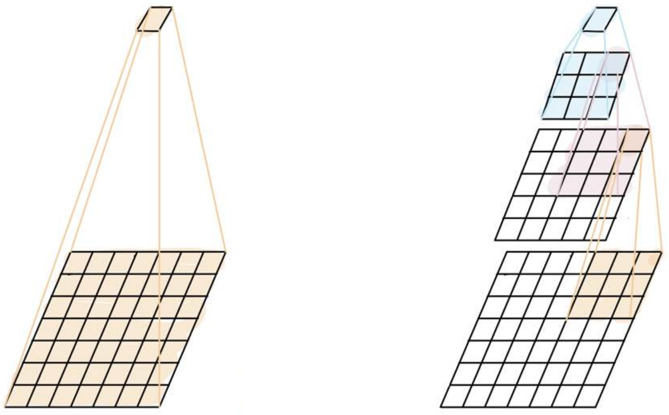

(a) *One stack of* 7×7 *filter* (b) *Three stacks of* 3×3 *filters*

Figure 1: *Three* 3×3 *layers perform the same function as one* 7×7 *layer.*

et al., 2016) disseminates the utilization of GPUs to boost the computing speed. The improved artificial neural network showed higher performance in various fields compared to conventional machine learning methods. After AlexNet won the ILSVRC in 2012 by a large gap compared with other existing models, several seminal models were developed. The models that won the ILSVRC in each year with their ingenious idea caused numerous follow-up studies.

In this paper, we focus on these influential models developed since AlexNet. For this reason, we select six models that demonstrated high performance with a unique structure in ILSVRC; They are VGG (Simonyan and Zisserman, 2014), ResNet (He *et al.*, 2016), Wide residual networks (Zagoruyko and Komodakis, 2016), NASNet (Zoph *et al.*, 2018), Mobilenet (Howard *et al.*, 2017), and ShuffleNet (Zhang *et al.*, 2018). For the models to be comparable, we attempt to modify their original structure to unify model complexity. ResNet, MobileNet, ShuffleNet are one group using about one million parameters. The others are rather complicated models employing over two million parameters. So, the modified models can be compared within each group, while maintaining their unique features of the architectures. We provide details of one variant that are closely related to the model we implement.

We apply these unified models to the four popular data sets, CIFAR-10 (Krizhevsky *et al.*, 2014), CIFAR-100 (Netzer *et al.*, 2011), Fashion-MNIST (Xiao *et al.*, 2017), and SVHN (Netzer *et al.*, 2011), and measure their prediction accuracy. We believe this paper provides not only a review but also a better understanding of the CNN models.

Following is the outline of this paper. We introduce the characteristics of each model in Section 2. Section 3 illustrates the details of the unified framework. Section 4 provides their application to the real-world data and our findings.

2. Review of convolutional neural network (CNN) models

2.1. VGGNet

Simonyan and Zisserman (2014) investigated the effect of network depth on large-scale image data using VGGNet. VGGNet uses a 3×3 filter in contrast to other popular models such as GoogLeNet (Szegedy *et al.*, 2015) and AlexNet (Iandola *et al.*, 2016), which employ 7×7 or 11×11 size filters. VGGNet dramatically improved the accuracy of image classification even with the small filter. Figure

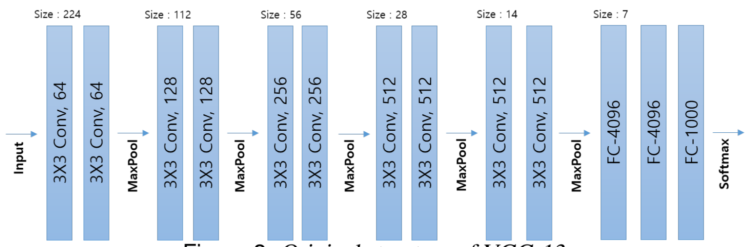


Figure 2: *Original structure of VGG-13.*

1 illustrates how three 3×3 layers work exactly the same as one 7×7(We use the expression ‘filter’ and ‘layer’ interchangeably as both the terms are commonly used).

There are two advantages when using three 3×3 filters. First, the decision function becomes more distinguishable because it can use the nonlinear function three times more. Second, the number of parameters in the model is reduced. 3×3 layer can achieve the same effect of one 7×7(11×11) layer using 55%(37%) number of parameters. There are largely five variants of VGG structure varying the model’s depth from 11 to 19 and the number of channels from 64 to 512. As an example, the architecture of VGG-13 that uses 13 convolution layers with an activation function ReLU (Nair and Hinton, 2010) is provided in Figure 2. First ten convolution layers map feature of sizes $\{224, 112, 56, 28, 14\}$ with $\{2, 2, 2, 2, 2\}$ layers, respectively. Max-pooling (Scherer *et al.*, 2010) is performed with a window pixel of 2×2 of stride 2 when reducing the size. The stack of these ten layers is followed by three fully connected (FC) layers with $\{4096, 4096, 1000\}$ channels. VGG-13 employs the softmax layer as its final layer.

2.2. ResNet

ResNet (He *et al.*, 2016) opened an era of true ‘deep’ learning by using lots of layers. Models before ResNet have at most around 20 layers, whereas ResNet attempted to make a model using more than 1,000 layers even for a rather small data set. ResNet is a short expression of the residual net. Residual is defined as the difference between the input x and the function $H(x)$ that maps the input to the target value y. ResNet trains the model in a way that minimizes $H(x) - x$ so that the output value of the network is x instead of y. This is reasonable for the problem of image classification because the target y represents a category of an image. The correct classification means the target should be the same category of the input x. $H(x) - x$ is called residual. This shows contrast to the previous methods that operate learning to minimize $H(x) - y$.

As the number of layers increases, the problem of gradient vanishing/exploding becomes worse. To overcome this issue, ResNet uses shortcut (or skip) connection and element-wise addition as illustrated in Figure 3. In the figure, $\mathcal{F}(x)$ is the residual $H(x)-x$, and ‘identity’ indicates identity mapping whose output is the same as the input.

There are two types of connections in the residual blocks of ResNet. Both use identity mapping, but the block’s positions are different. Figure 4 shows the structure of each block. The block in the first position where the size changes needs another convolution layer among mapping, so it is called the ‘convolution block’. The other one is called ‘identity block’. There are five versions of ResNet models utilizing these blocks in (Deng *et al.*, 2009) designed for ImageNet (Deng et al., 2009) data. We explain and illustrate one version, ResNet-34 in Figure 5.

In Figure 5, ‘Conv Block’ and ‘Id Block’ stand for the convolution block and identity block, respectively. The first 7×7 convolution in ResNet-34 makes the ImageNet image into 112×112

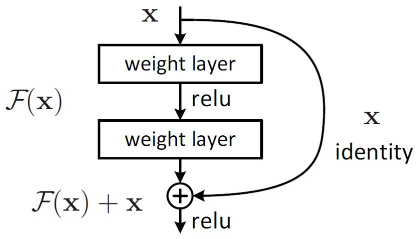
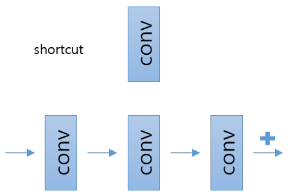
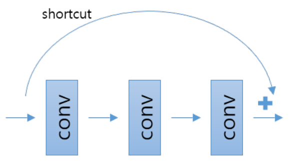

x
weight layer
relu
weight layer
$\mathcal{F}(\mathbf{x})$
x
identity
$\mathcal{F}(\mathbf{x}) + \mathbf{x}$
relu

Figure 3: *A building block skipping one or more layers in the residual learning (He et al., 2016).*



(a) *A convolution block*



(b) *A identity block*

Figure 4: *Two types of blocks in ResNet.*

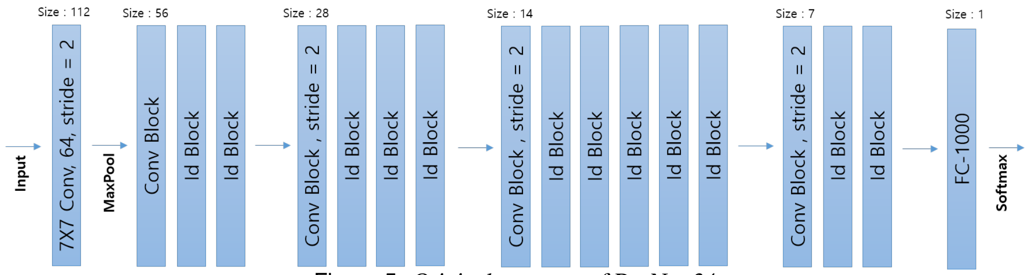


Figure 5: *Original structure of ResNet-34.*

with stride 2. 3×3 max-pooling layer then decreases the image size in half. Next, ResNet blocks are piled up. Each block consists of double-stacked 3×3 convolution layers with $\{64, 128, 256, 512\}$ channels respectively. In each step, the block reduces image size, not using a pooling layer but using the first block as a convolution block. Finally, the image passes through the average-pooling layer, fully connected layer with 1,000 channels, and softmax layer.

2.3. Wide residual networks (WRN)

Zagoruyko and Komodakis (2016) proposed wide residual networks (WRN) based on the structure of ResNet (He *et al.*, 2016). ResNet opened the era of true deep learning, showing better performance as the number of layers increases. However, there are too many layers compared to the improvement, which slow down the learning speed. To resolve this issue, Zagoruyko and Komodakis (2016) created a ResNet structure that was shortened in length and expanded in width. In ResNet, it increases depth by compressing the width as thin as possible. However, identity mapping does not force the gradient to go through the residual block when it flows through the network. This can cause a problem that only a few blocks learn useful information. In contrast, WRN increases performance and solves the problem by improving the residual block instead of increasing depth. WRN has about 50 times fewer layers

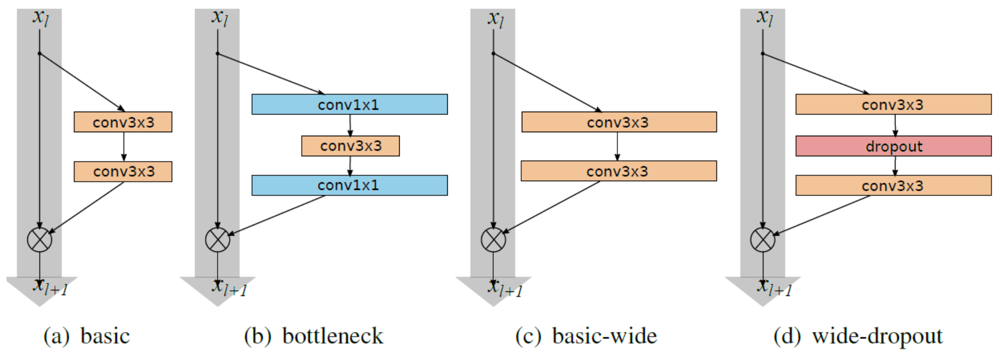


Figure 6: *Wide-dropout residual block (d) compared to basic (a), bottleneck (b), and basic-wide (c). This figure is adopted from Zagoruyko and Komodakis (2016).*

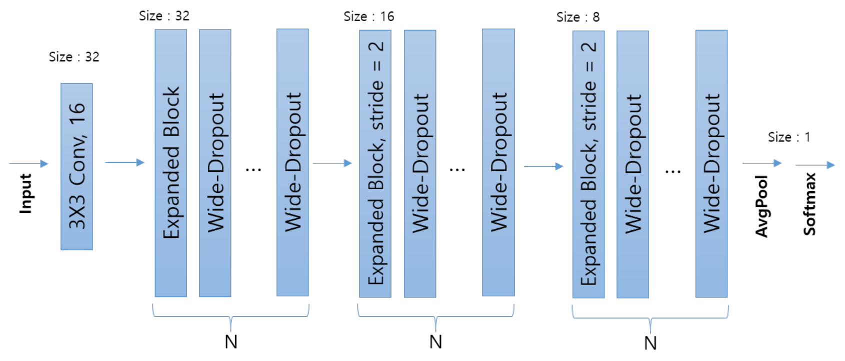


Figure 7: *Structure of WRN-n-k: Expanded convolution block has similar structure to a convolution block in Figure 4. The expanded convolution block has k times wider layer than that of the ResNet convolution block. Wide - dropout block has the same architecture as Figure 6(d) with widening factor k. N in the bottom is calculated as $N = (n-4)/6$.*

and runs about 2 times faster when compared to the ResNet using a similar number of parameters.

Another feature of WRN is the usage of dropout (Srivastava *et al.*, 2014) and batch normalization (Ioffe and Szegedy, 2015). Both have regularization effect of preventing the overfit. WRN inserts dropout between the convolutional layers whose effect have been empirically proved. Moreover, WRN changed the order of convolution, batch normalization (BN), and activation (ReLU) into *BN-ReLU-Conv* from the typical order *Conv-BN-ReLU*. This tends to train faster and achieve higher accuracy. Zagoruyko and Komodakis (2016) introduced six types of convolutions denoted by '$B(M)$'. The original version is $B(3,3)$, which is (d) in Figure 6. The other five type of convolutions are $B(3,1,3)$, $B(1,3,1)$, $B(1,3)$, $B(3,1)$, and $B(3,1,1)$, where the numbers in $B()$ indicate the list of kernel sizes of the convolutional layers in a block. For example, $B(3,1,3)$ has one additional 1×1 layer compared to $B(3,3)$. Using the notation of Zagoruyko and Komodakis (2016), WRN-n-k-$B(M)$ implies WRN model using total n convolution layers, widening factor k, and residual block $B(M)$. For example, WRN-40-2-$B(3,3)$ is the WRN model using $B(3,3)$ residual block of a total 40 layers and two times wider than the original one, which will be applied to real data sets in Section 4. The architecture of WRN-n-k is given in Figure 7.

The initial convolution layer in Figure 7 is not widened yet. Next, it consecutively connects to

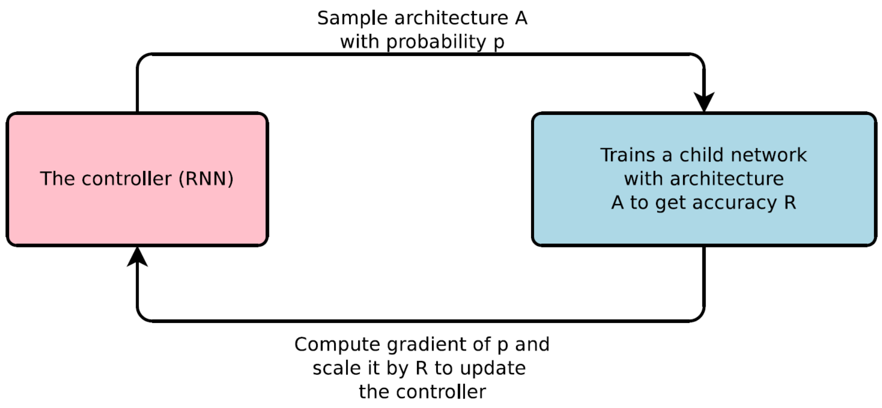


Figure 8: *Overview of neural architecture search (Zoph and Le, 2016). The RNN controller predicts a sample architecture A with probability p. The architecture A trains a child network. It finds the gradient value of p during training, and scale it by R to update the controller.*

three groups of blocks consisting of N blocks whose sizes are {32, 16, 8}. Each block has doubly stacked 3×3 convolution layer of {16, 32, 64} channels with widening factor k. This widening factor creates more channels so that it expands (or widens) the convolution layer. Finally, the average-pooling layer makes the size of output be 1×1.

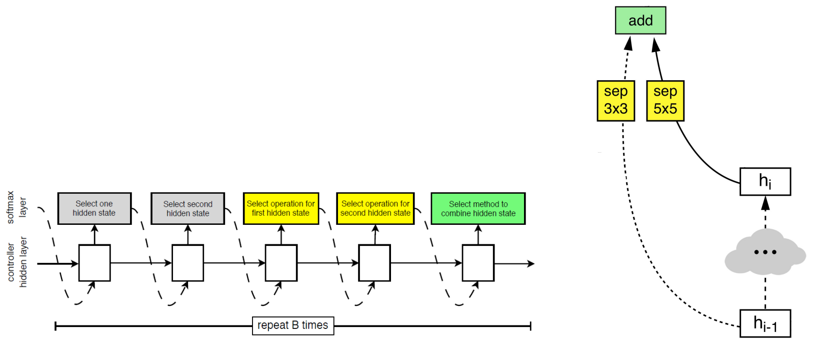
2.4. Neural architecture search (NAS) Net

Researchers manually designed the architectures of the described methods. In contrast, neural architecture search (NAS) automatically plans optimal networks and builds an architecture via machine learning. Zoph and Le (2016) proposed optimal architecture based on reinforcement learning. NAS-Net (Zoph *et al.*, 2018) suggested automatic machine learning (AutoML), which constructs its structure by machine learning. NASNet focuses on the automated process of feature learning, architecture search, and hyperparameter optimization. Figure 8 shows the process of NAS where the recurrent neural network (RNN) controller uses the target data to learn architecture and its performance based on the output.

NASNet consists of a large number of blocks, which is the smallest unit of its structure. The blocks in Figure 9(a) conduct two operations to yield one feature map. Five RNN controllers (white squares) construct one block. The combination operation depends on how to combine two outputs: element-wise or channel-wise. ‘add’ (green box) in Figure 9(b) represents combining two outputs as element-wise. The other type is ‘concatenate’ that channel-wisely combines two outputs.

After Nasnet constructs the blocks, it designs the convolution cells. There are two types of convolution cells; normal cell and reduction cell. The feature maps of the input and output have the same size in a normal cell. In contrast, the reduction cell halves the width and height of the feature map by setting some blocks’ stride as 2.

The process explained in Figure 8 is part of the learning process in the reinforcement learning. In general, reinforcement learning requires repeated heavy learning to improve its performance. When the image size or the number of images is large, such as ImageNet, heavy computation is needed to find the best model. Even with this limitation, NAS is meaningful because it showed the ability to find a model based on reinforcement learning comparable to the one designed by human.


softmax layer
controller hidden layer
Select one hidden state
Select second hidden state
Select operation for first hidden state
Select operation for second hidden state
Select method to combine hidden state
repeat B times
add
sep 3x3
sep 5x5
h_i
h_{i-1}

(a) *Block is formed from the left.* (b) *A formed block.*

Figure 9: *Hidden state inputs are selected from h_{i-1} and h_{i-1}. Various types of identity, convolutions, and pooling layers can be selected as operations. Adding or concatenating can be done in computer operation (Zoph et al., 2018). Two hidden state inputs (grey squares), two operations (yellow squares), and one combination operation are processed in the order in panel (a).*

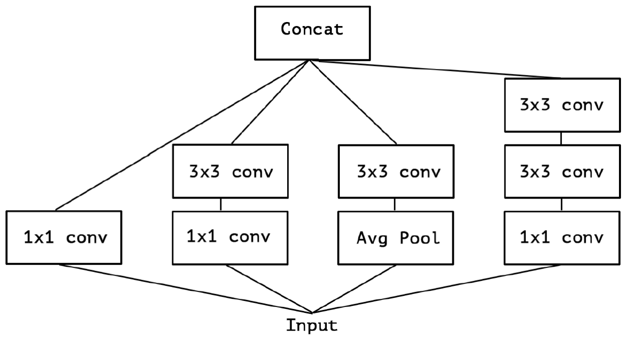


Figure 10: *Inception module described in Chollet (2017).*

NASNet can transfer its fitted architecture using small data to a larger one. Zoph *et al.* (2018) first used CIFAR-10 data (Krizhevsky *et al.*, 2014) to find an optimal architecture and then transferred it to ImageNet data (Deng *et al.*, 2009). The transferred structure using ImageNet is similar to the original one but adding a convolution layer and two reduction cells.

2.5. MobileNet

MobileNet (Howard *et al.*, 2017) reduces computation and model size by adopting the concept of 'depthwise separable convolution' and two hyperparameters called 'width multiplier' and 'resolution multiplier'. As a result, it becomes possible to run a neural network model in a limited environment such as mobile devices. Below are the details of the above terminologies used in MobileNet.

Depthwise separable convolution It was adopted from Xception (Chollet, 2017) that had used an idea of Inception (Szegedy *et al.*, 2016). Depthwise separable convolution is composed of depthwise convolution and pointwise convolution. Inception module is a method of applying the convolution to the feature map in the previous step with multiple kernel sizes, as presented in Figure 10. The Inception module is advangageous because it extracts various feature values using few parameters. Xception extended this idea, and the 'extreme' version of the Inception module allows one 3×3

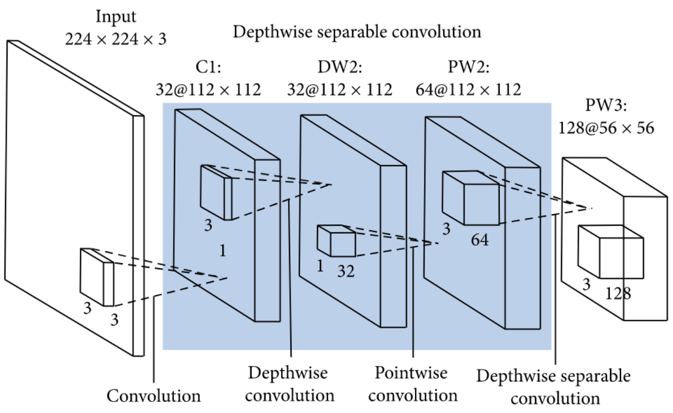


Figure 11: *The depthwise separable convolution adapted from Wei Wang et al. (2020). Depthwise convolution applies a single filter to each input channel, and then* 1×1 *convolution called pointwise convolution is adopted to combine the outputs.*

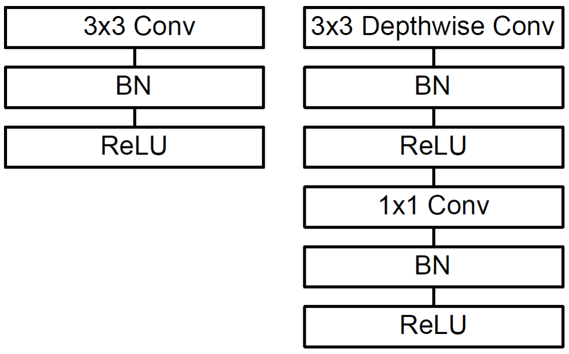


Figure 12: *The structure of depthwise separable convolution (right), compared to standard convolution (left). BatchNormalization (BN) and ReLU layer follow from each convolution layer (Howard et al., 2017).*

convolution per output channel. It is used in MobileNet, as shown in Figure 11. Figure 12 illustrates how the depthwise separable convolution differs from the standard one.

Width multiplier and resolution multiplier Howard *et al.* (2017) suggested two hyperparameters: width multiplier $\alpha \in (0, 1)$ and resolution multiplier $\rho \in (0, 1)$. Both hyperparameters efficiently allow models to tradeoff between latency and accuracy. The width parameter is multiplied by the number of input and output channels. The resolution parameter is multiplied by the feature map size. The number of all parameters is then reduced by α^2 and ρ^2 approximately. The variant of MobileNet is determined by two hyperparameters. For example, '0.75 MobileNet-192' has $\alpha = 0.75$ and $\rho = 0.857$, where the original feature map size(224×224) is reduced to 192×192($192=224 \cdot 0.857$).

The model with $\alpha = 1$ and $\rho = 1$, '1.0 MobileNet-224' is the baseline MobileNet. In addition, reduced MobileNets are the models with $\alpha < 1$ or $\rho < 1$. The architecture can be divided into five parts, as presented in Figure 13. First, there is one convolution layer with stride 2. It halves the input feature map size. Then two groups of blocks that are composed of six and five depthwise separable convolution blocks come next. Finally, two depthwise separable convolution blocks follow. The average pooling layer (AvgPool) makes each feature map size 1×1, and a fully connected layer

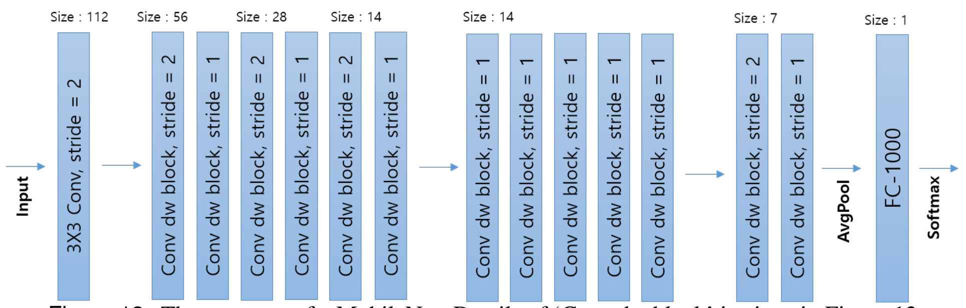

Input
Size : 112
3X3 Conv, stride = 2
Size : 56
Conv dw block, stride = 2
Conv dw block, stride = 1
Size : 28
Conv dw block, stride = 2
Conv dw block, stride = 1
Size : 14
Conv dw block, stride = 2
Conv dw block, stride = 1
Size : 14
Conv dw block, stride = 1
Conv dw block, stride = 1
Conv dw block, stride = 1
Conv dw block, stride = 1
Conv dw block, stride = 1
Size : 7
Conv dw block, stride = 2
Conv dw block, stride = 1
AvgPool
Size : 1
FC-1000
Softmax

Figure 13: *The structure of a MobileNet. Details of 'Conv dw block' is given in Figure 12.*

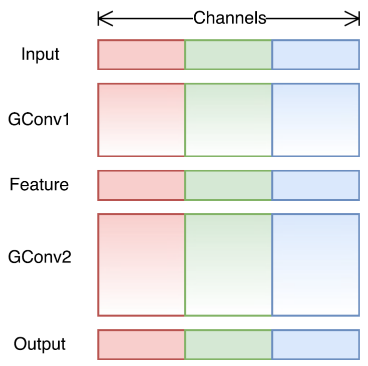


Figure 14: *There are two group convolutions (GConv1 and GConv2). Channels are combined into three groups (red, green and blue) in this network.*

comes in the end.

2.6. ShuffleNet

ShuffleNet (Zhang *et al.*, 2018) aims to create a minimal model by reducing the number of parameters and the computational cost, which is similar to MobileNet in Section 2.5. To build an efficient structure with reasonable performance, ShuffleNet employs depthwise separable convolution as MobileNet does. ShuffleNet hires two more devices; 'group convolution' and 'channel shuffle'.

The concept of group convolution has been introduced in AlexNet. It is motivated by the limited performance of GPU. As a result, the computational cost could be reduced, and the accomplishment became unexpectedly better. Iandola *et al.* (2016) divided the channels into two groups to train in AlexNet. Accordingly, ShuffleNet uses the 'group convolution' to boost its performance and to reduce computational cost. One can use a relatively large number of channels proportional to the amount of reduced computation. It allows the network to have more information. Figure 14 represents a network with group convolutions. When input data enters in the beginning, the channels are combined into groups. Each group conducts convolution separately. However, a problem occurs if there is no communication between the groups. Because the information flows inside each group only, the whole network's representation becomes weak. In another perspective, it is similar to learning individual networks group by group. To prevent this problem, ShuffleNet shuffles the channels within each group. This process relates the input and output channels. Figure 15 illustrate this process. First, the channels of each group are divided into subgroups again in Figure 15 (b) before the channel shuffle in

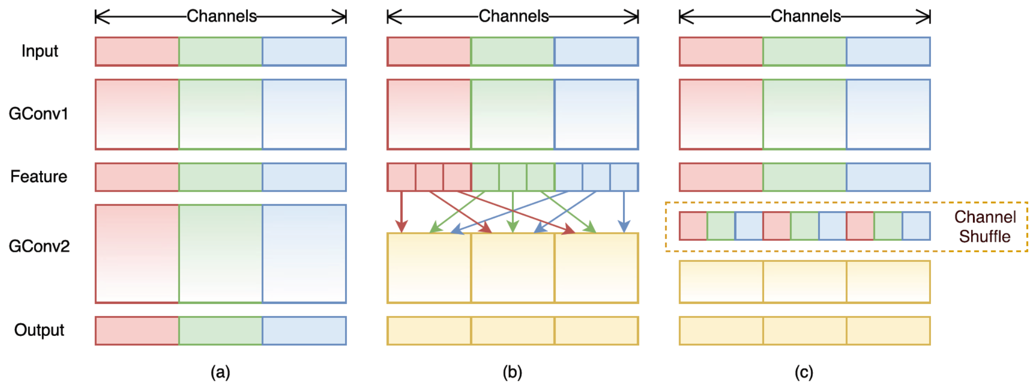


Figure 15: *Channel shuffles: Groups are divided into subgroups (squares in the feature layer of (b)), and conduct shuffling. Shuffle operation is simple; $g \times n$ channels are reshaped into (g, n). Then it becomes the input. Transposing, flattening are followed. The final shuffled channels are given in (c).*

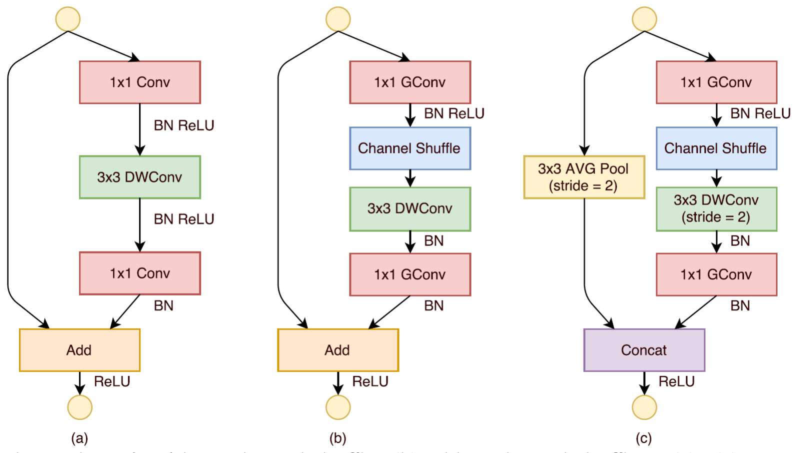


Figure 16: *(a) shows the unit without channel shuffle. (b) adds a channel shuffle to (a). (c) cuts the output size into half through controlling shortcut path using average pooling layer and developing channel dimension using channel concatenating.*

(c). Finally, the channels of each group are mixed, and therefore every group becomes related to all other groups.

Similar to the previously introduced models, the ShuffleNet repeats architecture of the ShuffleNet unit. Figure16 shows the structure of the ShuffleNet unit. Assuming the input size $c \times h \times w$ and the number of bottleneck channel m, the computational cost of ShuffleNet is $hw(2cm/g + gm)$. Comparing this with the cost of ResNet $hw(2cm + 9m^2)$, we can see the computational cost can be significantly reduced in general.

The architecture of ShuffleNet in Figure 17 consists of four stages. In the beginning, there are the convolution layer and max-pooling layer with 3×3 kernel size and 2 strides. Each layer halves image size so the feature map becomes 56×56 from 224×224. In stage 2, the first shuffle unit with 2 strides halves the feature map size, and three shuffle units follow. Similarly, the first shuffle unit with 2 strides halves the feature map size in stages 3 and 4. There are seven and three shuffle units after

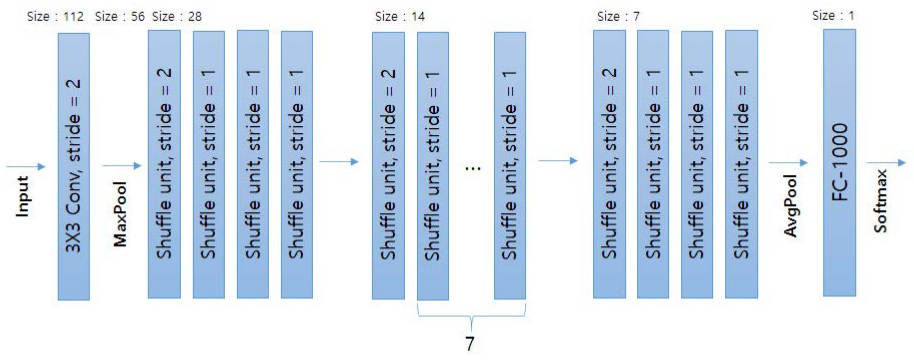


Figure 17: *There are four stages in the structure of ShuffleNet. Stage 1 has one shuffle unit. There are 4, 8, and 4 shuffle units in stage 2, 3, and 4, respectively.* 7×7 *kernel in the average pooling (AvgPool) converts the feature map into* 1×1*. FC-1000 represents a fully connected layer with 1000 channels.*

the first unit, respectively. The final global average pooling with 7×7 kernel converts the feature map into 1×1. The fully connected layer flattens the whole channels.

3. Transformed framework for comparison

The six models reviewed in Section 2 are modified to become relatively similar in size. Below are the details of each model.

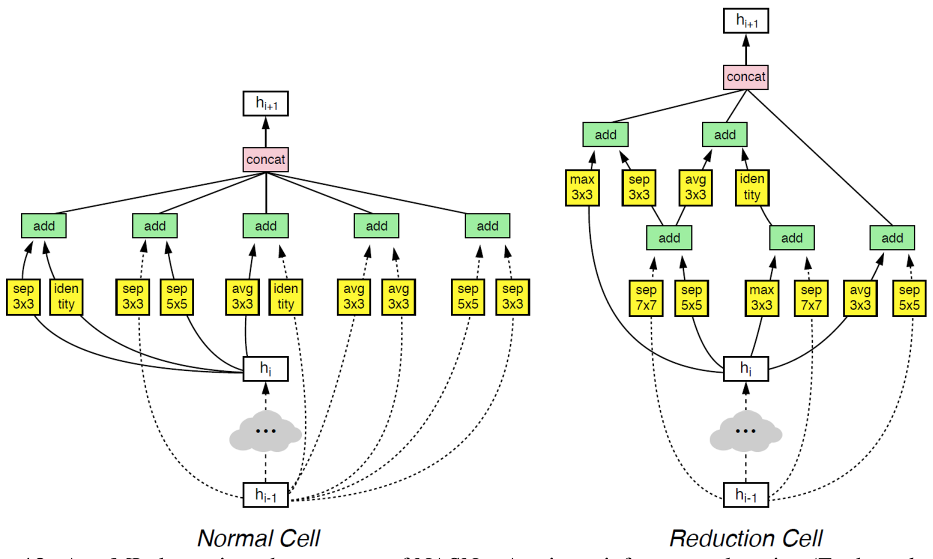


Figure 18: *AutoML determines the structure of NASNet-A using reinforcement learning (Zoph et al., 2018).*

1. **VGGNet:** We transform the original structure of VGG-13. There are three blocks, and each block reduces the output size in half. Each block uses two vgg layers using {64, 128, 256} number of filters, respectively. Input image then flows into six convolution layers and two fully connected

layers. Thus, the model is eight layers deep. The model is sketched in the first row (from the top) of Figure 19.

2. **ResNet:** The second row in Figure 19 shows ResNet that stacks 18 layers of 3×3 convolutions on the feature maps of sizes $\{32, 16, 8\}$, six layers for each feature map size. The 3×3 convolution layer is at the top, and the global average pooling layer is at the bottom of the stack of layers.

3. **WRN:** WRN using 40 layers with widening factor 2 is denoted by 'WRN-40-2'. Section 2.3 describes the specific structures of WRN. The structure of 'WRN-40-2' is shown in the third row of Figure 19.

4. **NASNet:** We use the original structure of NASNet-A without modification. We provide its structure in Figure 18.

5. **MobileNet:** The values of width parameter α and resolution parameter ρ are both set to one. We remove two depthwise convolution blocks from the model and fix the size of the feature map for input as 32×32. The fourth row in Figure 19 shows the structure of MobileNet.

6. **ShuffleNet:** The final row in Figure 19 shows the modified structure of the ShuffleNet. We use the max-pooling layer with stride 1 and remove stage 2 from the original model introduced in Section 2.6.

4. Real data applications

We use four data sets, which details are given below.

1. **CIFAR-10:** CIFAR-10 (Krizhevsky *et al.*, 2014) data set contains 60,000 number of 32×32 pixel color images of 10 different classes. The 10 classes represent airplanes, cars, birds, cats, deer, dogs, frogs, horses, ships, and trucks. There are 6,000 images of each class.

2. **CIFAR-100:** Similar to CIFAR-10, CIFAR-100 (Netzer *et al.*, 2011) contains 60,000 32×32 pixel color images of 100 different classes. Each class contains 600 images. The 100 classes are grouped into 20 categories; aquatic mammals, fish, flowers, food containers, fruit and vegetables, household electrical devices, household furniture, insects, large carnivores, large man-made outdoor things, large natural outdoor scenes, large omnivores and herbivores, medium-sized mammals, non-insect invertebrates, people, reptiles, small mammals, trees, vehicles 1, and vehicles 2.

3. **Fasion-MNIST:** Fashion-MNIST (Xiao *et al.*, 2017) data consists of 28×28 pixel gray-scale images of 10 classes. This data set is known as a representative of low pixel size and gray-scale image. It contains 70,000 images among which 60,000 are allocated for training data and the others for test data. The labels of images are assigned to one of the following; T-shirt/top, trouser, pullover, dress, coat, sandal, shirt, sneaker, bag, and ankle boot.

4. **SVHN:** SVHN (Netzer *et al.*, 2011) is a photographed data of house numbers collected by Google Street view. There are 73257 images for training and 26032 images for testing. There are two formats of SVHN: original images of full numbers with bounding boxes for each digit and cropped photos into 32×32 size. We use the latter format, which is widely used. There are 10 classes (or digits) from 0 to 9.

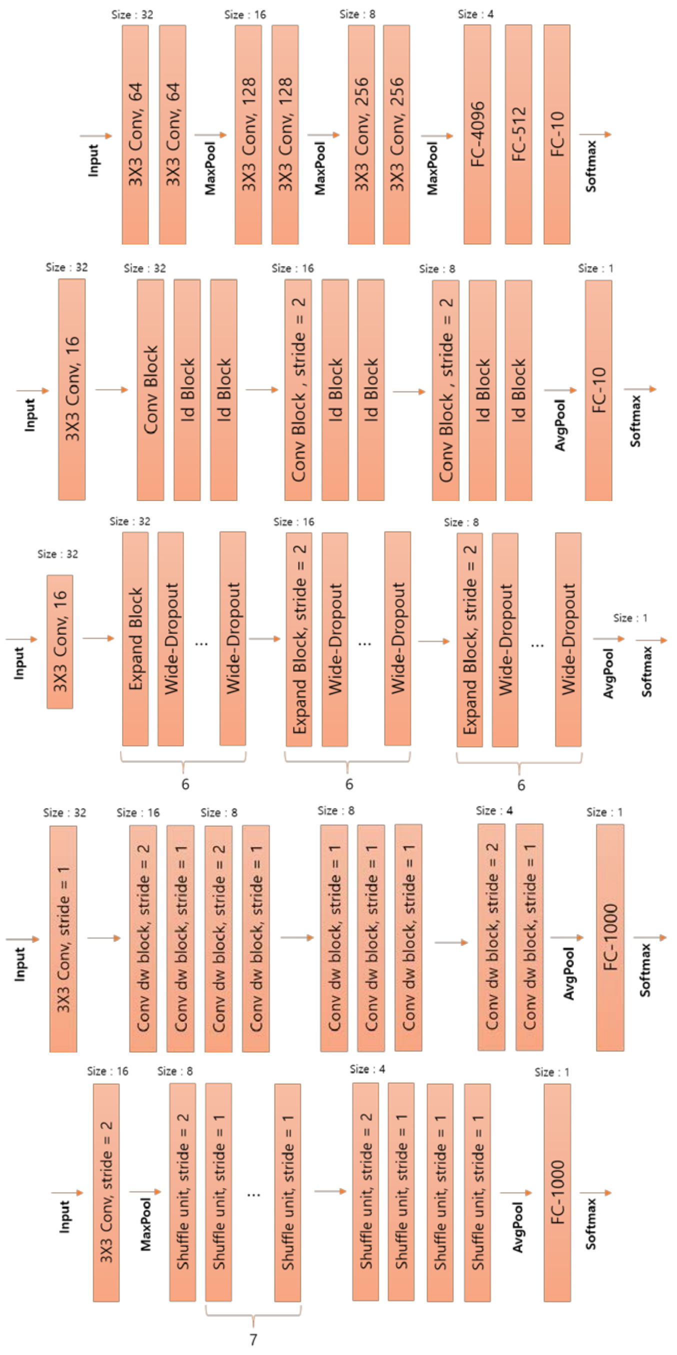


Figure 19: *The structure of the transformed models for VGGNet, ResNet, WRN, MobileNet, and ShuffleNet from top to bottom.*

Table 1: The mean prediction accuracy of six models averaged over three experiments. The accuracy for 'top5' considers top-ranked five candidates for predicting the target class. The first five rows show the results from our transformed framework, whereas the other six rows obtained from the original version of the models

Model	CIFAR-10	CIFAR-100	CIFAR-100 (top5)	Fasion-MNIST	SVHN
ResNet	87.70	59.51	86.39	**94.27**	**95.76**
MobileNet	**90.08**	**64.64**	**88.42**	93.95	95.73
ShuffleNet	84.70	41.86	71.83	88.82	94.56
VGGNet	86.51	57.56	84.77	91.59	95.28
WRN	85.24	46.61	73.32	93.27	95.52
NASNet (orgn)	**92.44**	**64.47**	**86.39**	**94.39**	**95.96**
ResNet (orgn)	88.76	55.03	82.62	93.55	95.97
MobileNet (orgn)	89.51	60.35	86.00	93.66	95.81
ShuffleNet (orgn)	83.83	42.37	71.28	88.62	94.83
VGGNet (orgn)	87.11	56.25	84.01	91.62	95.03
WRN (orgn)	81.52	45.22	76.13	93.34	94.69

An optimizer RMSProp (Mukkamala and Hein, 2017) is used. We use Google colaboratory's GPU, Tensorflow, and Keras for the deep learning framework in the data applications. Specific versions of them are as follows,

- Python: 3.6.9
- Tensorflow: 2.3.0
- Keras: 2.4.0
- CPU: Intel Xeon 2.30GHz
- GPU: Tesla P100-PCIE-16GB
- RAM: 12GB

We split the data into three parts; training, validation, and test data. The prediction accuracy is calculated over the test data. Because all values in the data sets are categorical, we calculate the classification accuracy to gauge the performance. In addition, we use the top-ranked five categories for the CIFAR-100 to measure the prediction accuracy. That is, our prediction is correct when the top five (predicted) classes include the target class. It is because there are too many classes. Table 1 presents the average prediction accuracy over three repetitions. The number of repetitions is small due to the restriction on computing time, but the standard errors of the mean prediction accuracy are all smaller than 0.005 except for the CIFAR-100 using WRN. The standard error for the exceptional case is 0.044. So, we regard the results do not have large variation. The top three models in Table 1 contains about one million parameters, and the next three models have more than two million. Six models in the bottom of the table are the original version of the models. Because we do not transform the NASNet, the results from the original NASNet are given only in the sixth row in Table 1.

ResNet shows good performance on relatively simple, grayscale, or numeric images. MobileNet performs well considering the computing time for relatively complex images with specific objects and large classes, such as CIFAR-10 and CIFAR-100. NASNet performs best among the large models regardless of the nature of the data. The results from the original models are comparable to the transformed models. There is no clear superiority between the transformed and original models.

Finally we provide the computing time for running each model once in Table 2.

Table 2: Computing time (in minutes) for running the model once under the described computing facilities

Model	CIFAR-10	CIFAR-100	Fasion-MNIST	SVHN
ResNet	53	72	52	93
MobileNet	45	36	40	68
ShuffleNet	47	29	24	86
VGGNet	135	101	119	193
WRN	120	146	148	206
NASNet (orgn)	450	500	551	499

5. Discussions

In the preceding sections, we have introduced six CNN models developed during ILSVRC. We then applied them to four popular real image data sets. We have introduced the architectural features of each model and modified the models to a similar size. The results of the prediction accuracy demonstrate that NASNet is in fact recommended for various types of image data. When the GPU specification is low, and fast learning is required, such as processing data in real-time, choosing a light model such as MobileNet or ResNet seems a reasonable choice. The results in this paper may not be generalized because there are numerous variants.

Acknowledgement

Yoonsuh Jung's work was partially supported by National Research Foundation of Korea (NRF) grant funded by Korea government (MIST)(2019R1A4A1028134 and 2021R1F1A1062347).

References

Chollet F (2017). Xception: Deep learning with depthwise separable convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1251–1258.

Deng J, Dong W, Socher R, Li LJ, Li K, and Fei-Fei L (2009). ImageNet: A large-scale hierarchical image database, *2009 IEEE Conference on Computer Vision and Pattern Recognition*, Miami, FL, USA.

He K, Zhang X, Ren S, and Sun J (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 770–778.

Howard AG, Zhu M, Chen B, Kalenichenko D, Wang W, Weyand T, Andreetto M, and Adam H (2017). *Mobilenets: Efficient Convolutional Neural Networks for Mobile Vision Applications*, arXiv preprint arXiv:1704.04861

Iandola FN, Han S, Moskewicz MW, Ashraf K, Dally WJ, and Keutzer K (2016). *SqueezeNet: AlexNet-level Accuracy with 50x Fewer Parameters and < 0.5 MB Model Size*, arXiv preprint arXiv:1602.07360

Ioffe S and Szegedy C (2015). Batch normalization: accelerating deep network training by reducing internal covariate shift. In *JMLR Workshop and Conference Proceedings*, **37**, 448–456.

Krizhevsky A, Nair V, and Hinton G (2014). The cifar-10 dataset, http://www.cs.toronto. edu/kriz/cifar

LeCun Y, Boser B, Denker JS, Henderson D, Howard RE, Hubbard W, and Jackel LD (1989). Backpropagation applied to handwritten zip code recognition, *Neural computation*, **1**, 541–551.

Nair V and Hinton GE (2010). Rectified linear units improve restricted boltzmann machines. In *ICML'10: Proceedings of the 27th International Conference on International Conference on Machine Learning*.

Netzer Y, Wang T, Coates A, Bissacco A, Wu B, and Ng AY (2011). Reading digits in natural images with unsupervised feature learning. In *Advances in Neural Information Processing Systems (NIPS)*.

Mukkamala MC and Hein M (2017). Variants of RMSP rop and a dagrad with logarithmic regret bounds. In *Proceedings of the 34th International Conference on Machine Learning*, **70**, 2545–2553.

Scherer D, Müller A, and Behnke S (2010). Evaluation of pooling operations in convolutional architectures for object recognition, *International Conference on Artificial Neural Networks*, 92–101.

Simonyan K and Zisserman A (2014). Very deep convolutional networks for large-scale image recognition, *Computer Vision and Pattern Recognition*, arXiv:1409.1556

Srivastava N, Hinton G, Krizhevsky A, Sutskever I, and Salakhutdinov R (2014). Dropout: a simple way to prevent neural networks from overfitting, *The Journal of Machine Learning Research*, **15**, 1929–1958.

Szegedy C, Liu W, Jia Y, *et al.* (2015). Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1–9.

Szegedy C, Vanhoucke V, Ioffe S, Shlens J, and Wojna Z (2016). Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2818–2826.

Wei W, Yiyang H, Ting Z, Hongmei L, Jin W, and Xin W (2020). A new image classification approach via improved mobilenet models with local receptive field expansion in shallow layers, *Computational Intelligence and Neuroscience*.

Xiao H, Rasul K, and Vollgraf R (2017). Fashion-MNIST: a novel image dataset for benchmarking machine learning algorithms, arXiv:1708.07747

Zagoruyko S and Komodakis N (2016). Wide residual networks. In *Proceedings of the British Machine Vision Conference (BMVC)*, **87**, 12.

Zhang X, Zhou X, Lin M, and Sun J (2018). Shufflenet: An extremely efficient convolutional neural network for mobile devices. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 6848–6856.

Zoph B and Le VQ (2016). Neural architecture search with reinforcement learning, In *CoRR*.

Zoph B, Vasudevan V, Shlens J, and Le QV (2018). Learning transferable architectures for scalable image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 8697–8710.

Received August 11, 2021; Revised December 07, 2021; Accepted December 08, 2021