IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received April 27, 2018, accepted June 6, 2018, date of publication June 18, 2018, date of current version July 12, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2848307

WPNets and PWNets: From the Perspective of Channel Fusion

DAOJUN LIANG[1], FENG YANG[1], TIAN ZHANG[1], JIE TIAN[1], AND PETER YANG[2]
[1]School of Information Science and Engineering, Shandong Normal University, Jinan 250014, China
[2]Amazon, Seattle, WA 98101, USA

Corresponding author: Feng Yang (yangfeng@sdnu.edu.cn)

This work was supported in part by the Natural Science Foundation of Shandong, China, under Grant BS2014DX006 and in part by the Taishan Scholar Project of Shandong, China.

ABSTRACT The performance and parameters of neural networks have a positive correlation, and there are a lot of parameter redundancies in the existing neural network architectures. By exploring the channels relationship of the whole and part of the neural network, the architectures of the convolution network with the tradeoff between the parameters and the performance are obtained. Two network architectures are implemented by dividing the convolution kernels of one layer into multiple groups, thus ensuring that the network has more connections and fewer parameters. In these two network architectures, the information of one network flows from the whole to the part, which is called whole-to-part connected networks (WPNets), and the information of the other network flows from the part to the whole, which is called part-to-whole connected networks (PWNets). WPNets use the whole channel information to enhance partial channel information, and the PWNets use partial channel information to generate or enhance the whole channel information. We evaluate the proposed architectures on three competitive object recognition benchmark tasks (CIFAR-10, CIFAR-100, SVHN, and ImageNet), and our models obtain comparable results even with far fewer parameters compared to many state of the arts. Our network architecture code is available at github.

INDEX TERMS Machine learning, computer vision, image processing.

I. INTRODUCTION

Convolutional neural networks have made great progress in many fields, and the research of the network architecture has never stopped. There are a lot of networks that have achieved very good performance by applying new architectures. AlexNet [1] is the first to demonstrate the generalization ability of convolutional neural networks on large data. VGGNets [2] show that better performance can be achieved with smaller convolutional kernels and deeper layers. GoogLeNets [3] use different convolution kernels to establish more connections and more diverse representations between adjacent layers. ResNets [4] and Highway Networks [5] add the front layer information to the back layer through the bypass structure, which is more conducive to the backpropagation of the gradient, thus further deepening the depth of the network. ResNeXts [6] combine group convolution into ResNets [4], which perform split-transform-merge operations on features to improve network performance while reducing parameters. DenseNets [7] pass the features of each preceding layer to all of its subsequent layers to alleviate the vanishing/exploding gradient problem [8], [9] and to facilitate information fusion between layers.

We conclude that the following principles need to be considered in neural network designs:

- Ensure the gradient efficient backpropagation to avoid gradient vanishing problem.
- Use different layers of information fusion to learn the diversity of representation.
- Reduce the parameters as much as possible so that the network can train and inference more quickly.

Which feature fusion method is the most effective? This is an open question, but there is a rough direction that uses more primitive features and makes more connections between layers as well as guarantee the efficiency of stochastic gradient descent (SGD) with backpropagation [10]. In this paper, we fully explore the relationship between the whole and the part of the neural network's channels, include whole-to-part and part-to-whole connection relation. In WPNets, the feature channel is first divided into several groups, and then the overall features are sequentially added to each group

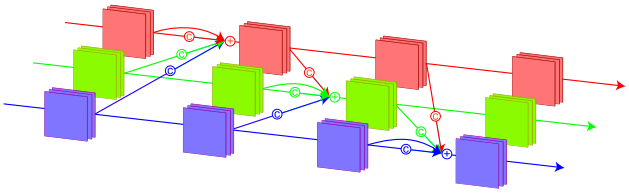


FIGURE 1. The symbol *C* in the figure represents a convolution operation, the "+" represents an addition operation, and the straight line with an arrow represents the flow of information. In WPNets block, the original channel is divided into 3 groups, each time compressing all the channels information to a group until all the groups are changed.

by convolution. Fig. 1 illustrates this layout schematically. In PWNets, the part of channels is convoluted and then add them to the entire channels. Fig. 2 depicts this network architecture.

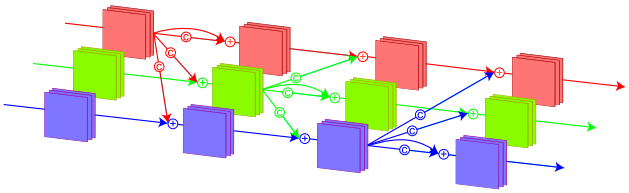


FIGURE 2. The various symbols in the figure are the same as those in FIG. 1. In PWNets block, the original channel is divided into 3 groups, the channels information of the 3 groups is connected to all groups in turn.

A network designed with the whole and partial relationships of the channel has many advantages over other networks. The group structure in WPNets are similar to "Inception Module" in GoogLeNets [3], and the bypass structure in WPNets are like that in ResNets [4], which transfers information to the back layer. The channel information for WPNets will flows all subsequent layers, which is similar to that in DenseNets [7].

Contrary to WPNets, the PWNets first amplifies the information of one group by convolution, and then fuse it with all channel information. As shown in Fig. 2, the number of channels in each group is magnified to the sum channels in all groups, and the amplified channels are then added to the channels of all groups. The partial channels information is "amplified" to the whole channels information, which increases the nonlinear computation and representation ability of the network.

From the perspective of group convolution, the number of intra-block connections for the two networks is $G \times G$, G is the number of groups. Half of the connections are between groups, and the other half of connections are between layers. There are so many connections that ensure the full fusion of the information in each group and the addition or concatenation operations are also present on each layer, so the problem of gradient vanishing is therefore alleviated. Because a single layer can be expanded into a block structure based on overall and local relationships, the network can be designed to be deeper while ensuring no additional parameters are added. For instance, we can split a layer with C channels into a block with a group number in $\{1, \cdots, C\}$, but the parameters of the block remain unchanged as the depth of the block increases.

The main contributions of this paper include:

- By analyzing the relationship between channels, two new network structures are proposed, one with information flowing from whole to part (WPNets), and the other one with information flowing from part to whole (PWNets).
- With the same network depth, WPNets and PWNets have more inter-group connections and inter-layer connections, which have fewer parameters and better performance than other network architectures.
- The traditional network layer can be replaced with the block of WPNets or PWNets without adding parameters, so as to achieve better performance.

The article is organized as follows: First, Section II reviews related works. Then two kinds of network blocks proposed in Section III. After that in Section IV we'll detail the network architecture. Next, we present our experimental results in Section V. Finally, we compare the various network architectures with ours in Section VI and conclude this paper in Section VII.

II. RELATED WORK

There are many networks with multi-branch structures [11]–[14]. GoogLeNets [3], [15] use an "Inception module" structure to convolve the features of the previous layer using different filter sizes, and then combines these features as input to the next layer. FractalNets [16] contain interactive subpaths of different lengths, but does not contain any straight-through or residual connections. This structural layout is a truncated fractal that allows more connections and representations between layers of the network. Wide residual networks (WPN) [17] try to prove that the depth is not the only factor of the ResNet [4] to achieve competitive performance; they mainly achieve good performance by extending the width. The Deeply-Fused Nets [18] combine the intermediate representations of base networks, where the fused output serves as the input of the remaining part of each base network, and perform such combinations deeply over several intermediate representations. The two networks we proposed have similarities with these networks, but the difference is that each layer of our network has channels information fusion.

ResNets [4] and Highway Networks [5] allow low-level layers information to flow to high-level layers through the bypass structure; this alleviates the vanishing-gradient that permitted to design hundreds of layers network. Unlike them, the WPNets add all channels information to the partial channels, and the PWNets add part of the channel information to all of its channels. The addition operations in the two networks are similar to the identity mapping in ResNets [4], which is used to transfer the previous layer information to the sequential layer, so the WPNets and PWNets can be designed very deep.

Group convolution was first proposed by AlexNet [1]; it was for distributing the model over two GPUs. ResNeXt [6] was exploiting grouped convolutions to improve the accuracy of ResNets. A special case of the grouped convolutions is the channel-wise convolution in which the number of groups is equal to the number of channels. Channel-wise convolution are used extensively to reduce parameters or improve performance in Xception [19] and MobileNets [20]. Similarly, our network structure also divides the original information into groups, but we convolute one group per layer instead of convoluting them all in one layer.

Instead of adding identity mapping to the next layer, DenseNets [7] concatenate the features of the current layer into the subsequent layers, it allowing the high layer to use all of the previous layer features, which alleviates the vanishing-gradient and implements the implicit deep supervision. There are many similarities between our networks and DenseNets [7]: the information flows to all its sequential layers, so they can use all previous layer information to produce the new layer. The main difference is that we only use all the previous information with the residual structure and the other groups with the original information to get the residual structure of the new layer. From the point of view of group convolution, the connections between layers in the block of WPNets are same as the DenseNets [7], the number is up to $\frac{G \cdot G}{2}$, but the WPNets has more $\frac{G \cdot G}{2}$ connections between groups.

III. WHOLE AND PART CONNECTED NETWORKS

Block structure is widely used in information fusion networks. The network structure usually has the same number of feature size. We regard the block structure as an extended version of the layer structure, so that in the process of information fusion, some layer can filter or enhance the information, then change the learning and representation of the feature. We first divide the networks into many blocks, and we consider the whole and partial relationships within the block channel.

We denote the features of the l^{th} layer as x_l, specially, the input layer of the block as x_0. We divide the channels into G groups, and denote the $j^{th}(1 \leq j \leq G)$ group's features of layer l as g_l^j, the number of channels of this group recorded as C_l^j. We define f as a composite function of multiple convolutional layers, and one layer includes many consecutive operations, such as convolution, batch normalization (BN) [21], and rectified linear unit (ReLU) [22].

A. WPNets

The block structure of WPNets can be defined as:

$$g_l^j = g_l^j + f(x_{l-1}) \tag{1}$$

$$x_l = [g_l^1, \cdots, g_l^j, \cdots, g_l^G] \tag{2}$$

Where $[g_l^1, \ldots, g_l^j, \ldots, g_l^G]$ refers to the concatenation of all groups' feature-maps in layer l. The maximum value of j in the equation is equal to the maximum value of l because the number of groups are equal to the depth of the layer in the block structure.

The f in eq. (1) is a composite function that "compresses" the features information in the $l-1$ layer to the j^{th} group in the l^{th} layer, so we call f a compression function. It should be noted that if the number of channels output by f is not equal to the number of channels in g_l^j. 0 will padded in the insufficient channel to complete the addition operation in eq. (1). After compression process, the information of the group will be changed. Eq. (2) shows the changed group will be concatenated with other unchanged groups as the inputs to next layer. We did it in an iterative way until the features of all groups have changed.

We can find that the network structure defined by eq. (1) is very similar to ResNets [4]: they both add low-level layers information to the high-level layers through the bypass structure. But the main difference is that WPNets add the information of the front layer to the part of channels of the back layer, rather than adding that to all channels like ResNets [4] does. Such small differences between the two networks will result in different ways of learning and representation information. WPNets emphasize the relationship between the part and the whole, and it forces each group to learn different information representations from other groups, thus allowing the network to have the diversity of representations. This conclusion can be obtained from the experiments we have done in section V: the performance of the network will be improved as the number of groups increases.

If we remove the bypass structure of eq. (1), we will get the following form:

$$g_l^j = f(x_{l-1}) \tag{3}$$

Using eq. (3), a new group is obtained at each layer to use each group of information more evenly. We find that this form can achieve very good performance but is a little worse than the structure which uses eq. (1).

Eq. (2) is an important structure adopted by DenseNets, which take all the front layers as input to produces the new layer. If we think of WPNets groups as layers in DenseNets, they will have a lot of similarities. They are both densely connected networks, and they both use information from all previous layers to produce or enhance the information of the back layer. If we change the eq. (3) and eq. (2) into the following form, we will get the DenseNets [7] architecture:

$$g_l^{G+1} = f(x_{l-1}) \tag{4}$$

$$x_l = [g_l^1, \cdots, g_l^j, \cdots, g_l^G, g_l^{G+1}] \tag{5}$$

Eq. (4) produce the new layer g_l^{G+1} and eq. (5) concatenates the layer with all existing layers in the block. It looks like the two structures are very similar, but in fact, there are essential differences: the DenseNets [7] do not change the information of the input features in the block, it just uses the input features to produce new information and concatenate it. In the shallow network, which is very effective, since the proportion of the number of channels of the newly generated

layer to the number of channels of the original layer is appropriate. But if the network becomes deeper, the proportion will become small due to the number of origin channels becoming larger, this will result in a very limited learning ability in the back layer. If we increase the number of new generation channels, this will lead to a dramatic increase in the number of network parameters, then making the network difficult to get deeper. This shows that the growth rate of the number of channels in DenseNets [7] and its depth are contradicting each other. One solution is to compress the channels through the transition structure. Contrary to DenseNets [7], WPNets only change the representation of the internal features in its block.

The block structure of the WPNets can increase the number of channels by replacing the origin group with a new produced group. We can use eq. (3) and eq. (2) to implement it. But if we use eq (1) and eq. (2) to gradually increase the number of channels of the block in WPNets, we should project the origin group's channels to the new groups by padding the inadequate channels with 0. Both methods are commonly used to implement WPNets.

B. PWNets

The block structure of PWNets can be defined as:

$$x_l = x_{l-1} + f(g_{l-1}^{j-1}) \tag{6}$$

$$g_l^j = x_l / [g_l^1, \cdots, g_l^{j-1}, g_l^{j+1}, \cdots, g_l^G] \tag{7}$$

where the " / " in eq. (7) represents the operation of obtaining differential sets, and eq. (7) representing the channels of group g_l^j is obtained by removing channels of other groups from the channel sets of x_l. $[g_l^1, \cdots, g_l^{j-1}, g_l^{j+1}, \cdots, g_l^G]$ representing the concatenation of all group features-maps except the g_l^j group.

Contrary to WPNets, eq. (6) shows the layer in the block of PWNets take the part of channels as inputs then addition its outputs into the whole channels. The composite function f is, to some extent, equivalent to a channel information amplifier, so we call f the amplification function. If we want to increase the number of channels at each layer, we will pad 0 in the area where the number of channels is insufficient. Eq. (7) shows that we will get the other group which added information from the front group as the inputs for next layer. This process will be done iteratively until all the groups had exchanged its information with the whole channels.

The channels of each layer in the block of PWNets will be magnified G times. From this point of view, PWNets are a ensemble of SqueezeNets [23]. SqueezeNets [23] combine the filter sizes of 1×1 and 3×3 into one convolution process and increase the number of channels in each layer of the nets, aiming to decrease the nets parameters as well as to keep the net's performance. Each group of PWNets is equivalent to the bottleneck block of SqueezeNets [23] and the output features will be preserved and flow to its sequence layer.

C. RELATIONSHIP WITH TRADITIONAL NETWORK

Let's analyze the two kinds of network structures from convolutional kernel's perspective. We can record the traditional convolutional progress as:

$$Y_l = f(W_l \star X_{l-1}) \tag{8}$$

where X, Y, W are all matrices and respectively represent the inputs, outputs and the parameters of the convolutional kernel, f is a nonlinear function, and the asterisk $\star$ is the convolution operation. We can divide the corresponding matrix into G blocks, which is equivalent to dividing features into G groups. The matrix W can be grouped by column or grouped by row, and we use the same superscript j to represent the grouping of the matrix W. If W is grouped by columns, the number of convolution kernels in each group is equal to the number of input channels. This grouping method corresponds to WPNets. If W is grouped by rows, the number of convolution kernels in each group is equal to the number of output channels. This grouping method corresponds to PWNets. The matrix W can be grouped by columns as:

$$Y_l^j = f(W_l^j \star X_{l-1}) \tag{9}$$

where the superscript $j (1 \leq j \leq G)$ is the index of matrix blocks. If we add some limited conditions to eq. (9), we can obtain the WPNets form with the follow conditions equation:

$$\begin{aligned} Y_l^j &= f(W_l^j \star X_{l-1}) + X_{l-1}^j \\ X_l &= \begin{pmatrix} Y_l^j \\ 0 \end{pmatrix} + X_{l-1} \end{aligned} \tag{10}$$

In eq. (10), we have applied the nonlinear function f many times. Because f has no trainable parameters, this structure does not add parameters but improves the network representation ability. Similarly, We can add another limited condition on eq. (9) to get the PWNets form:

$$X_l = f(W_l^j \star X_{l-1}^j) + X_{l-1} \tag{11}$$

where the matrix W is grouped by rows. The limited conditions in eq. (10) of WPNets and eq. (11) of PWNets are an inverse process, but they can both achieve a good performance compare to other networks. This inverse process has the same nature: First, the convolution kernel is divided into G groups, that is, the convolution kernel matrix is divided into many blocks, and then add some restrictions to these block matrices. These restrictions contain nonlinear function f, and f is applied to the block matrix to make it have more nonconvex representations. Because the activation function does not have trainable parameters, we do not introduce additional hyperparameters. No matter what the method of information fusion is, they actually achieve the same effect: making the overall channels information more diverse, allowing each channel to be more discriminative.

Each layer of WPNets use the whole channel information to update the local channel information, which conforms to the nature of the convolutional neural network (CNN):

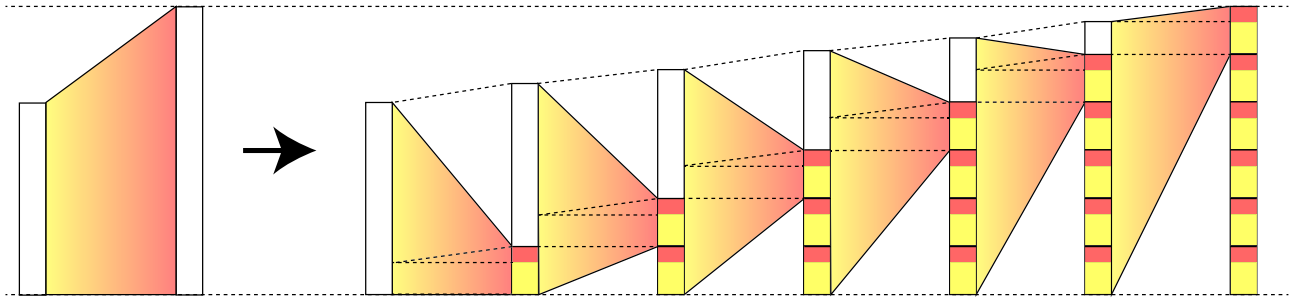


FIGURE 3. The rectangle in this figure represents the input or output features of the layer, and the quadrilateral with transition colors represents the convolution process. The left side of the figure represents the traditional single convolutional layer, whose input and output channels are C_1 and C_2 respectively. And the right shows the process of unfolding a single convolution layer into 6 layers, and the number of channels in each layer increases gradually.

the layer in CNN uses many groups of convolution kernels, and each group of convolution kernels requires learning to represent differently from other groups. Each convolution kernel is convolution with all the channels in the front layer, and then a summing operation is performed to produce a new posterior layer channel. The WPNets seem to be amplifying this process between layers: the channels are divided into many groups corresponding to multiple groups of convolution kernels, and then the whole block performs a depth fusion process between the groups, equivalent to the convolution process of convolution kernel. The information of the sliding window in the convolution kernel is different when sliding. In WPNets, however, the structure of the traditional layer is designed to be deeper, so that each layer will have a set of information that is updated. This allows the input information to be slightly different in each group's convolution process, so the whole block is equivalent to a large sliding convolution process.

WPNets are like a convolution kernel sliding process, gradually changing each part of the channel. PWNets are like the process of adding all channels to generate new back channels after the sliding of the convolution kernel, which delays the addition process of multiple channels to generate new channels, and gradually generate channels of the new layer. So the block structure of PWNets will be more powerful to represent information than the traditional convolutional layer.

D. THE PARAMETER OF THE BLOCK STRUCTURE

We assume that the convolutional kernel transfer an input layer with the channels number of C_1 to a output layer with the channels number of C_2 ($C_2 \geq C_1$). So we have $C_1^{j^1} = C_2^{j^2} = \cdots = C_G^{j^G} = \frac{C_2}{G}$, where $j^1, j^2, \cdots, j^G$ is an permutation of G, this means that we can start with any of these groups to unfolded the convolutional layer into a block structure. We will explore the changes in the parameters of a convolutional layer convert into a block structure. We assume the parameters of the convolution kernel which used in above two structure are K (if we use 1×1 convolution, $K = 1$, if we use 3×3 convolution, $K = 9$). Obviously, the parameters of the traditional conventional layer are KC_1C_2. We can calculate the parameters P of the block structure in follow equation:

$$\begin{aligned} P &= \sum_{j=1}^{G} K(C_1 + (j-1)\frac{C_2 - C_1}{G})\frac{C_2}{G} \\ &= K\frac{C_2}{G} \cdot GC_1 + K\frac{C_2}{G} \cdot \frac{G(G-1)}{2} \cdot \frac{C_2 - C_1}{G} \\ &= KC_1C_2 + K\frac{(G-1)}{2G}C_2(C_2 - C_1) \\ &\leq KC_1C_2 + \frac{1}{2}KC_2(C_2 - C_1) \\ &= \frac{1}{2}KC_2(C_2 + C_1) \end{aligned} \tag{12}$$

From eq. (12), we can find that the block structure and the traditional single convolution layer have the same parameters if $C_2 = C_1$. In fact, this is common in present networks. If $C_2 > C_1$, the number of the parameters of the block structure will be $\frac{1}{2}KC_2(C_2 - C_1)$ more than the convolutional layer if G becomes larger. Fig. 3 shows the process of unfolding a single convolution layer into 6 layers.

If $C_2 < C_1$, the block has to project the channels from C_1 to C_2, so there will be some channels that do not have the corresponding channels for addition because the bypass structure in eq. (1) does not compress the channels. When using bypass structure, we can choose to drop out some channels of the group if the lost information accounted for the proportion is not large, or add a group of channels to the latter group in overlapping ways, which maps multiple channels of the former group to one of the latter groups. This allows all information flow from the front layer to the back layer without introducing additional parameters.

From the above analysis we can conclude that the parameters of the two structures are almost the same. This method just divide the convolutional kernels into G groups, this means the traditional convolutional layer can be expanded into whole-part block forms by divides the channels into G groups. So we can think of the traditional convolutional layer structure as special whole-part block with a group number of $G = 1$, where G also represents the depth of the

block structure. In other words, the more the groups, the deeper the network, while the network parameters remain unchanged. In one extreme case, the layer with the number of channels C is divided into $C(G = C)$ groups, which would cause the neural network spreading rapidly to hundreds of layers with minimal parameters.

IV. NETWORK ARCHITECTURES

In this section, we introduce the network architecture of WPNets and PWNets in detail. Table 1 describes the general architecture of the network.

TABLE 1. General network architecture of WPNets and PWNets. The number of group in block i is g_i. Note that each "conv" layer shown in the table corresponds the sequence BN-ReLU-Conv, and the "avg" is shorthand for average.

Layers	Output Size	Network
Convolution	$n \times n$	3×3 conv, stride 1
Block (1)	$n \times n$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times g_1$
Transition (1)	$n \times n$	1×1 conv
	$\frac{n}{2} \times \frac{n}{2}$	2×2 avg pool, stride 2
Block (2)	$\frac{n}{2} \times \frac{n}{2}$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times g_2$
Transition (2)	$\frac{n}{2} \times \frac{n}{2}$	1×1 conv
	$\frac{n}{4} \times \frac{n}{4}$	2×2 avg pool, stride 2
$\vdots$	$\vdots$	$\vdots$
Block (i)	$\frac{n}{2^i} \times \frac{n}{2^i}$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times g_i$
Classification	1×1	$\frac{n}{2^i} \times \frac{n}{2^i}$ global avg pool
Layer		fully-connected,softmax

A. INITIAL LAYER

The initial layer is important to determine the number of groups G. We assume that the number of channels in the initial layer is C, we can choose any integer as the number of groups G if C is divisible by it. Of course, we can also precisely set the number of channels in each group. For simplicity, we specify the number of channels per group to be $C' = \frac{C}{G}$.

B. BLOCK

1) WPNets BLOCK

The 1×1 convolution and 3×3 convolution are used to implement the compression process in WPNets block, which is called bottleneck layers in ResNets [4]. The 1×1 convolution in composition function f reduces the input to $4C'$, and then the followed 3×3 convolution compresses the previous output to C'. Note that composition function f include BN [21] and ReLU [22] operations before each convolution layer. The number of group channels can be increased in the compression process, making it R times as many as the original channels, where R is called growth rate. In this paper, we use $R = 2$ in the WPNets architecture.

Different from the analysis in section III-D, the block of WPNets will reduce the number of parameters as the number of group increases. This is because we use 1×1 convolution in the compressed function, which first compresses the number of channels to $4C'$, and then compresses it to C' using 3×3 convolution. If G is increased, C' will be reduced. 1×1 convolution will be used more, and 3×3 convolution will be used less. The parameters of 1×1 convolution are much less than the parameters of 3×3 convolution, the parameters of the block will decrease with the increase of the number of the group.

2) PWNets BLOCK

Contrary to WPNets, the 3×3 convolution is first used to amplify each group of channels to $4C'$, and then the 1×1 convolution is used to amplify the previous output into C channels. Finally, The amplified channels information of the group are added to the original channels. In this case, the number of channels in the block will remain unchanged.

We can use more filters to increase the number of channels in the network, and also can leave a part of the amplified group channels, and concatenate this part of the channels directly into the back layer to achieve the increase the number of the channels. The addition operations are used for overlapping channels the number of which is equal to $G - 1$ times of the number of group channels. The concatenation operations are used for the redundant channels is equal to the number of group channels, so the number of the growth channels in each group is equal to C'.

C. TRANSITION

In the end of the block, the 1×1 convolution is used to implement the features information fusion and use average pooling with stride 2 to down sampling features. The number of channels will be increased by 2 times if the number of channels does not increase in the block structure, otherwise the number of channels will remain unchanged. We find this structure is every useful in the network of group convolution, and it was used behind each block, except for the last one.

V. EXPERIMENTS

A. DATASETS

We use representative benchmark datasets: CIFAR-10 [24] and CIFAR-100 [24] to evaluate the performance of our algorithm. CIFAR-10 and CIFAR-100 each contain 32×32-pixel color images, consisting of 50k training images and 10k testing images, respectively. In CIFAR-10, it includes 10 classes, and CIFAR-100 includes 100 classes. Channel means are computed and subtracted in preprocessing. We also apply standard augmentation [4], [5], [16], [25]–[29]: horizontal flipping and translation by 4 pixels are adopted in

TABLE 2. Error rates (%) on CIFAR and SVHN datasets. *C* is the number of channels in the initial layer, and *G* is the number of groups in each block. "+" indicates standard data augmentation. Models that contains too many parameters are not presented in this table.

Method	Depth	Params	C10	C10+	C100	C100+	SVHN
Network in Network [26]	-	-	10.41	8.81	35.68	-	2.35
All-CNN [28]	-	-	9.08	7.25	-	33.71	-
Deeply Supervised Net [25]	-	-	9.69	7.97	-	34.57	1.92
ResNet [4]	110	1.7M	-	6.61	-	-	-
ResNet with Stochastic Depth [29]	110	1.7M	11.66	5.23	37.80	24.58	1.75
	1202	10.2M	-	4.91	-	-	-
Wide ResNet [17]	16	11.0M	-	4.81	-	22.07	-
ResNet (pre-activation) [35]	164	1.7M	-	5.46	-	24.33	-
	1001	10.2M	-	4.62	-	22.71	-
DenseNet [7]	40	1.0M	7.00	5.24	27.55	24.42	1.79
DenseNet [7]	100	7.0M	5.77	4.10	23.79	20.20	1.67
DenseNet-BC [7]	100	0.8M	5.92	4.51	24.15	22.27	1.76
WPNet ($C = 50, G = 10$)	64	1.4M	5.86	4.37	24.58	21.96	1.71
WPNet ($C = 100, G = 25$)	154	4M	**5.45**	**4.10**	**21.78**	**20.03**	**1.70**
PWNet ($C = 50, G = 10$)	64	0.3M	6.88	5.67	27.89	25.33	1.76
PWNet ($C = 200, G = 25$)	154	8M	**5.41**	**4.03**	**21.31**	**19.75**	**1.56**

our experiments. We denote this augmentation scheme by a "+" mark at the end of the datasets name (marked as C10+ and C100+).

We also conducted more studies on the SVHN [30] datasets. It is a real-world dataset obtained from house numbers in Google Street View images. It consists 10 classes, where 73257 digits for training, 26032 digits for testing, and 531131 additional. All digits have been resized to 32-by-32 pixels. In order to facilitate comparison with other networks, we did not use data augmentation like common practice [25], [26], [29], [31], and separated 6k images from the training set as verification sets. We follow [17], [29] and divide the pixel values by 255 so that these pixel values are in the range [0, 1].

The ILSVRC 2012 classification dataset [32] contains 1.2 million images for training, 50k for verification, and a total of 1000 pre-defined category labels. We use the same data augmentation scheme as [7] for the training image and apply 224×224 center cropping to the image during testing. On this data set, in order to make a better comparison with DenseNet [7], we use DenseNet's [7] network architecture, but replace its block structure with the block structure of WPNets and PWNets, while keeping all hyperparameters consistent with DenseNets [7].

B. TRAINING SETTING

We use random gradient descent (SGD) to train all networks. On CIFAR [24], we trained 300 epochs using batch size of 64. While on SVHN [30], the network is trained 40 epochs. The initial learning rate is set to 0.1, which reduces the learning rate by a factor of 10 at 50% and 75% of the total number of training epochs. Following [7], we use a weight decay of 10^{-4} and a Nesterov momentum [33] of 0.9. A dropout [34] layer was added after each convolutional layer except the first, and the drop rate was set to 0.2. On ImageNet [32], we train models for 90 epochs with a mini-batch size of 256, and other hyperparameter settings are the same as DenseNets [7].

C. CLASSIFICATION PERFORMANCE ON CIFAR AND SVHN

The efficiency of parameters is the key to our method. We observe that even without a large number of parameters, our models obtain accuracy that is comparable with many state-of-the-art methods (Table 2).

1) WPNets ACCURACY

The number of channels will be increased in each layer of the block. We find that such a design makes the performance of the network slightly lower, but the number of parameters of the network is greatly reduced. The number of initial channels C corresponding to the table 2 are 50, 100, and 200, and the number of groups G in each block are the same. After we compare the tradeoff between parameters and performance, it shows that our models obtain comparable results even with far fewer parameters compared to many state of the arts. For example, our WPNets with 1.4M parameters outperforms ResNet [4] and some of its variants, such as Wide ResNets [17], ResNet with Stochastic Depth [29] and ResNet with pre-activation [35]. This model also achieves good performance on the SVHN [30] datasets. The error rate is 1.7%, exceeding almost all networks which with more parameters. We can find that performance of WPNets with $C = 100$ and $G = 25$ surpasses the shallow model which with $C = 50$ and $G = 10$ on CIFAR [24] datasets, but its performance does slightly increase on SVHN [30] datasets. We summarize that deep model tends to overfit to the training set. These experimental results show that the whole-part structure is very effective.

2) PWNets ACCURACY

To test the performance of PWNets, we trained a network with 0.3M parameters, and we found that the performance exceeded some ResNets and its variants with

1.7M parameters on CIFAR [24] datasets without data augmentation. On the SVHN datasets, it achieved the same result as Densenet-BC with 0.8M parameters. The PWNets with $C = 200$ and $G = 50$ gained a good tradeoff between performance and parameters. This model outperformed all models with the same parameter level and achieved the state of the art on the SVHN datasets. These results indicate that PWNets utilize parameters more effectively than alternative model architectures.

D. CLASSIFICATION PERFORMANCE ON ImageNet

We replace the block structure in DenseNets [7] with the block structure of WPNets and PWNets, so the depth of the three networks is the same. It should be noted that the initial number of channels needs to be set for WPNets and PWNets. As can be seen from Figure 4, the network parameters increase (the number of connections in the network is increased), while the performance has slightly improved. We carefully designed the initial number of channels for WPNets and PWNets to ensure they have the same order of magnitude as DenseNets [7]. As analyzed above, WPNets and PWNets will have more connections than DenseNets [7], which ensures that they can achieve better generalization performance. For example, when the amount of parameters is almost the same, the performance of the two networks is improved compared to DenseNets [7].

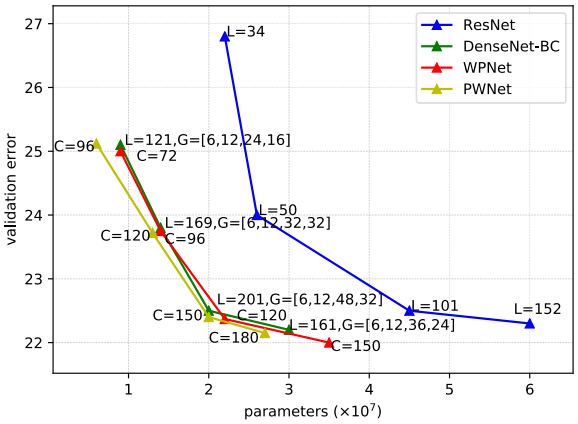


FIGURE 4. Performance comparison of various network structures on ImageNet. *L* represents the number of layers, *G* represents the number of blocks in DenseNets, and represents the number of groups in WPNets or PWNets. *C* represents the initial number of channels for WPNets and PWNets. When the grouping of each block structure is the same, the depths of WPNets, PWNets, and DenseNets are the same, so *L* and *G* in the figure are shared by three networks.

E. EFFECT OF DEPTH ON PERFORMANCE

It has been shown that deep model classes have an exponential advantage to represent certain natural target functions when compared to shallow model classes. But, this situation is not useful in all cases. Sun *et al.* [36] showed that with the increasing depth, the test error of neural networks may first decrease, and then increase. We conduct experiments to demonstrate the relationship between the groups/depth and performance of the network proposed in this paper. We do not use the 1×1 convolution used in the compress function of the WPNets and amplification function of PWNets. We use the 1×1 convolution in transition structure to expand the number of channels by 2 times, so the number of channels in each block will remain unchanged. This ensures that the two networks have the same amount of parameters, total 2M in this experiments. We compare the two kinds of blocks with groups number G value in $\{1, 5, 10, 25\}$. Note that the $G = 1$, there are the traditional networks and the number channels in each block are keep unchanged.

TABLE 3. The influence of the depth of WPNets on its performance.

Group	Depth	C10	C10+	C100	C100+
1	7	8.48	7.39	29.46	28.39
5	19	6.54	5.0	26.44	25.0
10	34	6.4	4.93	26.13	**24.23**
25	79	6.18	4.88	25.81	**24.25**

TABLE 4. The influence of the depth of PWNets on its performance.

Group	Depth	C10	C10+	C100	C100+
1	7	8.48	7.39	29.46	28.39
5	19	6.71	5.23	26.63	24.03
10	34	6.51	**4.92**	26.57	23.73
25	79	6.43	**5.03**	26.48	23.57

The performance in CIFAR [24] datasets are showed in table 3 and 4. From these two tables, we can find that with the group's number growthing, the performance is almost always improving. But we also find that the performance of the network is saturated in some cases. For example, the WPNets with a group number of 25 do not have a higher performance than group number of 10 on C100+. And in C10+, the performance of PWNets with a group number of 25 is a little worse than the group number of 10. When the number of groups/depth is the same, WPNets and PWNets can achieve very close performance, which shows that the two structures have some potential symmetries.

F. PARAMETER EFFICIENCY

In order to prove the effectiveness of network parameters, we use very few parameters to train WPNets and PWNets with different depth. These network structures are mainly compared to ResNets [4]. We want to replace the block structure of ResNets [4] with the block structure of WPNets or PWNets without sacrificing performance. In the original implementation of Resnets, it is divided into 3 block, and the number of channels for each block is 16, 32, and 64. The WPNets and PWNets we implement are very similar to ResNets [4]. We find that if the number of channels in the two networks remains the same as that of ResNets [4], the parameter of our model is less than 0.1M, which will be underfitting in CIFAR-10. So, we expand the number of channels for each block in our model by 2 times, and the

number of channels for each block is 16, 32, and 64. In this way, our network still has minimal parameters, such as the parameters that PWNet-52 has 0.15M, which has less than 10 times the amount of parameters than Resnets [4].

TABLE 5. The effectiveness of network parameters. ∗ indicates results run by ourselves.

Method	Depth	Params	C10	C10+
FitNet [27]	19	2.5M	-	8.39
Highway [5]	19	2.3M	-	7.72
	32	1.25M	-	8.80
ResNet [4]	44	0.66M	11.66*	7.17
	56	0.85M	11.41*	6.97
	110	1.73M	11.53*	**6.61**
WPNet	52	0.20M	**9.19**	7.05
	100	0.16M	**9.59**	6.99
PWNet	52	0.11M	**9.86**	7.22
	100	0.15M	**9.17**	**6.75**

In Table 5, we found that ResNets [4] has a over fitting problem on C10: the performance of ResNet-110 [4] is lower than that of ResNet-56 [4], but on C10+ this is the opposite. This phenomenon can also be found on WPNet: the increase in network depth only gains performance in C10+, but has a significant performance decline on C10. In general, our network can achieve almost the same performance as ResNets [4] under the condition that the number of layers is not very different from that of ResNets [4], and all of them exceed the performance of HighwayNets [5] and FitNets [27] with more parameters. This further shows that our network structure can replace ResNets's network structure, which can not only train and infer more quickly, but also greatly reduce the amount of parameters.

VI. DISCUSSION

We will discuss the relationship between the network architecture proposed in this paper and other network architectures. The channels information relationship is shown in fig. 5, each line in the figure being a group of features. The brackets ''[ ]'' are the convolution operations, to which the line it is connected add or concatenate the convolution results into the sequence layers.

We can find that the ResNets [4] adds all the channels of the front layer to the back layer, and the information fusion between the front and back layers is accomplished by the addition operation. DenseNets [7] transfer the information directly to the back layer. We can simply change the addition operation of ResNets [4] into the concatenation operation to implements the DenseNets [7].

A. DIFFERENT FROM ResNets

While Both WPNets and PWNets divide the channels into G groups, WPNets learns only the residual structure of one group at a time, until the whole block structure learns the residual structure. PWNets add a group of channel information to the whole channel information, which is a partial

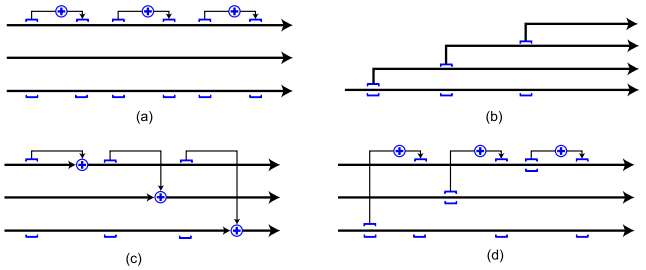


FIGURE 5. Channel information flow in various network architectures. (a) ResNets [4], (b) DenseNets [7], (c) WPNets, (d) PWNets. "+" represents the addition operation of the channels, the line with arrows represents the flow direction of the channel information, and brackets represent complex functions that contain convolution operations and other operations (e.g., BN [21] and ReLU [22]).

to whole residual structure. In the structure of WPNets and PWNets, $j-1$ times residuals have been added before convolution operations are performed in group j. The residual information comes from the previous $j-1$ groups. Because the addition operation of the residual structure is a linear transformation between the channels, the group j can utilize the information of other $j-1$ groups.

B. DIFFERENT FROM DenseNets

We first analyze the DenseNets [7] and its equivalent form. Except the transition layers, the intermediate layers of DenseNets [7] are obtained by concatenating all previous layer. If we decompose the block into some group convolution structure then add to produce a new layer. This equal form explicit shows all the connections between new layer and its previous layers. In the block, the number of connections from the input layer begins with 1, increased by layer by L, so the layer l^{th} will have L connections to its previous layers, which sums up to $\frac{L\times L}{2}$.

In WPNets, the group number is equal to that of the layers in the block of DenseNets [7]. All group features are used as the input layer to connect to one of the groups of it. This group features will be added by the convolution results of other group's layer. So just the added group's features have changed, the others will flow to its sequence layers without changing. We did it in an iterative way until all the features of the groups have changed. We can find the connections of above nets architecture will be up to the number of $G\times G$, half of it being the connections between groups, and the other half the connections between layers. The additional connections will help improve performance.

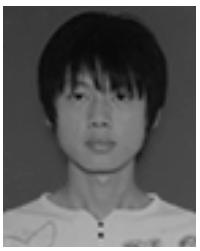
VII. CONCLUSION

In this paper, we proposed two network architectures by analyzing the whole and part of the channel relations in convolutional neural networks, which we refer to the whole-to-part connected networks (WPNets) and part-to-whole connected networks (PWNets). In WPNets, the whole channel information is compressed into one of the groups in each layer, forming a whole to part residual structure.

Contrary to the compression process, PWNets amplifies the partial channel information to the whole channel information, forming a part to whole residual structure. The compression and amplification processes of these two structures contain addition and concatenation operations, in which the addition operation forms a residual structure, and the concatenation operation ensures that the original information is passed to the back layer. After we compare the tradeoff between parameters and performance, it is shown that our models obtain comparable results even with far fewer parameters compared to many state of the arts. and the two structures can replace the traditional convolutional layer while keeping the parameters slightly increased or unchanged. We are going to study dynamic neural networks where the number of network layers increases and the parameters remain unchanged.

REFERENCES

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proc. Int. Conf. Adv. Neural Inf. Process. Syst.*, 2012, pp. 1–9.
[2] O. Russakovsky *et al.*, "ImageNet large scale visual recognition challenge," *Int. J. Comput. Vis.*, vol. 115, no. 3, pp. 211–252, Dec. 2015.
[3] C. Szegedy *et al.*, "Going deeper with convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 1–9.
[4] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 770–778.
[5] R. K. Srivastava, K. Greff, and J. Schmidhuber, "Training very deep networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 2377–2385.
[6] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He, "Aggregated residual transformations for deep neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 1492–1500.
[7] G. Huang, Z. Liu, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proc. CVPR*, Jun. 2016, pp. 4700–4708.
[8] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE Trans. Neural Netw.*, vol. 5, no. 2, pp. 157–166, Mar. 1994.
[9] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," *J. Mach. Learn. Res.*, vol. 9, pp. 249–256, May 2010.
[10] Y. LeCun *et al.*, "Backpropagation applied to handwritten zip code recognition," *Neural Comput.*, vol. 1, no. 4, pp. 541–551, 1989.
[11] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2323, Nov. 1998.
[12] B. Hariharan and P. Arbeláez, R. Girshick, and J. Malik, "Hypercolumns for object segmentation and fine-grained localization," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2015, pp. 447–456.
[13] E. Shelhamer, J. Long, and T. Darrell, "Fully convolutional networks for semantic segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 4, pp. 640–651, Apr. 2017.
[14] P. Sermanet, K. Kavukcuoglu, S. Chintala, and Y. Lecun, "Pedestrian detection with unsupervised multi-stage feature learning," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2013, pp. 3626–3633.
[15] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2016, pp. 2818–2826.
[16] G. Larsson, M. Maire, and G. Shakhnarovich, "Fractalnet: Ultra-deep neural networks without residuals," in *Proc. ICLR*, 2017, pp. 1–11.
[17] S. Zagoruyko and N. Komodakis. (2016). "Wide residual networks." [Online]. Available: https://arxiv.org/abs/1605.07146
[18] J. Wang, Z. Wei, T. Zhang, and W. Zeng. (2016). "Deeply-fused nets." [Online]. Available: https://arxiv.org/abs/1605.07716
[19] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 1251–1258.
[20] A. G. Howard *et al.* (2017). "Mobilenets: Efficient convolutional neural networks for mobile vision applications." [Online]. Available: https://arxiv.org/abs/1704.04861
[21] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proc. ICML*, 2015, pp. 1–11.
[22] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *Proc. Int. Conf. Artif. Intell. Statist.*, 2012, pp. 1–9.
[23] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer. (2016). "SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size." [Online]. Available: https://arxiv.org/abs/1602.07360
[24] A. Krizhevsky, "Learning multiple layers of features from tiny images," M.S. thesis, Dept. Comput. Sci., Univ. Toronto, Toronto, ON, USA, 2009.
[25] C.-Y. Lee, S. Xie, P. Gallagher, Z. Zhang, and Z. Tu. (2014). "Deeply-supervised nets." [Online]. Available: https://arxiv.org/abs/1409.5185
[26] M. Lin, Q. Chen, and S. Yan, "Network in network," in *Proc. ICLR*, 2014.
[27] A. Romero, N. Ballas, S. E. Kahou, A. Chassang, C. Gatta, and Y. Bengio, "Fitnets: Hints for thin deep nets," in *Proc. ICLR*, 2015.
[28] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller. (2014). "Striving for simplicity: The all convolutional net." [Online]. Available: https://arxiv.org/abs/1412.6806
[29] G. Huang, Y. Sun, Z. Liu, D. Sedra, and K. Q. Weinberger, "Deep networks with stochastic depth," in *Proc. Eur. Conf. Comput. Vis.*, 2016, pp. 646–661.
[30] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng, "Reading digits in natural images with unsupervised feature learning," in *Proc. NIPS Workshop Deep Learn. Unsupervised Feature Learn.*, 2011, pp. 1–9.
[31] I. J. Goodfellow, D. Warde-Farley, M. Mirza, A. Courville, and Y. Bengio, "Maxout networks," in *Proc. ICML*, 2013.
[32] J. Deng, W. Dong, R. Socher, L. J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2009, pp. 248–255.
[33] I. Sutskever, J. Martens, G. Dahl, and G. Hinton, "On the importance of initialization and momentum in deep learning," in *Proc. Int. Conf. Int. Conf. Mach. Learn.*, 2013, pp. 1139–1147.
[34] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1929–1958, 2014.
[35] K. He, X. Zhang, S. Ren, and J. Sun, "Identity mappings in deep residual networks," in *Computer Vision—ECCV*, B. Leibe, J. Matas, N. Sebe, and M. Welling, Eds. Cham, Switzerland: Springer, 2016, pp. 630–645.
[36] S. Sun, W. Chen, L. Wang, X. Liu, and T.-Y. Liu, "On the depth of deep neural networks: A theoretical view," in *Proc. 13th AAAI Conf. Artif. Intell.* Phoenix, AZ, USA: AAAI Press, 2016, pp. 2066–2072.

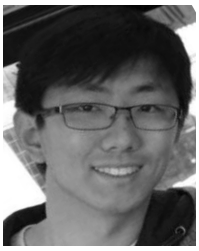


DAOJUN LIANG received the B.S. degree in computer science from Taishan University, China, in 2016. She is currently pursuing the master's degree with the School of Information Science and Engineering, Shandong Normal University. Her research interests include deep learning and computer vision.

FENG YANG received the B.S. degree and the M.S. degree in electronics and communications from Shandong University in 1985 and 1988, respectively. He is currently a Professor with the School of Information Science and Engineering, Shandong Normal University, where he is also the Director of the Department of Communication Engineering. He presided over five provincial and university-level education reform projects. He participated in one national fund project. He has published more than 30 papers and written five books. He holds seven national invention patents and four utility model patents. He was a recipient of the provincial-level Teaching Achievement Award and the school-level Teaching Achievement Award. He was a recipient of three provincial awards for scientific and technological progress.

TIAN ZHANG received the B.S. and M.S. degrees from Shandong Normal University, Jinan, China, in 2006 and 2009, respectively, and the Ph.D. degree from Shandong University, Jinan, in 2014. He was a Visiting Ph.D. Student with Tsinghua University from 2010 to 2013. Since 2014, he has been with Shandong Normal University, where he is currently an Associate Professor. His research interests include wireless communications and smart grid. He served as a TPC member for the IEEE GLOBECOM 2017. He was a recipient of the 2016 Shandong Province Higher Educational Science and Technology Award (third class), the 2015 Excellent Doctoral Dissertation Award of Shandong University, and the 2010 Science and Technology Progress Award of Shandong Province (second class).

JIE TIAN received the B.E. and M.E. degrees from Shandong Normal University, Jinan, China, in 2008 and 2011, respectively, and the Ph.D. degree in communication and information systems from the School of Information Science and Engineering, Shandong University, China, in 2016. She is currently a Lecturer with the School of Information Science and Engineering, Shandong Normal University. Her research interests include cross-layer design of wireless communication networks, radio resource management in heterogeneous networks, and signal processing for communications.



PETER YANG received the B.S. degree from Shandong University, Jinan, China, in 2013, and the M.S. degree from the University of California, Irvine, CA, USA, in 2015. He is currently a Software Engineer with Amazon, where he is responsible for Amazon's web services. His research interests include machine learning, software engineering, and data analysis.