Deeply-Fused Nets

Jingdong Wang[1], Zhen Wei[2], Ting Zhang[3], Wenjun Zeng[1]

[1]Microsoft Research, [2]Shanghai Jiao Tong University
[3]University of Science and Technology of China

Abstract. In this paper, we present a novel deep learning approach, deeply-fused nets. The central idea of our approach is deep fusion, i.e., 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 resulting deeply fused net enjoys several benefits. First, it is able to learn multi-scale representations as it enjoys the benefits of more base networks, which could form the same fused network, other than the initial group of base networks. Second, in our suggested fused net formed by one deep and one shallow base networks, the flows of the information from the earlier intermediate layer of the deep base network to the output and from the input to the later intermediate layer of the deep base network are both improved. Last, the deep and shallow base networks are jointly learnt and can benefit from each other. More interestingly, the essential depth of a fused net composed from a deep base network and a shallow base network is reduced because the fused net could be composed from a less deep base network, and thus training the fused net is less difficult than training the initial deep base network. Empirical results demonstrate that our approach achieves superior performance over two closely-related methods, ResNet and Highway, and competitive performance compared to the state-of-the-arts.

Keywords: Deep neural network, deep fusion, recognition.

1 Introduction

Deep neural network has been popular again since the breakthrough performance [1] in the ImageNet classification [2]. In the past few years (from 2012), the top-5 classification accuracy on the 1000-class ImageNet dataset has increased from $\sim 84\%$ [1] to $\sim 97\%$ [3]. Besides, deep neural network has been shown to have very impressive performance for other vision applications, such as object detection [4, 5], image segmentation [6], edge detection [7], and so on.

Nevertheless, the fundamental problem, learning a deep hierarchical structure effectively and efficiently, still remains a challenge and has been attracting a lot of research efforts. Dropout [8] and other regularization techniques, such as weight decay and path regularization [9], have been developed to prevent neural network from over-fitting. Normalized variance-preserving weight initialization,

such as [10–12], has been shown to be helpful for handling the vanishing gradient problem and thus boosts the performance. Batch normalization [13] is shown to improve both the training speed and the recognition performance. Skip-layer connections between layers (including the output layer) and other network structure modifications, such as deeply-supervised nets [14] and its variant [7], Highway [15], ResNet [16], inception-v4 [3], are able to improve the flow of information and accordingly help train a very deep network. The teacher-student framework shows that learning a deep network can benefit from an already-trained network that is relatively easy to be learnt, e.g., FitNets [17] and Net2Net [18].

In this paper, we introduce a deep fusion approach and present a deeply-fused neural net formed by combining a group of base networks. The main idea is to perform fusion over the intermediate representations of the base networks, where the fused output serves as the input of the remaining part of each base network, rather than only over the final representations or the final classification scores, and such fusions are performed several times at different intermediate layers. There is a block-exchangeable property (the block is the subnetwork between two successive fusions in a base network): switch blocks from one base network to another one within one fusion, resulting in two different base networks with possibly different depth from the originals (e.g., deep network being less deep and shallow network being less shallow), but the fused net is not changed. In other words, a fused net can be formed by different groups of base networks. Thus, the deeply-fused net is able to learn multi-scale representations from much more base networks, and even same-scale representations can be different and learnt from different base networks.

There is one more benefit from deep fusion: the flow of information is improved. Consider the case where one base network is very deep but the other base network is not deep, which is the choice we suggest. The earlier intermediate layer in the deeper base network might have a shorter path through the other base network to the output, which implies that the supervision can be fast transformed to the earlier intermediate layer. On the other hand, the later intermediate layer might also have a shorter path from the input, which indicates that the input can be fast flowed to the later intermediate layer. As a result, training the fused net composed from a very deep base network is less difficult than training the very deep base network itself. Furthermore, the deep and shallow base networks are jointly learnt and can benefit from each other. We also show that the recently-developed networks, Highway and ResNet, can be viewed as specific examples of deep fusion. Empirical results demonstrate that our approach achieves superior performance over the plain network, the naive network fusion method, ResNet and Highway, and competitive performance compared to the state-of-the-arts.

2 Related Work

The past few years have witnessed the rapid and great progress of deep neural networks in various aspects, from optimization techniques as well as initializa-

tion, regularization, activation and pooling functions, network structure design, to applications. In this section, we mainly discuss two closely-related lines: network structure design and network optimization with the aid of another already-trained network.

Averaging over a set of network predictors, which we call decision fusion, is able to improve the generalization accuracy and has been widely used, e.g., to boost the ImageNet recognition performance [1, 16, 19, 20]. Multi-column deep neural networks [21] presents an empirical study about decision fusion, later extended to an adaptive version, weighted averaging with the weights depending on the input [22]. The averaging approach learns each network separately, which is equivalent to learn the network jointly that averages the loss functions. Our approach, in contrast, performs the feature fusion deeply over several intermediate layers and simultaneously learns the representations of the (base) networks.

The inception module in GoogLeNet [20] can be viewed as a fusion stage: concatenate the outputs of several subnetworks with different lengths. It is different from our approach using the summation for fusion. The GoogLeNet architecture, consisting of a sequence of inception modules, is also a kind of deep fusion, i.e., deep concatenation fusion. But it is not as direct as our deep summation fusion. The output of each subnetwork in an inception module is narrower than the input of the subsequent inception module. Hence it is necessary to append many channels with all 0 entries in the output to match the size with the input of the subsequent inception module or add more convolution operations to form the fused network. Skip-layer connection, such as deeply-supervised nets [14] and its variant [7], Highway [15], ResNet [16], as we will show, resembles our approach and can be regarded as special examples of our approach.

The teacher-student framework suggests that learning a hard-trained network can benefit from an easily-trained network. For instance, FitNets [17] uses the intermediate representation of a wider and shallower (but still deep) teacher net that is relatively easy to be trained, as the target of the intermediate representation of a thinner and deeper student net. Net2Net [18] also uses a teacher net to help train a (wider or deeper) student net, through a function-preserving transform to initialize the parameters of the student net according to the parameters of the teacher net. Our approach, in our suggested choice: including one deep base network and one shallow (but could still be deep) network, also uses the shallow network to help train the deep base network, meanwhile the deep base network also helps train the shallow network, i.e., they benefit from each other and are trained simultaneously.

3 Deeply-Fused Nets

A feedforward network typically consists of L representation extraction layers, H_L, followed by a classification layer, e.g., a fully-connected layer and a linear classifier. The lth layer applies a nonlinear transformation h_l (parameterized by $\mathbf{w}_l$), e.g., a linear convolution followed by a nonlinear activation function:

$$\mathbf{x}_l = h_l(\mathbf{x}_{l-1}; \mathbf{w}_l), \tag{1}$$

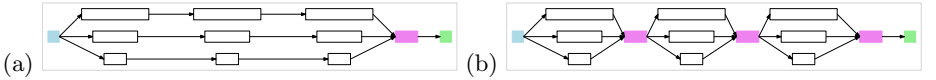


Fig. 1. Illustrating (a) shallow fusion and (b) deep fusion. The light blue, white, violet, light green boxes correspond to the input, a block of layers (subnetwork), the fusion layer, and the classification layer. (Best viewed in color.)

where $\mathbf{x}_{l-1}$ is the input of the lth layer and also the output of the $(l-1)$th layer, or is the input of the whole network if $l = 1$. The representation extraction part of the network can be written in a function form, $H_L(\mathbf{x}_0) = h_L(H_{L-1}(\mathbf{x}_0))$, where $H_l(\mathbf{x}_0) = h_l(H_{l-1}(\mathbf{x}_0))$.

3.1 Shallow Fusion

Network fusion is a process of combining multiple base networks, e.g., K base networks $\{H^1_{L_1}, \ldots, H^K_{L_K}\}$[1]. The conventional fusion in general includes two approaches: feature fusion, fusing the representations extracted from the networks together, followed by a classification layer; and decision fusion (a.k.a., model ensemble), fusing the classification scores computed from the networks. This paper focuses on feature fusion and the fusion can be formulated in the function form, $H(\mathbf{x}_0) = F(H^1_{L_1}(\mathbf{x}_0), \ldots, H^K_{L_K}(\mathbf{x}_0))$, where the fusion function $F(\cdot)$, in this paper, is the sum of the representations,

$$F(H^1_{L_1}(\mathbf{x}_0), \ldots, H^K_{L_K}(\mathbf{x}_0)) = \sum\nolimits_{k=1}^{K} H^k_{L_k}(\mathbf{x}_0). \tag{2}$$

The fusion function could be in other forms, e.g., concatenation or maximization, and we will discuss them later.

3.2 Deep Fusion

Deep fusion performs feature fusion not only over the final feature representation but also over the intermediate feature representations. The forward process and backward propagation are presented as follows.

Forward process. A network H^k_L with L feature extraction layers is divided into B blocks, $\{G^k_1, \ldots, G^k_b, \ldots, G^k_B\}$ where a block G^k_b is a sequence of several nonlinear transformations, $\{h^k_{b_{start}}, \ldots, h^k_{b_{end}}\}$:

$$G^k_b = h^k_{b_{end}}(h^k_{b_{end-1}}(\ldots(h^k_{b_{start}}(\mathbf{x}_{b-1}))\ldots)),$$

[1] For simplicity, we only use the representation extraction part $H^k_{L_k}$ to describe a full network.

or simply an identity connection. Deep fusion is a process of fusing K networks (assuming each consisting of B blocks), with B summation fusion operations:

$$\bar{\mathbf{x}}_1 = F_1(G_1^1(\mathbf{x}_0), G_1^2(\mathbf{x}_0), \ldots, G_1^K(\mathbf{x}_0)), \quad (3)$$

$$\bar{\mathbf{x}}_2 = F_2(G_2^1(\bar{\mathbf{x}}_1), G_2^2(\bar{\mathbf{x}}_1), \ldots, G_2^K(\bar{\mathbf{x}}_1)), \quad (4)$$

$$\ldots \quad (5)$$

$$\bar{\mathbf{x}}_B = F_B(G_B^1(\bar{\mathbf{x}}_{B-1}), G_B^2(\bar{\mathbf{x}}_{B-1}), \ldots, G_B^K(\bar{\mathbf{x}}_{B-1})), \quad (6)$$

where $F_b(G_b^1(\bar{\mathbf{x}}_{b-1}), G_b^2(\bar{\mathbf{x}}_{b-1}), \ldots, G_b^K(\bar{\mathbf{x}}_{b-1})) = \sum_{k=1}^K G_b^k(\bar{\mathbf{x}}_{b-1})$, $\bar{\mathbf{x}}_b$ is the output of the bth fusion, and it is assumed that the output sizes of the K blocks $G_b^k(\bar{\mathbf{x}}_{b-1})$ are the same. Figure 1 illustrates the difference between shallow and deep fusions.

Backward propagation. Gradient back-propagation is the same as the conventional back-propagation. Here, we present the back-propagation form with respect to the input and the output of each fusion, i.e., over the blocks. According to the definition, the gradient of $\bar{\mathbf{x}}_{b+1}$ with respect to $\bar{\mathbf{x}}_b$, called fused block gradient, can be computed as follows,

$$\frac{\partial \bar{\mathbf{x}}_{b+1}}{\partial \bar{\mathbf{x}}_b} = \sum\nolimits_{k=1}^K \frac{\partial G_{b+1}^k}{\partial \bar{\mathbf{x}}_b}, \quad (7)$$

which intuitively means that the gradient is the summation of block gradients $\{\frac{\partial G_{b+1}^k}{\partial \bar{\mathbf{x}}_b}\}_{k=1}^K$, and block gradient $\frac{\partial G_{b+1}^k}{\partial \bar{\mathbf{x}}_b}$ is the gradient of the $(b+1)$th block of the kth base network. Suppose the loss function is $O(\bar{\mathbf{y}}, \mathbf{y}^*)$, with $\bar{\mathbf{y}}$ and $\mathbf{y}^*$ being the estimated and ground-truth labels. The gradient with respect to the hidden response $\bar{\mathbf{x}}_b$ using back-propagation can be computed in the following form,

$$\frac{\partial O}{\partial \bar{\mathbf{x}}_b} = \frac{\partial \bar{\mathbf{x}}_{b+1}}{\partial \bar{\mathbf{x}}_b}\frac{\partial O}{\partial \bar{\mathbf{x}}_{b+1}} = \prod\nolimits_{t=b}^{B-1} \frac{\partial \bar{\mathbf{x}}_{t+1}}{\partial \bar{\mathbf{x}}_t}\frac{\partial O}{\partial \bar{\mathbf{x}}_B} = \prod\nolimits_{t=b}^{B-1} (\sum_{k=1}^K \frac{\partial G_{t+1}^k}{\partial \bar{\mathbf{x}}_t})\frac{\partial O}{\partial \bar{\mathbf{x}}_B}. \quad (8)$$

Base network selection. From the above descriptions, we can see that the computation complexity for both the forward and backward processes is almost equal to the complexity of all the base networks with the negligible element-wise addition cost. Therefore, deep fusion does not introduce additional parameters, nor increase the computation complexity.

Deep fusion typically chooses a very deep network, and a shallower (might also be deep) network in which each block contains only a convolution layer, with a few blocks/fusions (for example, each block corresponds to one scale). Consequently, the computation complexity is approximately equal to that of the very deep network. This nice property makes deep fusion comparable to the plain network (i.e., the deep one used for deep fusion), Highway [15], and ResNet [16] that includes some non-identity connections whose extra cost is similar to ours. In addition, training the fused net formed from a very deep base network and a shallow base network is less difficult than training the initial very deep base network because the fused net could be composed from a less deep base network,

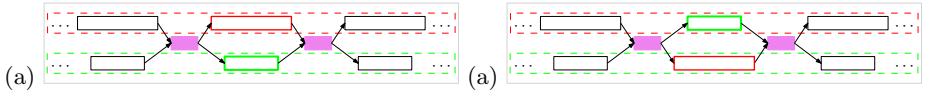


Fig. 2. Illustrating that the same fused net can be composed of a group of 2 base networks shown in (a) and another group of 2 different base networks shown in (b). The difference lies in that the middle block is exchanged. Similarly, the third block can be exchanged. Thus, many different base network groups can form the same fused net.

and a less shallow base network, which will be shown later, and the essential depth of a fused net is reduced.

Such a choice resembles the teacher and student framework (e.g., FitNets [17] and Net2Net [18]) where the student network is learnt from the already-trained teacher network. But in our approach, the teacher (shallow) and student (deep) networks are jointly learnt and benefit from each other. And the vanishing gradient problem if it seriously exists for the deep base network, is alleviated, according to the gradient back propagation shown in Equation (8). Besides, we will show that such a choice is advantageous in the flow of information.

3.3 Analysis

High capability of combining multi-scale representations. A deeply-fused net composed from a group of K base networks can also be composed from another group of K different base networks, which is shown below. Considering the mathematical formulation of deep fusion, we can rewrite Equation (4) into an equivalent form,

$$\bar{\mathbf{x}}_2 = F_2(G_2^1(\bar{\mathbf{x}}_1), G_2^2(\bar{\mathbf{x}}_1), \ldots, G_2^K(\bar{\mathbf{x}}_1)) = F_2(G_2^2(\bar{\mathbf{x}}_1), G_2^1(\bar{\mathbf{x}}_1), \ldots, G_2^K(\bar{\mathbf{x}}_1)),$$

which does not change the fused net. This property is called block exchangeability. It can be regarded as changing the first two base networks: $G_1^1 \rightarrow G_2^1 \rightarrow \cdots \rightarrow G_B^1$ and $G_1^2 \rightarrow G_2^2 \rightarrow \cdots \rightarrow G_B^2$, to two other base networks: $G_1^1 \rightarrow G_2^2 \rightarrow \cdots \rightarrow G_B^1$ and $G_1^2 \rightarrow G_2^1 \rightarrow \cdots \rightarrow G_B^2$. Similarly, we can obtain more base networks, but resulting in the same fused net. The number of unique combinations of K base networks can reach up to $(K!)^{B-1}$(In practice the number will be smaller than $(K!)^{B-1}$, but it is still very large). Figure 2 shows an example to illustrate this block-exchangeable property.

Each possible base network, in our implementation, will output an 8×8 feature map, in which each element corresponds to a receptive field with the size (scale) depending on the base network. The fused net can be formed from many base networks, and thus there exist many receptive fields with various sizes. In addition, the two base networks, with same sizes of receptive field, may have different extraction processes, and thus the extracted representations are different, and are able to capture different characteristics.

Improvement of the information flow. We show that an earlier intermediate layer might have a shorter path to the output layer. Consider an early intermediate representation, e.g., $\bar{\mathbf{x}}_1$, the shortest path to the final feature representation

is $\sum_{b=2}^{B} \min_{k=1,\ldots,K} |G_b^k|$, which intuitively means that for each fused block the smallest block is chosen as the path. This implies that the path from the intermediate layer in the deeper base network to the output becomes shorter, and thus the supervision can be fast flowed to an early intermediate layer.

Similarly, a later intermediate layer may have a shorter path from the input layer, indicating that the input information can be quickly fed into a later intermediate layer instead of through a long path. This benefit to the network learning is in some sense related to relay back-propagation [23], which explicitly fixes the earlier layers (something like directly connecting the input to the later layers) when updating the later layers. In summary, the flow of information from the input to the intermediate layers and from the intermediate layers to the output are both improved, which is beneficial to training a deep network.

4 Discussions

Concatenation, Maximization, and Summation. With the summation fusion in the intermediate layers, there is almost no change for each base network: the network structures are not changed. The only effect is that the output is changed with some signals added from other networks. Maximization fusion that performs an element-wise maximization,

$$F(H_{L_1}^1(\mathbf{x}_0), \ldots, H_{L_K}^K(\mathbf{x}_0)). = \max_{k=1,\ldots,K} \{H_{L_k}^k(\mathbf{x}_0).\},$$

is studied in [24]. Similar to summation fusion, there is almost no change for each base network. In contrast, with the concatenation fusion, the fusion function Equation (2) is changed to

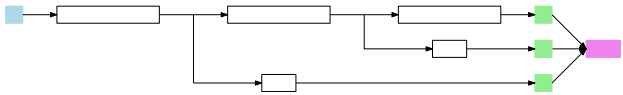
$$F(H_{L_1}^1(\mathbf{x}_0), \ldots, H_{L_K}^K(\mathbf{x}_0)) = [(H_{L_1}^1(\mathbf{x}_0))^\top, \ldots, (H_{L_K}^K(\mathbf{x}_0))^\top]^\top,$$

e.g., an inception module in GoogLeNet. The base networks have to be changed and more parameters are needed: the input size of the subsequent sub-network immediately after the fusion in each base network is increased as the fusion output becomes larger (or in the original base network, there are many channels with all 0 entries appended in the output of a block so that the total output matches the size with the input of the subsequent inception module). The combination of concatenation fusions and summation fusions is studied in [3], which shows better ImageNet performances.

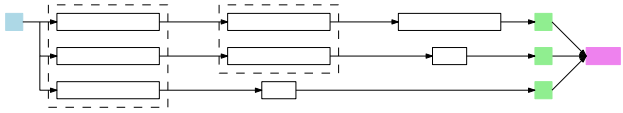
Relation to Deeply-Supervised Nets. The deeply supervised net estimates the network parameters through optimizing multiple losses, some of which come from the intermediate layers. The formulation could be written as follows,

$$\ell(C(H(\mathbf{x}_0), \mathbf{y}^*) + \sum\nolimits_{b=1}^{B} \ell(C_b(H_b'(\mathbf{x}_0)), \mathbf{y}^*), \tag{9}$$

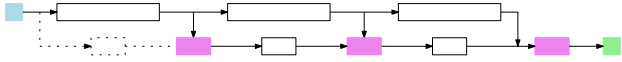
where $H_b'(\mathbf{x}_0)$ is a subnetwork of H and consists of the part from the input to the layer $h_{b_{end}}$, and $C(\cdot)$ and $C_b(\cdot)$ are the classifiers. A similar network used



(a) A deeply-supervised net



(b) A network equivalent to the deeply-supervised net



(c) Unidirectional fusion

Fig. 3. (a) a deeply-supervised net; (b) the shallow fusion view of the deeply-supervised net: the blocks in the dashed box share the same structure and parameters; (c) Unidirectional deep fusion can be regarded as an alternative of a deeply-supervised net. The dotted part could be removed if only the intermediate outputs, without the input, are flowed from the main network H_o to the below network H_i.

for edge detection [7] is shown to be able to combine multi-scale information. This formulation can be interpreted as a shallow decision/loss fusion process: combine $(B+1)$ networks with shared parameters across the $(B+1)$ networks $\{H'_1, \ldots, H'_B, H\}$, which is illustrated in Figure 3(b).

In addition, we show that the unidirectional version of deeply fused network is closely related to the deeply-supervised net with weights sharing for the classification layers. The unidirectional deeply fused network, (combining two networks, the signal from the network H_o is flowed to the network H_i) is mathematically given as follows,

$$\mathbf{x}_1^o = G_1^o(\mathbf{x}_0),\ \mathbf{x}_1^i = \mathbf{x}_1^o + G_1^i(\mathbf{x}_0), \tag{10}$$

$$\mathbf{x}_2^o = G_2^o(\mathbf{x}_1^o),\ \mathbf{x}_2^i = \mathbf{x}_2^o + G_1^i(\mathbf{x}_1^i), \tag{11}$$

$$\ldots \tag{12}$$

$$\mathbf{x}_B^o = G_B^o(\mathbf{x}_{B-1}^o),\ \mathbf{x}_B^i = \mathbf{x}_B^o + G_1^i(\mathbf{x}_{B-1}^i), \tag{13}$$

and a classification layer, $C^i(\mathbf{x}_B^i)$, is defined over $\mathbf{x}_B^i$.

Figure 3(c) shows an example of unidirectional deep fusion. Compared with the deeply-supervised net in Figure 3(a), we can observe that the unidirectional deep fusion uses progressive feature fusion, while deep supervision in deeply-supervised nets uses loss fusion.

Relation with Highway and ResNet. Skip-layer connection means that a layer can take not only the layer at the previous level as input but also some of the lower layers. It resembles deep fusion and in some sense they can be equivalently transformed to each other. Here we consider two recently-developed network examples: Highway [15] and ResNet [16].

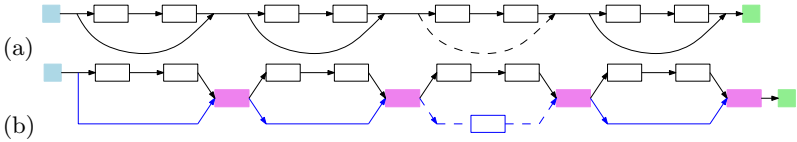


Fig. 4. Illustrating that ResNet is an example of deep fusion. The solid skip-layer connection is an identity connection, and the dashed skip-layer connection is a linear projection. From (b), we can see that ResNet is a fused network from the plain network and a short network (highlighted in blue).

Table 1. Base network architectures. In our implementation, the blocks are formed with the layers before FC1 (but in general, the block could also include FC1), and FC1 and Ip1 together are called the classification layer in this paper.

Network			N1	N2	N3	N4	N5	N6	N7
#Layers			19	50	5	8	10	11	14
Layer name	Output size	Parameters	Repeat times						
C1.	32×32	$(3 \times 3, 32)$	$\times 5$	$\times 16$	$\times 1$	$\times 2$	$\times 2$	$\times 3$	$\times 4$
2×2 max pool, stride 2									
C2.	16×16	$(3 \times 3, 80)$	$\times 6$	$\times 16$	$\times 1$	$\times 2$	$\times 3$	$\times 3$	$\times 4$
2×2 max pool, stride 2									
C3.	8×8	$(3 \times 3, 128)$	$\times 6$	$\times 16$	$\times 1$	$\times 2$	$\times 3$	$\times 3$	$\times 4$
FC1	8×8	$(1 \times 1, 100)$	$\times 1$	$\times 1$	$\times 1$	$\times 1$	$\times 1$	$\times 1$	$\times 1$
8×8 avg pool, stride 8									
Ip1	1×1	$(1 \times 1, 100$ or $10)$	$\times 1$	$\times 1$	$\times 1$	$\times 1$	$\times 1$	$\times 1$	$\times 1$

Highway and ResNet can be viewed as combining two networks: one is deep, called a plain network, and the other one is shallow, a sequence of virtual layers (identity connection) and possible extra layers, which are just down-sample pool layers (same to the plain network) in Highway, and are projection layers in ResNet (e.g., the blue box means a linear projection in Figure 4). Figure 4 illustrates that ResNet is an example of a deeply-fused net when the number of base networks is 2. Similarly, Highway can be transformed to deep fusion, and the difference is that the fusion in Highway is a weighted sum and the weight is data customized through the transform gate.

The block in the deep/plain network in Highway and ResNet is typically small, consisting of 1-3 layers. And thus there are many blocks/fusions, while in our approach the number of fusions is suggested smaller. We use the non-identity connection to form the block in the shallow network: usually one block in one scale (could be more in a very deep network), and no extra layer except pooling layer across scales; while ResNet uses non-identity connection to match the size which is changed when across scales in the network. Thus the number of parameters in ResNet and our approach are similar and both are smaller than that in Highway.

5 Experiments

We evaluate our approach on the CIFAR-10 and CIFAR-100 datasets. The CIFAR-10 and CIFAR-100 datasets [25] are both subsets drawn from the 80-

Table 2. Block division. If one block starts from C21 or C31, the block also includes the preceding pooling layer (in implementation, we only perform max pooling one time for all the base networks as the input and the operation are the same), and for clarity this is not explicitly described in the table. N33, N46, and N58 are shallow networks, with each block containing only one convolution layer. We use N13N33 to represent a fused net formed from base networks N1 and N3 with block division N13 and N33.

N13	N33	N43	N63	N73	N16	N26	N46	N18	N28	N58
3 blocks					6 blocks			8 blocks		
C11-C15	C11	C11-C12	C11-C13	C11-C14	C11-C12	C11-C18	C11	C11-C12	C11-C18	C11
					C13-C15	C19-C116	C12	C13-C15	C19-C116	C12
C21-C26	C21	C21-C22	C21-C23	C21-C24	C21-C23	C21-C28	C21	C21-C22	C21-C25	C21
								C23-C24	C26-C210	C22
					C24-C26	C29-C216	C22	C25-C26	C211-C216	C23
C31-C36	C31	C31-C32	C31-C33	C31-C34	C31-C33	C31-C38	C31	C31-C32	C31-C35	C31
								C33-C34	C36-C310	C32
					C34-C36	C39-C316	C32	C311-C316	C35-C36	C33

million tiny image database [26]. The CIFAR-10 dataset consists of 60000 32×32 colour images in 10 classes, with 6000 images per class. There are 50000 training images and 10000 test images. The CIFAR-100 dataset is like the CIFAR-10, except that it has 100 classes each containing 600 images. There are 500 training images and 100 testing images per class. The 100 classes in the CIFAR-100 are grouped into 20 superclasses. Each image comes with a "fine" label (the class to which it belongs) and a "coarse" label (the superclass to which it belongs).

The architecture of the base network is built upon basic units: the convolution layer, the nonlinear ReLU activation function, the max-pooling layer, the fully-connected layer, and the softmax layer for training, which are intentionally chosen to directly show the benefits from deep fusion. We also use a batch normalization layer right after each convolution layer. The details of the network architectures used in our experiments are presented in Table 1 (base networks) and Table 2 (block division).

We train the networks using the SGD algorithm, with the weight decay regularization, the momentum set to 0.9, the mini-batch size set to 100, and the maximum number of epochs set to 400. An exponentially decay learning rate is used: the learning rate is reduced by a factor of 10 after the 200th, 300th, 350th epoch. The results of our approach, the baseline algorithms and the plain network are reported with the weight decay coefficient and the initial learning rate tuned in the ranges: $\{0.0001, 0.0005\}$ and $\{0.01, 0.1\}$ for 19 layers, and $\{0.0002, 0.0005, 0.001\}$ and $\{0.02, 0.05, 0.1\}$ for 50 layers. The weights are initialized using the scheme [11]. Experiments are conducted using Caffe [27]. The datasets are preprocessed using a common setting, as described in [15], including the global contrast normalization, four-zero pixels padding at all borders with 32×32 random crops, and random horizontal flipping.

5.1 Empirical Analysis

Convergence curve. Figure 5 shows the training error and the testing error between a fused net, N13N33, composed of two base networks N13 and N33 (See

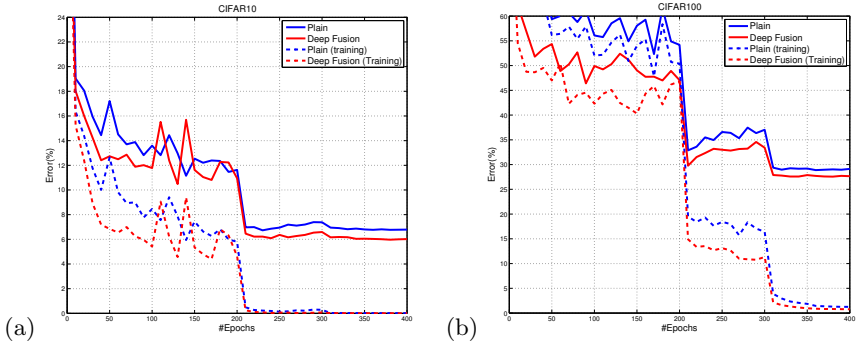


Fig. 5. Training and testing errors (%) vs. #epoches for the plain network and the deeply-fused net (N13N33) on (a) CIFAR-10 and (b) CIFAR-100.

Table 3. Accuracy (%) comparison between different fusion methods.

	CIFAR-10	CIFAR-100
19 layers (N13N33)		
Plain	93.5	70.87
Deep concatenation fusion	93.4	70.64
Deep max fusion	93.14	69.55
Shallow fusion	93.37	71.23
Decision fusion (separate train)	93.48	72.28
Decision fusion(joint train)	93.06	71.11
Deep summation (fusion before ReLU)	**93.98**	**72.29**
Deep summation (fusion after ReLU)	**93.77**	**72.64**
50 layers (N26N46)		
Plain	92.08	65.48
Decision fusion (separate train)	93.39	68.9
Deep summation	93.6	72.39

Table 2), and a plain network, N13, the deeper base network in the fused net. It can be observed that both the training error and testing error of deep fusion are lower than the plain network in the later iterations, and deep fusion achieves the same error with plain network using much fewer training steps.

Fusion. We present the results from our deep summation fusion, deep max fusion, deep concatenation fusion (the channel size of the convolution layer right before the pooling layer is reduced by half in order to match the size of the subsequent network), shallow feature fusion (the feature fusion is only performed at the final output of C3.), decision fusion with the networks jointly trained and with each network separately trained, where the base networks are N13 and N33 for 19 layers, and are N26 and N46 for 50 layers. The comparison is given in Table 3. It can be seen that our approach achieves superior results. There is a slight performance difference when the fusion is conducted before or after ReLU, and our other experiments are reported with fusion before ReLU. It is interesting that the performance of decision fusion with separate training for 19 layers on CIFAR-100 is also good, but the performance on CIFAR-100 for 50 layers is dramatically deteriorated. In contrast, our approach performs similarly for 19 and 50 layers, which shows that our approach indeed can help train a very deep network.

Table 4. Accuracy (%) comparison with different #blocks for deep fusion.

#Blocks	CIFAR-10	CIFAR-100
3 (N13N33)	93.98	72.64
6 (N16N46)	94.08	72.02
8 (N18N58)	94.01	71.92

Performance with different #blocks. We empirically study the performance with different number of blocks. We consider three fused nets, N13N33, N16N46, N18N58, in which the base networks have 3, 6, and 8 blocks respectively, and the deep base network is 19 layers (see Table 2 for more details). The comparison given in Table 4 shows that the performances with different #blocks are close, and over the challenging dataset CIFAR-100, more blocks result in worse performance. Thus, we will use 3 blocks to compare with the baseline networks as the computation complexity is almost the same as the plain network.

5.2 Comparison to Baseline Algorithms

We compare our approach with three baseline networks: plain network, Highway [15] and ResNet [16]. The three baseline networks use the same plain network N1 with 19 layers, and our approach uses the plain network as the deeper base network together with another base network N33 to form the deeply fused net, N13N33. The number of parameters as well as the computation cost of our approach and ResNet are almost the same, as (1) ResNet includes 8 residual connections for the 19-layer network, consisting of two non-identity connections to match the dimensions, and (2) there are three blocks in the shallow network in our approach, and the first one is small and thus computational negligible and the other two are almost the same with the two non-identity connections in ResNet. Highway includes more parameters as it introduces transform gates and hence is computationally more expensive. In addition to the results from our implementation of the plain network, Highway (using batch normalization w/o dropout) and ResNet (batch normalization used before ReLU), we also report the results of Highway and ResNet from the original papers.

The results on CIFAR-10 and CIFAR-100 are shown in Table 5 and Table 6. Compared with Highway that uses the gate to select only a part of the plain network for prediction, our approach uses a small network to help train the whole plain network, resulting in better performance. In comparison to ResNet that uses identity-connection blocks over the layers with the same dimension and non-identity blocks over the layers with different dimensions (appear when only scale changes in our 19-layer network), leading to too many blocks, our approach uses non-identity connection to form the block which is not across scales, and has fewer blocks/fusions, which might be the reason for superior performance. Compared to the second best method, ResNet, our approach achieves more significant gain on CIFAR-100 than CIFAR-10.

In addition, we also report the performance with a much deeper base network, a 50-layer network in Table 5 and Table 6. We have several observations. On the

Table 5. Accuracy (%) comparison on CIFAR-10.

Network	#Layers	#Parameters	Accuracy
Plain	19	$\sim 1.20M$	93.50
Highway (our implementation with dropout)	19	$\sim 2.26M$	93.35
Highway (our implementation without dropout)	19	$\sim 2.26M$	93.06
Highway [15]	19	$\sim 2.3M$	92.46
Highway [15]	32	$\sim 1.25M$	91.20
Resnet (our implementation)	19	$\sim 1.21M$	93.87
Resnet [16]	20	$0.27M$	91.25
Resnet [16]	32	$0.46M$	92.49
Resnet [16]	44	$0.66M$	92.83
Resnet [16]	56	$0.85M$	93.03
Resnet [16]	110	$1.7M$	93.57
FitNet [17]	19	$\sim 2.5M$	91.61
DFN (N13N33)	19	$\sim 1.31M$	**93.98**
Plain	50	$\sim 3.36M$	92.08
DFN (N26N46)	50	$\sim 3.7M$	93.6
DFN (N28N58)	50	$\sim 3.9M$	**93.76**

Table 6. Accuracy (%) comparison on CIFAR-100.

Network	#Layers	#Parameters	Accuracy
Plain	19	$\sim 1.21M$	70.87
Highway (our implementation with dropout)	19	$\sim 2.27M$	68.98
Highway (our implementation without dropout)	19	$\sim 2.27M$	67.97
Highway [15]	19	$\sim 2.3M$	67.76
Resnet (our implementation)	19	$\sim 1.22M$	71.17
FitNet [17]	19	$\sim 2.5M$	64.96
DFN (N13N33)	19	$\sim 1.32M$	**72.64**
Plain	50	$\sim 3.36M$	65.48
DFN (N26N46)	50	$\sim 3.7M$	72.39
DFN (N28N58)	50	$\sim 3.9M$	**72.48**

one hand, the performance from 50 layers for the plain network is lower than that from 19 layers (e.g., decreased to 65.48% from 70.87% on CIFAR-100), while the performances of our approach from 50 layers and 19 layers (72.56% and 72.64%) are only slightly different. On the other hand, on CIFAR-10, our approach with 50 layers performs better than ResNet with similar depth (44 and 56 layers), but ResNet has less parameters, showing that our approach is helpful for training a deep and complex network. Considering that the 50-layer network has much more parameters than the 19-layer network, deep fusion indeed helps train very deep network even with more parameters.

5.3 Comparison to State-of-the-Art Methods

The results of the top-performing algorithms on CIFAR-10 and CIFAR-100 are shown in Table 7. Overall speaking, our approach performs very well under the common data augmentation. There are two competitive algorithms (both published in ICLR 2016), LSUV [12] that focuses on initialization and ELU [28] that focuses on a new nonlinear activation layer, achieving similar performance as our approach. We also report the results of our approach with more base networks (N13N33N43, N13N33N43N63N73): both show better performance on CIFAR-100, while on CIFAR-10 N13N33N43N63N73 does not make improvement.

Table 7. Accuracy (%) comparison to state-of-the-art algorithms.

Common data augmentation		
Algorithm	CIFAR-10	CIFAR-100
DFN (N13N33)	93.98	**72.64**
DFN (N16N46)	**94.08**	72.02
DFN (N13N33N43)	**94.13**	**72.92**
DFN (N13N33N43N63N73)	93.97	**72.99**
HighWay [15] (19 layers) (2015)	92.46	–
HighWay [15] (2015)	92.40	67.67
ResNet [16] (110 layers) (2015)	93.57	–
CMSC [24] (2015)	93.13	72.44
ALL-CNN [30] (2014)	92.75	66.29
LSUV [12] (19 layers) (2015)	93.94	72.34
LSUV [12] (maxout) (2015)	**94.16**	–
GPF [31] (2015)	93.95	67.63
DSN [14] (2015)	92.03	65.43
NiN [32] (2013)	91.19	64.32
Maxout [33] (2013)	90.02	65.46
MIN [34] (2015)	93.25	71.14
DNGO [35] (2015)	93.63	72.60
ELU [28] (2015)	93.45	**75.72**
Extreme data augmentation		
Large ALL-CNN [30] (2014)	95.59	–
Fractional MP [29] (1 test) (2014)	95.50	68.55
Fractional MP [29] (12 tests) (2014)	**96.53**	73.61
SSCNN [36] (2014)	93.72	75.70

Compared to LSUV [12], our approach (N13N33 and N16N46) performs better on CIFAR-100, but worse on CIFAR-10. On CIFAR-10, our approach performs the second best: slightly lower than LSUV [12] using maxout, but greater than LSUV [12] (19 layers) without using the maxout layer. Our approach uses simple basic layers to show the benefit of deep fusion, and we believe that potentially our approach can benefit from other advanced layers, e.g., maxout.

Compared to ELU [28], our approach performs better on CIFAR-10, but worse on CIFAR-100. On CIFAR-100, our approach performs the second best: lower than ELU [28] whose result is very high with common data augmentation. ELU [28] introduces an exponential linear unit, which is complementary to our approach and can be combined with our approach.

It is worth noting that our deeply-fused net outperforms even much larger networks with extreme data augmentation like fractional max-pooling [29].

6 Conclusion

Deep fusion is an approach that fuses not only the final representation but also the intermediate representations of the base networks. It is advantageous in (1) Multi-scale representations can be learnt; (2) The information flow is improved, and training a fused net composed from a very deep base network and a shallow network is less difficult than training the deep base network itself. (3) The deep and shallow networks learning benefit from each other. Experimental results show that our approach achieves superior performance over ResNet and Highway, and competitive performance compared to the state-of-the-arts.

References

1. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States. (2012) 1106–1114
2. Deng, J., Dong, W., Socher, R., Li, L., Li, K., Li, F.: Imagenet: A large-scale hierarchical image database. In: 2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2009), 20-25 June 2009, Miami, Florida, USA. (2009) 248–255
3. Szegedy, C., Ioffe, S., Vanhoucke, V.: Inception-v4, inception-resnet and the impact of residual connections on learning. CoRR abs/1602.07261 (2016)
4. Girshick, R.B., Donahue, J., Darrell, T., Malik, J.: Rich feature hierarchies for accurate object detection and semantic segmentation. In: 2014 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2014, Columbus, OH, USA, June 23-28, 2014. (2014) 580–587
5. Girshick, R.B., Donahue, J., Darrell, T., Malik, J.: Region-based convolutional networks for accurate object detection and segmentation. IEEE Trans. Pattern Anal. Mach. Intell. 38(1) (2016) 142–158
6. Long, J., Shelhamer, E., Darrell, T.: Fully convolutional networks for semantic segmentation. In: IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015. (2015) 3431–3440
7. Xie, S., Tu, Z.: Holistically-nested edge detection. In: 2015 IEEE International Conference on Computer Vision, ICCV 2015, Santiago, Chile, December 7-13, 2015. (2015) 1395–1403
8. Srivastava, N., Hinton, G.E., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: a simple way to prevent neural networks from overfitting. Journal of Machine Learning Research 15(1) (2014) 1929–1958
9. Neyshabur, B., Salakhutdinov, R., Srebro, N.: Path-sgd: Path-normalized optimization in deep neural networks. CoRR abs/1506.02617 (2015)
10. Glorot, X., Bengio, Y.: Understanding the difficulty of training deep feedforward neural networks. In: Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2010, Chia Laguna Resort, Sardinia, Italy, May 13-15, 2010. (2010) 249–256
11. He, K., Zhang, X., Ren, S., Sun, J.: Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In: 2015 IEEE International Conference on Computer Vision, ICCV 2015, Santiago, Chile, December 7-13, 2015. (2015) 1026–1034
12. Mishkin, D., Matas, J.: All you need is a good init. CoRR abs/1511.06422 (2015)
13. Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015. (2015) 448–456
14. Lee, C., Xie, S., Gallagher, P.W., Zhang, Z., Tu, Z.: Deeply-supervised nets. In: Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2015, San Diego, California, USA, May 9-12, 2015. (2015)
15. Srivastava, R.K., Greff, K., Schmidhuber, J.: Training very deep networks. CoRR abs/1507.06228 (2015)

16. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. CoRR abs/1512.03385 (2015)
17. Romero, A., Ballas, N., Kahou, S.E., Chassang, A., Gatta, C., Bengio, Y.: Fitnets: Hints for thin deep nets. CoRR abs/1412.6550 (2014)
18. Chen, T., Goodfellow, I.J., Shlens, J.: Net2net: Accelerating learning via knowledge transfer. CoRR abs/1511.05641 (2015)
19. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. CoRR abs/1409.1556 (2014)
20. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A.: Going deeper with convolutions. In: IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015. (2015) 1–9
21. Ciresan, D.C., Meier, U., Schmidhuber, J.: Multi-column deep neural networks for image classification. In: 2012 IEEE Conference on Computer Vision and Pattern Recognition, Providence, RI, USA, June 16-21, 2012. (2012) 3642–3649
22. Agostinelli, F., Anderson, M.R., Lee, H.: Robust image denoising with multi-column deep neural networks. In: Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States. (2013) 1493–1501
23. Shen, L., Lin, Z., Huang, Q.: Learning deep convolutional neural networks for places2 scene recognition. CoRR abs/1512.05830 (2015)
24. Liao, Z., Carneiro, G.: Competitive multi-scale convolution. CoRR abs/1511.05635 (2015)
25. Krizhevsky, A., Hinton, G.E.: Learning Multiple Layers of Features from Tiny Images. Technical Report 4 (2009)
26. Torralba, A., Fergus, R., Freeman, W.T.: 80 million tiny images: A large data set for nonparametric object and scene recognition. IEEE Trans. Pattern Anal. Mach. Intell. 30(11) (2008) 1958–1970
27. Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., Guadarrama, S., Darrell, T.: CAFFE: Convolutional Architecture for Fast Feature Embedding. (2014)
28. Clevert, D., Unterthiner, T., Hochreiter, S.: Fast and accurate deep network learning by exponential linear units (elus). CoRR abs/1511.07289 (2015)
29. Graham, B.: Fractional max-pooling. CoRR abs/1412.6071 (2014)
30. Springenberg, J.T., Dosovitskiy, A., Brox, T., Riedmiller, M.A.: Striving for simplicity: The all convolutional net. CoRR abs/1412.6806 (2014)
31. Lee, C., Gallagher, P.W., Tu, Z.: Generalizing pooling functions in convolutional neural networks: Mixed, gated, and tree. CoRR abs/1509.08985 (2015)
32. Lin, M., Chen, Q., Yan, S.: Network in network. CoRR abs/1312.4400 (2013)
33. Goodfellow, I.J., Warde-Farley, D., Mirza, M., Courville, A.C., Bengio, Y.: Maxout networks. In: Proceedings of the 30th International Conference on Machine Learning, ICML 2013, Atlanta, GA, USA, 16-21 June 2013. (2013) 1319–1327
34. Chang, J., Chen, Y.: Batch-normalized maxout network in network. CoRR abs/1511.02583 (2015)
35. Snoek, J., Rippel, O., Swersky, K., Kiros, R., Satish, N., Sundaram, N., Patwary, M.M.A., Prabhat, Adams, R.P.: Scalable bayesian optimization using deep neural networks. In: Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015. (2015) 2171–2180
36. Graham, B.: Spatially-sparse convolutional neural networks. CoRR abs/1409.6070 (2014)