DelugeNets: Deep Networks with Efficient and Flexible Cross-layer Information Inflows

Jason Kuen[1] jkuen001@ntu.edu.sg
Xiangfei Kong[1] xfkong@ntu.edu.sg
Gang Wang[2] gangwang6@gmail.com
Yap-Peng Tan[1] eyptan@ntu.edu.sg
Nanyang Technological University[1] Alibaba Group[2]

Abstract

Deluge Networks (DelugeNets) are deep neural networks which efficiently facilitate massive cross-layer information inflows from preceding layers to succeeding layers. The connections between layers in DelugeNets are established through cross-layer depthwise convolutional layers with learnable filters, acting as a flexible yet efficient selection mechanism. DelugeNets can propagate information across many layers with greater flexibility and utilize network parameters more effectively compared to ResNets, whilst being more efficient than DenseNets. Remarkably, a DelugeNet model with just model complexity of 4.31 GigaFLOPs and 20.2M network parameters, achieve classification errors of 3.76% and 19.02% on CIFAR-10 and CIFAR-100 dataset respectively. Moreover, DelugeNet-122 performs competitively to ResNet-200 on ImageNet dataset, despite costing merely half of the computations needed by the latter.

1. Introduction

Deep learning methods [1, 24], particularly convolutional neural networks (CNN) [20] have revolutionized the field of computer vision. CNNs are integral components of many recent computer vision techniques which spread across a diverse range of vision application areas [7]. Hence, developing more sophisticated CNNs has been a prime research focus. Over the years, many variants of CNN architectures have been proposed. Some works focus on improving the activation functions [8, 34], and some focus on increasing the heterogeneity of convolutional filters within the same layers [30, 31]. Lately, the idea of improving CNNs by greatly deepening them has gained much traction, following the immense successes of Residual Networks (ResNets) [9, 10] in image classification.

ResNets make use of residual connections to support relatively unobstructed information flows (shortcut) between layers. Each succeeding layer receives the sum of all its preceding layers[1] outputs as input. Compared to traditional non-residual deep networks, outputs of preceding layers in ResNets can reach succeeding layers with minimal obstructions, even if the preceding layer and succeeding layer is separated by a very long layer-distance. However, the cross-layer connections between preceding and succeeding layers of ResNet are fixed and not "selective", and therefore the succeeding layers are not able to prioritize or deprioritize output channels of certain preceding layers. Instead, the outputs of preceding layers are lumped together via simplistic additive operation, making it very tough for succeeding layers to perform layer-wise information selection. The inflexibility of residual connections also hinders the ability of ResNets to learn cross-layer interactions and correlations.

Densely connected networks (DenseNets) [13] aim to overcome this drawback of ResNets, by having convolutional layers to consider an extra dimension - the depth/cross-layer dimension, in addition to the spatial and feature channel dimensions used in regular convolutions. In DenseNets, the input feature maps to succeeding layers are concatenations of preceding layers outputs, rather than simple summations. Hence, when applying convolution operations on the concatenated feature maps, the convolutional filters have to learn spatial, cross-channel, and cross-layer correlations altogether, entailing heavy amounts of parameters (filter width $\times$ filter height $\times$ # input channels $\times$ # output channels $\times$ # preceding layers) and computations. DenseNet-BC [13] was recently introduced as a more efficient variant of DenseNet, in which the filters have to consider just cross-channel and cross-layer correlations. Despite that, considering that DenseNets' composite layers receive inputs from several dozens of preceding layers, the computation and parameter requirements are still rather high.

To counter excessive computational complexity and parameter growth, DenseNet models are specifically configured to have much lower output *width* (between 12 and 48

[1]The unit layer in ResNet, ResNet-like, and DenseNet models refers to a *composite layer* formed by several basic layers. See Section 3.1.

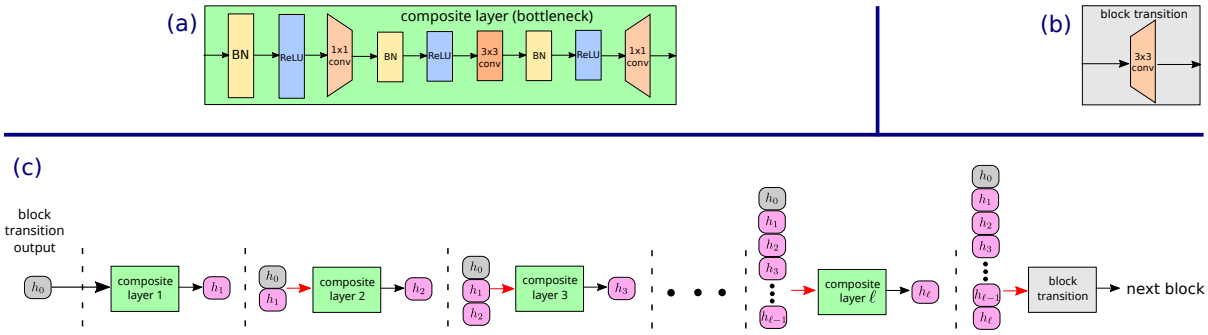

Figure 1. Deluge Network components: (a) a composite layer, (b) a block transition component, and (c) a block. Red-colored arrows indicate 1×1 cross-layer depthwise convolutions.

output channels) at each layer, compared to typical image classification CNNs. However, it is crucial to have considerable network width as contended by [35], and decreasing output width too much is harmful to networks representational power. Furthermore, by visualizing DenseNet's weight norms, Huang et al. [13] showed that the features of preceding *composite layers* get reused directly by the succeeding *composite layers* in a rather infrequent manner. Yet, these "diminished" features have to be processed by relatively expensive convolution operations in DenseNets.

Thus, in this paper, we propose a new class of CNNs called Deluge Networks (DelugeNets) which enable **flexible cross-layer connections** yet have **regular output width** in each composite layer. As a result of using regular output width, the information inflows from preceding layers to succeeding layers in DelugeNets are massive, in contrast to the lesser information inflows in DenseNets. DelugeNets are inspired by separable convolutions [16, 15, 2, 22]. The efficiency of convolutions can be improved by separating the combined dimensions involved, resulting in separable convolutions. DelugeNets are designed such that the depth/cross-layer dimension is processed independently from the rest (channel and spatial dimensions), using a novel variant of convolutional layer called *cross-layer depthwise convolutional layer* (see Figure 2) as described in Section 3.2. Cross-layer depthwise convolutional layers handle only **cross-layer** interactions and correlations, without getting burdened by other dimensions. They facilitate cross-layer connections in DelugeNets in a very efficient and effective manner. Experiments show the superior performances of DelugeNets in terms of classification accuracy, parameter efficiency, and more remarkably computational complexity.

2. Related Work

2.1. Training Deep Networks

Developing methods for training very deep neural networks is a rather significant research topic that has received much attention over the years. Lee et al. [21] incorporate classification losses into intermediate hidden layers, allowing unimpeded supervised signals to reach the layers. In a similar spirit as [21], GoogleNets [30] and Inception [31] models attach auxiliary classifiers to a few intermediate layers to encourage feature discriminativeness in lower network layers. DelugeNets, by contrast, can readily back-propagate the supervised signals to earlier layers without relying on additional losses, due to connections supporting flexible information inflows from preceding to succeeding layers.

There is another stream of works focusing on improving the information flows between layers of very deep networks, which is also the focus of our work. Highway Networks [28, 6] make use of a Long-Short-Term-Memory (LSTM [11])-inspired gating mechanism to control information flow from linear and nonlinear pathways. Through appropriately learned gating functions, information can flow unimpededly across many layers, which can be thought of as a kind of flexible mechanism to combine cross-layer information inflows. He et al. [9] propose Residual Networks (ResNets) which compute the residual (additive) functions of the outputs of linear and nonlinear pathways, without complex gating mechanisms. ResNets have shown to tackle well the vanishing gradient and network degradation problems that occur in very deep networks. The pre-activation variants of ResNet (ResNet-v2) [10] normalize incoming activations at the beginnings of residual blocks to improve information flow and regularization.

Instead of going deeper, Wide-ResNets [35] improve upon originally proposed ResNets by having more convolutional filters/channels (width) and less numbers of layers (depth). Motivated by the high model complexity of ResNets in terms of depths and parameter numbers, several "dropping"-based regularization methods [14, 27] have been developed for regularizing large ResNet models. ResNets can be seen as a less flexible special case of DelugeNets, in which the cross-layer connection weights are not

learnable and fixed as ones. Densely connected networks (DenseNets) [13] which we discuss extensively in Section 1 belong to the same stream of works.

2.2. Separable Convolutions

Separable convolutions have been adopted to construct efficient convolutional networks. Earlier works [15, 4] compress convolutional networks by finding low-rank approximation of convolutional layers of pre-trained networks. Network-in-network [22] employs 1×1 pointwise (cross-channel) convolutional layers to enrich representation learning in an efficient manner. 1×1 pointwise convolutions are generally coupled with other convolution variants (e.g., spatial convolutions) to achieve separable convolutions. Flattened convolutional networks [16] are equipped with one-dimensional convolutional filters of 3 dimensions (channel, horizontal, and vertical) which are processed sequentially and trained from scratch. For maximal channel-spatial separability, a conventional convolutional layer can be replaced with depthwise separable convolution (spatial depthwise convolution followed by a 1×1 pointwise convolution), as demonstrated by Xception [2]. In contrast to these existing works which mainly deal with channel-spatial separability, the work in this paper deals with "cross-layer"-channel separability. Also, to the best of our knowledge, this paper is the first work on cross-layer depth/channelwise convolutions.

3. Deluge Networks

Similar to existing CNNs (ResNet [9, 10], VGGNet [26], and AlexNet [18]), DelugeNets gradually decrease spatial sizes and increase feature channels of feature maps from bottom to top layers, with a linear classification layer attached to the end. The layers operating on the same feature map dimensions can be grouped to form a *block*. In DelugeNets, the input to a particular layer comes from **all of its preceding layers** of the same *block*. There is no information directly flowing from other *blocks*. Within any *block*, the cross-layer information flows through connections established by the cross-layer depthwise convolutions (see Section 3.2). For transition to the next *block* as described in Section 3.3, we perform cross-layer depthwise convolution followed by strided spatial convolution to obtain feature map of matching dimensions. The structure of *block* in DelugeNets is illustrated in Figure 1(c), with individual layers separated by vertical dashed lines.

3.1. Composite Layer

In CNNs, a *layer* often refers to a *composite layer* of several basic layers such as Rectified Linear Unit (ReLU), Convolutional (Conv), Batch Normalization (BN) layers. Inspired by [10], we use the bottleneck-kind of *composite layer* BN-ReLU-Conv-BN-ReLU-Conv-BN-ReLU-Conv in DelugeNets, as illustrated in Figure 1(a). This kind of *composite layer* is designed to improve parameter efficiency in deep networks, by employing 1×1 spatial convolutional layers at the beginning to reduce channel dimension, and at the end to increase channel dimension. In the ResNet models proposed by [10], the base channel dimensions are increased by 4 times. We however only increase them by 2 times in this paper, for the reason that we can allocate more computational and parameter budgets to train deeper DelugeNets.

Such a *composite layer* has also shown to work very well for very deep neural networks which combine information from multiple sources, such as ResNets and the proposed DelugeNets. The primary reason that it works well is that combined multi-source information is normalized via BN layers before it is passed into the upcoming weight (convolutional) layers. This reduces internal covariate shift and regularizes the model more effectively [10] than just passing unnormalized multi-source information to the weight layers.

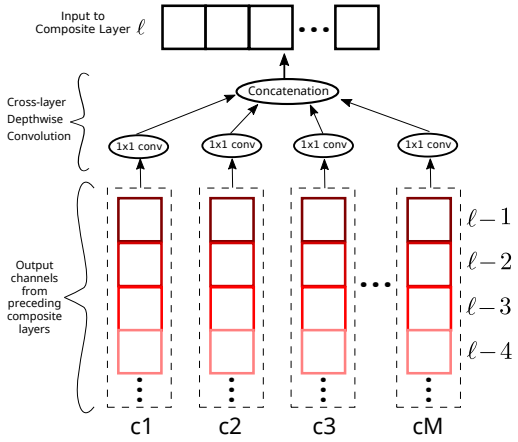

Figure 2. Cross-layer Depthwise Convolution. The columns correspond to feature channel indices, and the rows correspond to preceding *composite layer* indices.

3.2. Cross-layer Depthwise Convolutional Layers

To facilitate efficient and flexible cross-layer information inflows, in this paper, we develop a cross-layer depthwise convolution method. A cross-layer depthwise convolutional layer concatenates the channels of feature map outputs of many layers, and then applies (channel,spatial)-independent filters to the concatenated channels. Equipped with such filters, DelugeNets are able to process the depth/layer dimension independently of the rest (channel and spatial dimensions), as mentioned in Section 1. Figure 2 gives a graphical illustration of cross-layer depthwise convolution operation.

Cross-layer depthwise convolutional layers facilitate the inflows of information from preceding *composite layers* to succeeding *composite layers*. Suppose that ℓ denotes the

layer of an arbitrary *composite layer*, and $h^c_{\ell-i} \in \mathbb{R}$ denotes the c-th channel of the preceding $(\ell-i)$-th *composite layer*'s output. And, there are N number of preceding *composite layers*, as well as one preceding block transition output $\boldsymbol{h}_0$. The c-th channel of the input $\boldsymbol{x}_\ell$ to *composite layer* ℓ is:

$$x^c_\ell = \sum_{i=1}^{N+1} w^c_{\ell-i} h^c_{\ell-i} + b^c_\ell \tag{1}$$

where $w^c_{\ell-i} \in \mathbb{R}$ and $b^c_\ell \in \mathbb{R}$ are the filter weights and bias respectively, for each channel. We streamline the equations by not having spatial location-related notations, and the weights and biases are assumed to be shared across all 1×1 spatial locations (spatially independent) in the input feature maps as mentioned earlier.

The parameter cost of adding cross-layer depthwise convolutional layer to any existing network architecture is relatively low compared to other network parameters. For an arbitrary *composite layer* in the network, the number of additional parameters is merely $N \times M + 1$, where M is the number of feature channels. Experimentally, we find that these extra parameters on average make up about 3% of entire model parameters. In terms of computational complexity (measured in floating-point operation numbers or FLOPs), cross-layer depthwise convolutions on average cost 3% more, compared to baseline models without such convolutions. DenseNets [13], on the other hand, require heavy amounts of computations and parameters to connect to preceding layers, through cross-layer output concatenations followed by 3×3 or 1×1 spatial convolutions.

Advantages: Cross-layer depthwise convolutional layers are beneficial because they encourage features generated by a preceding *composite layers* to be taken as inputs for many times by the succeeding layers (feature reuse). This naturally leads to parameter efficiency because there is no need to redundantly learn filters which generate the same features in succeeding layers, in case those features are needed again later. Furthermore, in conventional ReLU-based convolutional networks, features that get turned off by ReLU activation functions (at the beginnings of blocks) cannot be recovered by other network parts or layers. In DelugeNets, via the use of cross-layer depthwise convolutional layers, output of a preceding *composite layer* can be transformed differently for each succeeding *composite layer* to serve as input. Consequently, input features that get turned off at the beginning of certain succeeding *composite layers* may be active in others.

In CNNs, the filter weights are shared across many spatial locations in the feature maps. The weight sharing mechanism acts as an effective regularizer. Similar to the CNN's weight sharing mechanism, the same features in DelugeNets' preceding *composite layers* are shared by the succeeding *composite layers*. As a result, the weights of composite layers in DelugeNets become more regularized. Based on such consideration, we allocate more model parameters to the spatially smaller network blocks, by setting the number of *composite layer* in succeeding block to be larger than preceding block. The motivation behind this is to achieve **lower computational complexity** (since smaller feature maps are computationally cheaper to process), relying more on cross-layer feature reuse and less on parameter-sharing across spatial locations, for regularization. Such an allocation scheme differs from ResNets in which many layers and parameters are allocated for blocks with ge feature maps to regularize filters better, and less for blocks with spatially small feature maps to reduce overfitting (see Section 4.2).

Besides encouraging feature reuse, cross-layer depthwise convolutional layers are advantageous from the perspective of gradient flow. The gradient flows in DelugeNets are regulated by multiplicative interactions with the filter weights in cross-layer depthwise convolutional layers, such that the *composite layers* all receive unique backpropagated gradient signals even if they come from the same block. This is not true for ResNet models, in which the *composite layers* within the same block receive identical backpropagated gradient signals, due to simple addition (residual) operation.

3.3. Block Transition

Different network blocks operate on feature maps of different spatial and channel dimensions. For block transition, there is a need to transform the feature map to match the spatial and channel dimensions of next block. In ResNet-like models, block transition can be done with either 1×1 strided convolution, or strided average pooling with channel padding. These block transition designs aim to preserve the information from previous block by having only minimal transformation as well as dismissing any non-linear activation function. Such block transition designs are suboptimal for DelugeNets because they allow direct information flow from just the last *composite layer* of the previous block, and they conceivably hinder the information flows from other *composite layers*.

To this end, we propose a new block transition component which has a cross-layer depthwise convolutional layer followed by 3×3 spatial convolutional layer. The cross-layer depthwise convolutional layer allows direct information inflow from all *composite layers* from the previous block, therefore summarizing the outputs of all *composite layers* of previous block. Then, the 3×3 strided spatial convolutional layer (see Figure 1(b)) transforms the summarized feature map to have matching spatial and channel dimensions. 3×3 strided convolutional layer is chosen over 1×1 strided convolutional layer as the latter wastes the features it receives, for many of the feature map's spatial

Model	#Params	Depth	GigaFLOPs	CIFAR-10	CIFAR-100
Highway Network [28]	-	-	-	7.60	32.24
FractalNet [19]	38.6M	20	-	4.59	22.85
ResNet [9]	1.7M	164	-	5.93	25.16
ResNet [9]	10.2M	1001	-	7.61	27.82
ResNet with ELU [25]	-	110	-	5.62	26.55
ResNet with Identity Mappings [10]	1.7M	164	-	5.46	24.33
ResNet with Identity Mappings [10]	10.2M	1001	-	4.62	22.71
ResNet with Swapout [27]	7.4M	32	-	4.76	22.72
ResNet with Stochastic Depth [14]	1.7M	32	-	5.23	24.98
ResNet with Stochastic Depth [14]	10.2M	1202	-	4.91	-
Wide-ResNet (04×width) [35]	8.7M	40	2.60	4.53	21.18
Wide-ResNet (08×width) [35]	11.0M	16	3.10	4.27	20.43
Wide-ResNet (10×width) [35]	36.5M	28	10.49	4.00	19.25
DenseNet ($k = 12$) [13]	7.0M	100	3.65	4.10	20.20
DenseNet ($k = 24$) [13]	27.2M	100	14.56	3.74	19.25
DenseNet-BC ($k = 24$) [13]	15.3M	250	10.09	3.62	17.60
DenseNet-BC ($k = 40$) [13]	25.6M	190	18.67	**3.46**	**17.18**
DelugeNet-146	6.7M	146	1.43	3.98	19.72
DelugeNet-218	10.0M	218	2.13	3.88	19.31
Wide-DelugeNet-146	20.2M	146	4.31	3.76	19.02

Table 1. CIFAR-10 and CIFAR-100 test errors (percentage) of existing models and DelugeNets.

locations, while the former does not. Similar to the block transition designs in ResNets, we do not add non-linear activation functions after the spatial convolutional layer.

4. Experiments

To rigorously validate the effectiveness of DelugeNets, we evaluate DelugeNets on 3 image classification datasets with varied degrees of challengingness: CIFAR-10 [17], CIFAR-100 [17], ImageNet [23]. The experimental code is written in Torch [3], and is available at https://github.com/xternalz/DelugeNets.

4.1. CIFAR-10 and CIFAR-100

Datasets: CIFAR-10 and CIFAR-100 are 2 subsets of the Tiny Images dataset [32] annotated to serve as image classification datasets. There are 50,000 training images and 10,000 testing images for each of the 2 CIFAR datasets. For pre-processing, we subtract channel-wise means from the images, and divide them by channel-wise standard deviations. During training, data augmentation is carried out moderately as in [35, 13], with horizontal flipping and random crops taken from images padded by 4 pixels on each side. For all CIFAR-based models, the training is carried out using a single GPU.

Implementation: A total of 3 different DelugeNet models are implemented and evaluated on CIFAR datasets. Similar to [10, 35], all the 3 DelugeNet models have 3 *blocks* - the first block works on spatially 32×32 feature maps, followed by 16×16 and 8×8 feature maps for second and third blocks respectively. They vary only in terms of numbers of *composite layers* and feature channel dimensions for the 3 blocks. To minimize manual tuning of architectural hyper-parameters, we design different DelugeNet models based on a simple principle that follows the parameter allocation scheme mentioned in Section 3.2 - the second block has 2 times the **numbers of *composite layers*** and **feature channel dimension** (width) of the first block, the third block has 2 times of the second's, and so on:

DelugeNet-146 has base feature channel dimensions (widths) of {32,64,128}, and *composite layer* counts of {8,16,24}, for its 3 blocks (in sequential ordering) respectively.

DelugeNet-218 shares the same base widths as DelugeNet-146, but it comes with larger *composite layer* counts of {12,24,36} which make it a much deeper model.

Wide-DelugeNet-146 is a $1.75\times$ wider variant of DelugeNet-146, having base widths of {56,112,224}, while the *composite layer* counts remain the same.

The proposed models (DelugeNet-146, DelugeNet-218, and Wide-DelugeNet-146) for the 2 CIFAR datasets differ only in the numbers of output labels (10 and 100). To train the models, we run Stochastic Gradient Descent (SGD) over a total of 300 training epochs, with Nestorov Momentum [29] and weight decay rate of $1e{-}4$. As most of the existing models we compare with in this paper do not use any dropout-like regularization, we do not use any either, for fairer comparison. The starting learning rate is 0.1, and it is decayed by factor of 0.1 at epoch 150 and 225. We set the mini-batch size as 64. All DelugeNet model parameters are initialized using He's initialization method [8], and they

are trained using the same settings. The training settings are in fact identical to the settings employed in [13] to train DenseNets.

Results: The top-1 classification errors achieved by the DelugeNets and existing models on both CIFAR datasets are presented in Table 1. The results of existing models are obtained directly from their respective papers. As shown in Table 1, DelugeNets can benefit from "deepening" (DelugeNet-218) and "widening" (Wide-DelugeNet-146).

Parameter efficiency: DelugeNets are able to perform well despite requiring much lower numbers of learnable parameters compared to existing models. The parameter efficiencies of Delugenets are second only to DenseNet-BCs [13] which aggressively compress and reduce feature channels to save parameters. Notably, DelugeNet-218 performs competitively to Wide-ResNet ($10\times$width), on both CIFAR-10 and CIFAR-100 datasets, with merely 10M parameters compared to 36.5M parameters in Wide-ResNet. Besides, Wide-DelugeNet-146 achieves CIFAR classification errors comparable to those of DenseNet ($k = 24$), with 7M fewer parameters.

Computational complexity: In addition to parameter numbers, we report the model complexities of DelugeNets and several other comparable models (Wide-ResNets, DenseNets, and DenseNet-BCs), in terms of floating-point operation (FLOP in giga prefix unit, Giga/GFLOP) numbers. We find that in overall DelugeNets have significantly fewer model complexities than other models. Surprisingly, DelugeNet-218 requires just $1/5$ of the FLOPs required by Wide-ResNet ($10\times$width) to achieve similar classification errors. Although DelugeNets cannot exactly match or outperform DenseNet-BC, they (DelugeNets) can achieve appreciable classification errors which are rather close to those of DenseNet-BCs, at fractions of DenseNet-BCs' complexity costs. The lower model complexities of DelugeNets are attributed to the parameter allocation scheme (Section 3.2) as well as the capability of cross-layer depthwise convolutions at alleviating overfitting, even when spatially smaller network blocks have more parameters/layers than their spatially larger counterparts.

Ablation study: In this work, we propose cross-layer depthwise convolutional layer and a new kind of block transition design with 3×3 spatial convolution, which differentiate DelugeNets from existing networks. To better understand the contributions of these components, we construct ResNet-like baselines on the 3 proposed DelugeNet models. There are 2 types of baselines for each of the DelugeNet models: The **first baseline** has all of its cross-layer depthwise convolutions replaced by residual connections. Alternatively, the residual connections can be seen as cross-layer depthwise convolutional layers, whose weights are fixed as ones as pointed in Section 2. The **second baseline** is similar to the first one except that it is equipped with 3×3 convolutional shortcuts for block transitions, similar to our proposed block transition design. Other than those mentioned, all aspects of the baselines and their corresponding DelugeNets are the same, including training settings. We evaluate the baselines on CIFAR-100. The results are shown in Table 2.

Model	#Params	GFLOPs	Error	PDiff
ResNet-like baseline				
- 1x1 conv shortcut	6.15M	1.33	21.14	-
- 3x3 conv shortcut	6.50M	1.39	20.84	+0.30
DelugeNet-146	6.69M	1.43	**19.72**	+1.12
ResNet-like baseline				
- 1x1 conv shortcut	9.26M	1.98	20.78	-
- 3x3 conv shortcut	9.55M	2.05	20.31	+0.47
DelugeNet-218	10.00M	2.13	**19.31**	+1.00
ResNet-like baseline				
- 1x1 conv shortcut	18.76M	4.05	19.79	-
- 3x3 conv shortcut	19.82M	4.25	19.98	-0.19
Wide-DelugeNet-146	20.19M	4.31	**19.02**	+0.77

Table 2. Comparison with ResNet-like baselines on CIFAR-100 test errors. The fourth column reports the performance differences (**PDiff**) between baselines and DelugeNets.

Block transitions with 3×3 convolutional shortcuts can mildly improve the performances of DelugeNet-146's and DelugeNet-218's baselines. However, there is slight overfitting ($19.79\% \rightarrow 19.98\%$) from adding 3×3 convolutional shortcuts to Wide-DelugeNet-146's baseline. The overfitting issue is greatly eased by having cross-layer depthwise convolutions in Wide-DelugeNet-146. As evidenced by the significant performance improvements (about 1%) of DelugeNets over the baselines, the biggest contributor is cross-layer depthwise convolutional layer. Yet, the parameter costs incurred by adding these layers are very marginal. The smallest DelugeNet model, DelugeNet-146 (19.72%) with just 6.69M parameters and complexity of 1.43 GFLOPs, suprisingly outperforms the biggest ResNet-like baseline (19.79%) with 18.76M parameters and complexity of 4.05 GFLOPs. Furthermore, just tiny increases in complexity (about 3%) are needed by cross-layer depthwise convolutions to achieve considerable performance gains. These findings reaffirm the advantages of the proposed cross-layer depthwise convolutional layer for deep networks.

Cross-layer connectivity: For better understanding of cross-layer depthwise convolutional layers, we compute layer-wise L2-norms of the cross-layer depthwise convolutional filter weights of DelugeNet-218, on CIFAR-10 and CIFAR-100. We provide visualizations in Figure 3. The weight's L2-norms are normalized[2] by dividing them with the maximum layer-wise L2-norms of every block. We consider only cross-layer depthwise convolutional layers in the **Block Transition 1** (from Block 1 to Block 2), **Block Tran-**

[2]The relative (as opposed to absolute) L2-norm values are sufficient, since BN layers follow cross-layer depthwise convolutional layers.

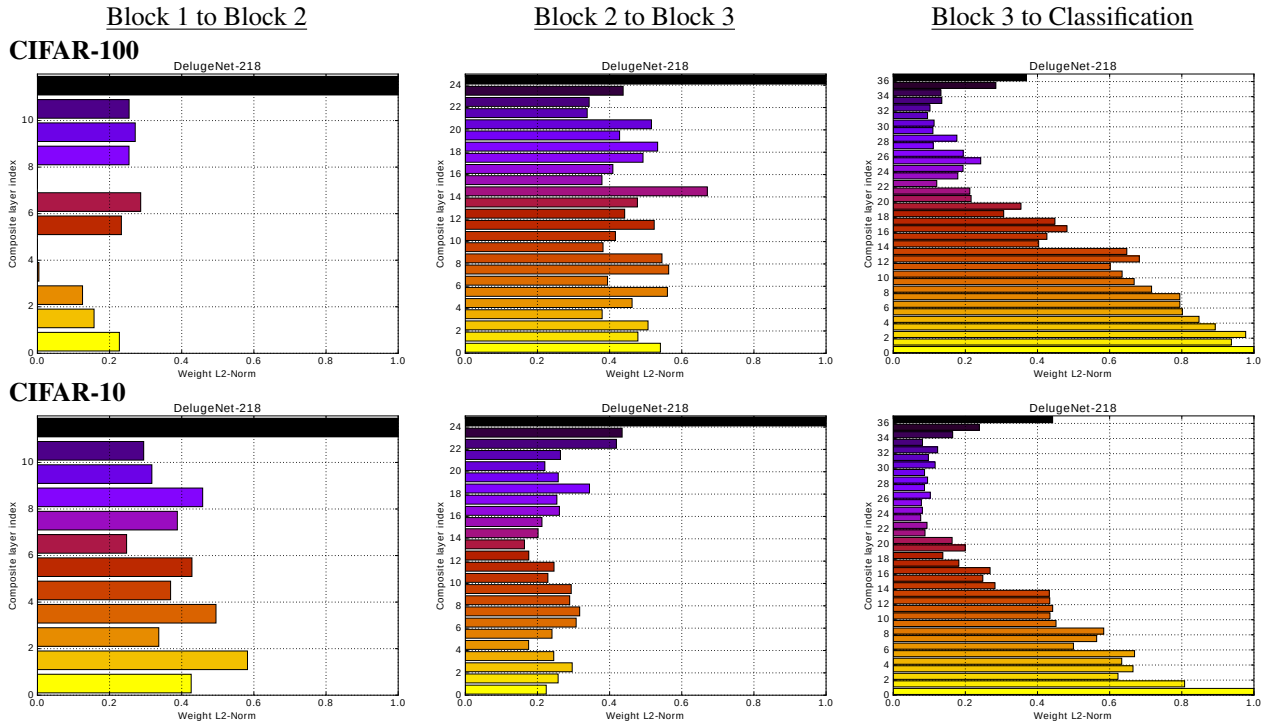

Figure 3. Layer-wise L2-norms of cross-layer depthwise convolution weights. Each of the 3 columns corresponds to a different block transition stage in the networks. Vertical axes indicate the indices of the preceding *composite layers*, and horizontal axes indicate normalized L2-norm values. The longer the horizontal bar of a *composite layer*, the larger its contribution.

sition 2 (from Block 2 to Block 3), and the cross-layer depthwise convolutional layer (from Block 3 to classification) **before classification layer**. These are the cross-layer depthwise convolutional layers with the highest numbers cross-layer connections in the networks.

Generally, all of the preceding *composite layers* contribute reasonably, with a few dominating. The weights (initialized uniformly) are no longer uniform for all layers in the trained models, being different from the connection rigidity exhibited by ResNets. For first and second block transitions, the last *composite layers* always contribute the most, somehow approximating the behaviors of conventional neural networks where all incoming information comes solely from the layer just before the current layer. On the other hand, for the cross-layer depthwise convolutional layer (before classification layer) connected to the third network block, the early *composite layers* generally contribute the most, and the final *composite layer* contributes moderately. We reckon that the features computed by the earlier *composite layers* are fairly ready for classification, and the subsequent *composite layers* just refine them further. Such phenomenon has also been observed in ResNets [33], where upper layers could be deleted without hurting performance much. In addition, we notice that some *composite layers* in the first block of DelugeNet-218 (CIFAR-100) hardly have any contributions to Block Transition 1. This observation may suggest that *layer sparsity* can be potentially exploited for training DelugeNets.

4.2. ImageNet

Dataset: ImageNet (1000 classes) dataset [23] is the most widely used large-scale image classification dataset in recent years. We report the results for validation images. We follow the data augmentation scheme adopted in GoogleNet/Inception [30, 31] and ResNet-v2 [10] with the following augmentation techniques: scale [18] & aspect ratio [30] augmentation, PCA-based lighting augmentation [18], photometric distortions [12], and horizontal flipping. The images are normalized by subtracting them from channel-wise means and dividing them by channel-wise standard deviations.

Implementation: We implement and evaluate 3 different DelugeNet models on ImageNet dataset. Similar to ResNet models [10], before being passed to the first block, the feature map (after first layer) is downsampled to spatial dimensions of 56×56 via max-pooling. We set the base feature channel dimensions (widths) of all ImageNet-based DelugeNet models to be identical to those of ResNets [10] - $\{64,128,256,512\}$. Most of the network architectural details follow ResNets' closely, and they are not necessarily optimal for DelugeNets. Moreover, we emphasize on great simplicity when choosing the *composite layer* counts for DelugeNets, setting the number of *composite layers* in each block to be larger than or equal to that of its preceeding

block. This is in contrast to the carefully tuned *composite layer* counts [10] (e.g., {3,4,23,3}, {3,8,36,3}) in ResNets. The specifications of DelugeNet models are as follows:

DelugeNet-92 has *composite layer* counts of {7,7,8,8}, for its 4 blocks (in sequential ordering) respectively. **DelugeNet-104** and **DelugeNet-122** are two deeper DelugeNet models, with *composite layer* counts of {7,8,9,10} and {7,9,11,13} respectively.

The ImageNet-based models are initialized similarly to the CIFAR models. Training is carried out with SGD over a total of 100 training epochs, with Nestorov Momentum [29] and weight decay rate of $1e{-}4$. We start with learning rate of 0.1, and decay it by factor of 0.1 at the end of every thirty epoch. The training mini-batch size is 256. In view of large model and image sizes, we train the models in multi-GPU mode with 8 GPUs, splitting each mini-batch into 8 portions. These are standard training settings and similar to those [5] used to train ImageNet-based ResNets.

Results: The top-1 and top-5 classification errors achieved by DelugeNets on ImageNet validation dataset are presented in Table 3, along with the numbers of floating-point operations (GigaFLOPs/GFLOPs) required by the models to process one image. For comparison, we include the results of ResNet-v2 [10], Wide-ResNet [35], and DenseNet [13].

DelugeNet-92 with merely 43.4M parameters outperform ResNet-101 (top1 +0.39%, top5 +0.18%) and even ResNet-152 (top1 +0.11%, top5 +0.13%). Besides, with 25.5M less parameters and about half (11.8 GFLOPs) of the Wide-ResNet-50's FLOPs (22.8 GFLOPs), DelugeNet-92 performs comparably to Wide-ResNet-50. Both deeper models DelugeNet-104 and DelugeNet-122 further push down the classification errors substantially. Remarkably, DelugeNet-122 attains classification errors comparable to ResNet-200's, despite needing just about half (15.2 GFLOPs) of the computations required by ResNet-200 (30.1 GFLOPs). With the flexible cross-layer connections established by cross-layer depthwise convolutions, DelugeNet-122 is robust against the overfitting issue caused by allocating more parameters to the spatially smaller blocks. Moreover, DelugeNet-122 outperforms DenseNet-161 (best DenseNet model reported for ImageNet dataset) given similar model complexities.

Given similar or considerably lower model budgets (GFLOPs, number of parameters), DelugeNets are able to surpass ResNets, although DelugeNets' *composite layer* counts are configured in a rather simple manner.

5. Memory Usage

In ResNets, the residual (addition) operation allows memory buffers to be shared or reused across consecutive composite layers. However, for DelugeNets and DenseNets [13], the output activations and gradients of the last convolutional layer of every composite layer have to be retained persistently during training. For instance, when doing training ($\hat{\mathbf{T}}$) and inference ($\hat{\mathbf{I}}$) with Wide-DelugeNet-146 on CIFAR-100 (batch size of 32), the occupied GPU memory is roughly {$\hat{\mathbf{T}}$: 2.8G, $\hat{\mathbf{I}}$: 0.61G}, while its ResNet-baseline counterpart only requires {$\hat{\mathbf{T}}$: 1.3G, $\hat{\mathbf{I}}$: 0.44G}. The gap is smaller during inference ($1.5\times$) than in training ($2.2\times$). On the other hand, DenseNet($k = 24$), DenseNet-BC($k = 24$), and DenseNet-BC($k = 40$) require {$\hat{\mathbf{T}}$: 6.3G, $\hat{\mathbf{I}}$: 0.87G}, {$\hat{\mathbf{T}}$: 8.6G, $\hat{\mathbf{I}}$:0.75G}, and {$\hat{\mathbf{T}}$: 8.4G, $\hat{\mathbf{I}}$: 1.1G} respectively. Wide-DelugeNet-146 is more memory-efficient than DenseNet($k = 24$), while DenseNet-BCs are very memory-costly.

Model	#Params	GFLOPs	Top-1	Top-5
ResNet-101 [10]	44.6M	15.7	22.44	6.21
ResNet-152 [10]	60.3M	23.1	22.16	6.16
ResNet-200 [10]	64.8M	30.1	21.66	**5.79**
Wide-ResNet-50 [35]	68.9M	22.8	21.9	6.03
DenseNet-161 [13]	28.7M	15.5	22.2	-
DelugeNet-92	43.4M	11.8	22.05	6.03
DelugeNet-104	51.4M	13.2	21.86	5.98
DelugeNet-122	63.6M	15.2	**21.53**	5.86

Table 3. ImageNet validation errors (single 224×224 crops).

6. Conclusion

We extend depthwise convolutional layers to *cross-layer depthwise convolutional layers*, which facilitate cross-layer connections in our proposed DelugeNets. The cross-layer information inflows in DelugeNets are **flexible** (cross-layer depthwise convolutional filters are learned) yet **massive** (output widths of composite layers are regular). Experiments indicate that DelugeNets are quite comparable to state-of-the-art models in terms of accuracies, and yet DelugeNets have lower model complexities. This suggests that DelugeNets may have potentials in energy-efficient deep learning. In future, we would like to investigate regularization techniques (e.g., layer dropout [14]) in the context of cross-layer connectivity, as well as applying DelugeNets to other vision applications.

7. Acknowledgement

This research was carried out at the Rapid-Rich Object Search (ROSE) Lab at Nanyang Technological University (NTU), Singapore. ROSE Lab is supported by the National Research Foundation, Singapore, under its Interactive Digital Media (IDM) Strategic Research Programme. We gratefully acknowledge the GPU resources and support provided by NVAITC (NVIDIA AI Technology Centre) Singapore.

References

[1] Y. Bengio. Learning deep architectures for ai. *Foundations and Trends in Machine Learning*, 2(1):1–127, 2009. 1

[2] F. Chollet. Xception: Deep learning with depthwise separable convolutions. *arXiv preprint arXiv:1610.02357*, 2016. 2, 3

[3] R. Collobert, K. Kavukcuoglu, and C. Farabet. Torch7: A matlab-like environment for machine learning. In *BigLearn, NIPS Workshop*, number EPFL-CONF-192376, 2011. 5

[4] E. L. Denton, W. Zaremba, J. Bruna, Y. LeCun, and R. Fergus. Exploiting linear structure within convolutional networks for efficient evaluation. In *NIPS*, 2014. 3

[5] Facebook. Resnet training in torch. https://github.com/facebook/fb.resnet.torch, 2016. 8

[6] K. Greff, R. K. Srivastava, and J. Schmidhuber. Highway and residual networks learn unrolled iterative estimation. In *ICLR*, 2017. 2

[7] J. Gu, Z. Wang, J. Kuen, L. Ma, A. Shahroudy, B. Shuai, T. Liu, X. Wang, and G. Wang. Recent advances in convolutional neural networks. *arXiv preprint arXiv:1512.07108*, 2015. 1

[8] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *CVPR*, pages 1026–1034, 2015. 1, 5

[9] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, June 2016. 1, 2, 3, 5

[10] K. He, X. Zhang, S. Ren, and J. Sun. Identity mappings in deep residual networks. In *ECCV*, 2016. 1, 2, 3, 5, 7, 8

[11] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997. 2

[12] A. G. Howard. Some improvements on deep convolutional neural network based image classification. *arXiv preprint arXiv:1312.5402*, 2013. 7

[13] G. Huang, Z. Liu, and K. Q. Weinberger. Densely connected convolutional networks. *arXiv preprint arXiv:1608.06993*, 2016. 1, 2, 3, 4, 5, 6, 8

[14] G. Huang, Y. Sun, Z. Liu, D. Sedra, and K. Weinberger. Deep networks with stochastic depth. In *ECCV*, 2016. 2, 5, 8

[15] M. Jaderberg, A. Vedaldi, and A. Zisserman. Speeding up convolutional neural networks with low rank expansions. In *BMVC*, 2014. 2, 3

[16] J. Jin, A. Dundar, and E. Culurciello. Flattened convolutional neural networks for feedforward acceleration. In *ICLR*, 2015. 2, 3

[17] A. Krizhevsky. Learning multiple layers of features from tiny images. Technical report, 2009. 5

[18] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, pages 1097–1105, 2012. 3, 7

[19] G. Larsson, M. Maire, and G. Shakhnarovich. Fractalnet: Ultra-deep neural networks without residuals. *arXiv preprint arXiv:1605.07648*, 2016. 5

[20] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. 1

[21] C.-Y. Lee, S. Xie, P. Gallagher, Z. Zhang, and Z. Tu. Deeply-supervised nets. In *AISTATS*, volume 2, page 6, 2015. 2

[22] M. Lin, Q. Chen, and S. Yan. Network in network. In *ICLR*, 2014. 2, 3

[23] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. Imagenet large scale visual recognition challenge. *IJCV*, 115(3):211–252, 2015. 5, 7

[24] J. Schmidhuber. Deep learning in neural networks: An overview. *Neural Networks*, 61:85 – 117, 2015. 1

[25] A. Shah, E. Kadam, H. Shah, S. Shinde, and S. Shingade. Deep residual networks with exponential linear unit. In *VisionNet*, pages 59–65, 2016. 5

[26] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015. 3

[27] S. Singh, D. Hoiem, and D. Forsyth. Swapout: Learning an ensemble of deep architectures. In *NIPS*. 2016. 2, 5

[28] R. K. Srivastava, K. Greff, and J. Schmidhuber. Training very deep networks. In *NIPS*. 2015. 2, 5

[29] I. Sutskever, J. Martens, G. E. Dahl, and G. E. Hinton. On the importance of initialization and momentum in deep learning. In *ICML*, 2013. 5, 8

[30] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *CVPR*, pages 1–9, 2015. 1, 2, 7

[31] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. In *CVPR*, 2016. 1, 2, 7

[32] A. Torralba, R. Fergus, and W. T. Freeman. 80 million tiny images: A large data set for nonparametric object and scene recognition. *IEEE TPAMI*, 30(11):1958–1970, Nov. 2008. 5

[33] A. Veit, M. Wilber, and S. Belongie. Residual networks are exponential ensembles of relatively shallow networks. In *NIPS*, 2016. 7

[34] B. Xu, N. Wang, T. Chen, and M. Li. Empirical evaluation of rectified activations in convolutional network. In *ICML Deep Learning Workshop*, 2015. 1

[35] S. Zagoruyko and N. Komodakis. Wide residual networks. In *BMVC*, 2016. 2, 5, 8