Sparsely Aggregated Convolutional Networks

Ligeng Zhu[1(✉)], Ruizhi Deng[1], Michael Maire[2], Zhiwei Deng[1], Greg Mori[1], and Ping Tan[1]

[1] Simon Fraser University, Burnaby, Canada
{lykenz,ruizhid,zhiweid,mori,pingtan}@sfu.edu

[2] University of Chicago, Chicago, USA
mmaire@uchicago.edu

Abstract. We explore a key architectural aspect of deep convolutional neural networks: the pattern of internal skip connections used to aggregate outputs of earlier layers for consumption by deeper layers. Such aggregation is critical to facilitate training of very deep networks in an end-to-end manner. This is a primary reason for the widespread adoption of residual networks, which aggregate outputs via cumulative summation. While subsequent works investigate alternative aggregation operations (*e.g.* concatenation), we focus on an orthogonal question: which outputs to aggregate at a particular point in the network. We propose a new internal connection structure which aggregates only a sparse set of previous outputs at any given depth. Our experiments demonstrate this simple design change offers superior performance with fewer parameters and lower computational requirements. Moreover, we show that sparse aggregation allows networks to scale more robustly to 1000+ layers, thereby opening future avenues for training long-running visual processes.

1 Introduction

As convolutional neural networks have become a central component of many vision systems, the field has quickly adopted successive improvements in their basic design. This is exemplified by a series of popular CNN architectures, most notably: AlexNet [25], VGG [32], Inception [34,35], ResNet [16,17], and DenseNet [20]. Though initially targeted to image classification, each of these designs also serves the role of a backbone across a broader range of vision tasks, including object detection [7,14] and semantic segmentation [3,28,41]. Advances in backbone network architecture consistently translate into corresponding performance boosts to these downstream tasks.

We examine a core design element, internal aggregation links, of the recent residual (ResNet [16]) and dense (DenseNet [20]) network architectures. Though vital to the success of these architectures, we demonstrate that the specific

Electronic supplementary material The online version of this chapter (https://doi.org/10.1007/978-3-030-01258-8_12) contains supplementary material, which is available to authorized users.

V. Ferrari et al. (Eds.): ECCV 2018, LNCS 11216, pp. 192–208, 2018.
https://doi.org/10.1007/978-3-030-01258-8_12

structure of aggregation in current networks is at a suboptimal design point. DenseNet, considered state-of-the-art, actually wastes capacity by allocating too many parameters and too much computation along internal aggregation links.

We suggest a principled alternative design for internal aggregation structure, applicable to both ResNets and DenseNets. Our design is a sparsification of the default aggregation structure. In both ResNet and DenseNet, the input to a particular layer is formed by aggregating the output of all previous layers. We switch from this full aggregation topology to one in which only a subset of previous outputs are linked into a subsequent layer. By changing the number of incoming links to be logarithmic, rather than linear, in the overall depth of the network, we fundamentally reduce the growth of parameters in our resulting analogue of DenseNet. Experiments reveal our design to be uniformly advantageous:

- On standard tasks, such as image classification, SparseNet, our sparsified DenseNet variant, is more efficient than both ResNet and DenseNet. This holds for measuring efficiency in terms of both parameters and operations (FLOPs) required for a given level of accuracy. A much smaller SparseNet model matches the performance of the highest accuracy DenseNet.
- In comparison to DenseNet, the SparseNet design scales in a robust manner to instantiation of extremely deep networks of 1000 layers and beyond. Such configurations magnify the efficiency gap between DenseNet and SparseNet.
- Our aggregation pattern is equally applicable to ResNet. Switching ResNet to our design preserves or improves ResNet's performance properties. This suggests that aggregation topology is a fundamental consideration of its own, decoupled from other design differences between ResNet and DenseNet.

Section 4 provides full details on these experimental results. Prior to that, Sect. 2 relates background on the history and role of skip or aggregation links in convolutional neural networks. It places our contribution in the context of much of the recent research focus on CNN architecture.

Section 3 presents the details of our sparse aggregation strategy. Our approach occupies a previously unexplored position in aggregation complexity between that of standard CNNs and FractalNet [26] on one side, and ResNet and DenseNet on the other. Taken together with our experimental results, sparse aggregation appears to be a simple, general improvement that is likely to filter into the standard CNN backbone design. Section 5 concludes with a synthesis of these observations, and discussion of potential future research paths.

2 Related Work

Modern CNN architectures usually consist of a series of convolutional, ReLU, and batch normalization [23] operations, mixed with occasional max-pooling and subsampling stages. Much prior research focuses on optimizing for parameter efficiency within convolution, for example, via dimensionality reduction bottlenecks [16,17,22], grouped convolution [19,39], or weight compression [4]. These efforts all concern design at a micro-architectural level, optimizing structure that fits inside a single functional unit containing at most a few operations.

At a macro-architectural level, skip connections have emerged as a common and useful design motif. Such connections route outputs of earlier CNN layers directly to the input of far deeper layers, skipping over the sequence of intermediate layers. Some deeper layers thus take input from multiple paths: the usual sequential path as well as these shortcut paths. Multiple intuitions motivate inclusion of skip connections, and may share in explaining their effectiveness.

2.1 Skip Connections for Features

Predicting a detailed labeling of a visual scene may require understanding it at multiple levels of abstraction, from edges and textures to object categories. Taking the plausible view that a CNN learns to compute increasingly abstract visual representations when going from shallower to deeper layers, skip connections can provide a pathway for assembling features that combine many levels of abstraction. Building in such connections alleviates the burden of learning to store and maintain features computed early that the network needs again later.

This intuition motivates the skip connection structures found in many semantic segmentation CNNs. Fully convolutional networks [28] upsample and combine several layers of a standard CNN, to act as input to a final prediction layer. Hypercolumn networks [13] similarly wire intermediate representations into a concatenated feature descriptor. Rather than use the end layer as the sole destination for skip links, encoder-decoder architectures, such as SegNet [1] and U-Net [31], introduce internal skip links between encoder and decoder layers of corresponding spatial resolutions. Such internal feature aggregation, though with different connectivity, may also serve to make very deep networks trainable.

2.2 Training Very Deep Networks

Training deep networks end-to-end via stochastic gradient descent requires backpropagating a signal through the entire network. Starting from random initialization, the gradient received by earlier layers from a loss at the end of the network will be noisier than that received by deeper layers. This issue worsens with deeper networks, making them harder to train. Attaching additional losses to intermediate layers [27,35] is one strategy for ameliorating this problem.

Highway networks [33] and residual networks [16] (ResNets) offer a more elegant solution, preserving the ability to train from a single loss by adding skip connections to the network architecture. The addition of skip connections shortens the effective path length between early network layers and an informative loss. Highway networks add a gating mechanism, while residual networks implement skip connections by summing outputs of all previous layers. The effectiveness of the later strategy is cause for its current widespread adoption.

Fractal networks [26] demonstrate an alternative skip connection structure for training very deep networks. The accompanying analysis reveals skip connections function as a kind of scaffold that supports the training processes. Under special conditions, the FractalNet skip connections can be discarded after training.

DenseNets [20] build directly on ResNets, by switching the operational form of skip connections from summation to concatenation. They maintain the same aggregation topology as ResNets, as all previous layer outputs are concatenated.

2.3 Architecture Search

The dual motivations of building robust representations and enabling end-to-end training drive inclusion of internal aggregation links, but do not dictate an optimal procedure for doing so. Absent insight into optimal design methods, one can treat architectural details as hyperparameters over which to optimize [42]. Training of a single network can then be wrapped as a step in larger search procedure that varies network design.

However, it is unclear whether skip link topology is an important hyperparameter over which to search. Our proposed aggregation topology is motivated by a simple construction and, as shown in Sect. 4, significantly outperforms prior hand-designed structures. Perhaps our topology is near optimal and will free architecture search to focus on more consequential hyperparameters.

2.4 Concurrent Work

Concurrent work [18], independent of our own, proposes a modification of DenseNet similar to our SparseNet design. We make distinct contributions in comparison:

- Our SparseNet image classification results are substantially better than those reported in Hu *et al.* [18]. Our results represent an actual and significant improvement over the DenseNet baseline.
- We explore sparse aggregation topologies more generally, showing application to ResNet and DenseNet, whereas [18] proposes specific changes to DenseNet.
- We experiment with networks in extreme configurations (*e.g.* 1000 layers) in order to highlight the robustness of our design principles in regimes where current baselines begin to break down.

While we focus on skip connections in the context of both parameter efficiency and network trainability, other concurrent work examines alternative mechanisms to ensure trainability. Xiao *et al.* [38] develop a novel initialization scheme that allows training very deep vanilla CNNs. Chang *et al.* [2], taking inspiration from ordinary differential equations, develop a framework for analyzing stability of reversible networks [8] and demonstrate very deep reversible architectures.

3 Aggregation Architectures

Figure 1 sketches our proposed sparse aggregation architecture alongside the dominant ResNet [16] and DenseNet [20] designs, as well as the previously

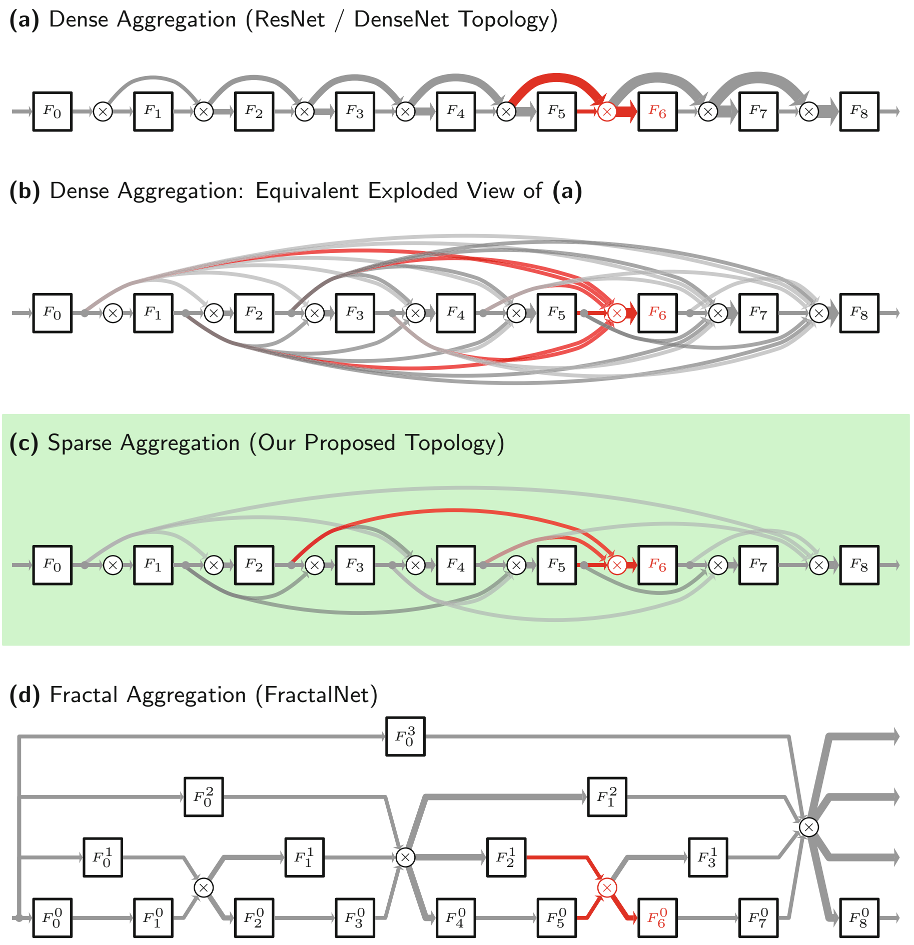
(a) Dense Aggregation (ResNet / DenseNet Topology)
(b) Dense Aggregation: Equivalent Exploded View of (a)
(c) Sparse Aggregation (Our Proposed Topology)
(d) Fractal Aggregation (FractalNet)
F_0 F_1 F_2 F_3 F_4 F_5 F_6 F_7 F_8
F_0^3 F_0^2 F_1^2 F_0^1 F_1^1 F_2^1 F_3^1
F_0^0 F_1^0 F_2^0 F_3^0 F_4^0 F_5^0 F_6^0 F_7^0 F_8^0

Fig. 1. Aggregation topologies. Our proposed sparse aggregation topology devotes less machinery to skip connections than DenseNet [20], but more than FractalNet [26]. This is apparent by comparing the exploded view (b) of the ResNet [16] or DenseNet topology (a), as well as the fractal topology (d), with our proposal (c). All of these architectures are describable in terms of a basic parameterized functional unit $F(\cdot)$ (*e.g.* convolution-ReLU-batchnorm), an aggregation operator $\otimes$, and a connection pattern. For ResNet, $\otimes$ is addition $[+]$; for DenseNet, $\otimes$ is concatenation $[\oplus]$; for FractalNet $\otimes$ is averaging $[\boxplus]$. Note how the compact view (a) feeds the result of one aggregation into the next; the exploded view (b) of DenseNet is the correct visualization for comparison to (c) and (d). For a network of depth N, dense aggregation requires $O(N^2)$ connections, sparse aggregation $O(N \log(N))$, and fractal aggregation $O(2N)$. These differences are visually apparent by comparing incoming links at a common depth. For example, compare the density of the (highlighted red) links into layer $F_6(\cdot)$.(Color figure online)

proposed FractalNet [26] alternative to ResNet. This macro-architectural view abstracts away details such as the specific functional unit $F(\cdot)$, parameter counts and feature dimensionality, and the aggregation operator $\otimes$. As our focus is on a novel aggregation topology, experiments in Sect. 4 match these other details to those of ResNet and DenseNet baselines.

We define a network with a sparse aggregation structure to be a sequence of nonlinear functional units (layers) $F_\ell(\cdot)$ operating on input x, with the output y_ℓ of layer ℓ computed as:

$$y_0 = F_0(x) \tag{1}$$

$$y_\ell = F_\ell(\otimes(y_{\ell-c^0}, y_{\ell-c^1}, y_{\ell-c^2}, y_{\ell-c^3}, \ldots, y_{\ell-c^k})) \tag{2}$$

where c is a positive integer and k is the largest non-negative integer such that $c^k \leq \ell$. $\otimes$ is the aggregation function. This amounts to connecting each layer to previous layers at exponentially increasing offsets. Contrast with ResNet and DenseNet, which connect each layer to all previous layers according to:

$$y_\ell = F_\ell(\otimes(y_{\ell-1}, y_{\ell-2}, y_{\ell-3}, y_{\ell-4}, \ldots, y_0)) \tag{3}$$

For a network of total depth N, the full aggregation strategy of ResNet and DenseNet introduces N incoming links per layer, for a total of $O(N^2)$ connections. In contrast, sparse aggregation introduces no more than $\log_c(N)$ incoming links per layer, for a total of $O(N\log(N))$ connections.

Our sparse aggregation strategy also differs from FractalNet's aggregation pattern. The FractalNet [26] design places a network of depth N in parallel with networks of depth $\frac{N}{2}$, $\frac{N}{4}$, ..., 1, making the total network consist of $2N-1$ layers. It inserts occasional join (aggregation) operations between these parallel networks, but does so with such extreme sparsity that the total connection count is still dominated by the $O(2N)$ connections in parallel layers.

Our sparse connection pattern is sparser than ResNet or DenseNet, yet denser than FractalNet. It occupies a previously unexplored point, with a fundamentally different scaling rate of skip connection density with network depth.

3.1 Potential Drawbacks of Dense Aggregation

The ability to train networks with depth greater than 100 layers using DenseNet and ResNet architectures can be partially attributed to to their feature aggregation strategies. As discussed in Sect. 2, skip links serve as a training scaffold, allowing each layer to be directly supervised by the final output layer, and aggregation may help transfer useful features from shallower to deeper layers.

However, dense feature aggregation comes with several potential drawbacks. These drawbacks appear in different forms in the ResNet-styled aggregation by summation and the DenseNet-styled aggregation by concatenation, but share a common theme of over-constraining or over-burdening the system.

In general, it is impossible to disentangle the original components of a set of features after taking their sum. As the depth of a residual network grows, the number of features maps aggregated grows linearly. Later features may corrupt or wash-out the information carried by earlier feature maps. This information loss caused by summation could partially explain the saturation of ResNet performance when the depth exceeds 1000 layers [16]. This way of combining features is also hard-coded in the design of ResNets, giving the model little flexibility to learn more expressive combination strategies. This constraint may be the reason that ResNet layers tend to learn to perform incremental feature updates [10].

In contrast, the aggregation style of DenseNets combines features through direct concatenation, which preserves the original form of the previous features. Concatenation allows each subsequent layer a clean view of all previously computed features, making feature reuse trivial. This factor may contribute to the better parameter-performance efficiency of DenseNet over ResNet.

But DenseNet's aggregation by concatenation has its own problems: the number of skip connections and required parameters grows at a rate of $O(N^2)$, where N is the network depth. This asymptotically quadratic growth means that a significant portion of the network is devoted to processing previously seen feature representations. Each layer contributes only a few new outputs to an ever-widening concatenation of stored state. Experiments show that it is hard for the model to make full use of all the parameters and dense skip connections. In the original DenseNet work [20], a large fraction of the skip connections have average absolute weights of convolution filters close to zero. This implies that dense aggregation of feature maps maintains some extraneous state.

The pitfalls of dense feature aggregation in both DenseNet and ResNet are caused by the linear growth in the number of features aggregated with respect to the depth. Variants of ResNet and DenseNet, including the post-activation ResNets [17], mixed-link networks [36], and dual-path networks [5] all use the same dense aggregation pattern, differing only by aggregation operator. They thus inherit potential limitations of this dense aggregation topology.

3.2 Properties of Sparse Aggregation

We would like to maintain the power of short gradient paths for training, while avoiding the potential drawbacks of dense feature aggregation. SparseNets do, in fact, have shorter gradient paths than architectures without aggregation.

Table 1. SparseNet properties. We compare architecture-induced scaling properties for networks of depth N and for individual layers located at depth ℓ.

	Parameters	Shortest gradient path	Aggregated features
Plain	$O(N)$	$O(N)$	$O(1)$
ResNets	$O(N)$	$O(1)$	$O(\ell)$
DenseNets	$O(N^2)$	$O(1)$	$O(\ell)$
SparseNets (sum)	$O(N)$	$O(\log(N))$	$O(\log \ell)$
SparseNets (concat)	$O(N \log N)$	$O(\log(N))$	$O(\log \ell)$

In plain feed-forward networks, there is only one path from a layer to a previous layer with offset S; the length of the path is $O(S)$. The length of the shortest gradient path is constant in dense aggregation networks like ResNet and DenseNet. However, the cost of maintaining a gradient path with $O(1)$ length between any two layers is the linear growth of the count of aggregated features. By aggregating features only from layers with exponential offset, the length of the shortest gradient path between two layers with offset S is bounded by $O((c-1)\log(S))$. Here, c is again the base of the exponent governing the sparse connection pattern.

It is also worth noting that the number of predecessor outputs gathered by the ℓ^{th} layer is $O(\log(\ell))$, as it only reaches predecessors with exponential offsets. Therefore, the total number of skip connections is

$$\sum_{\ell=1}^{N} \lfloor \log_c \ell \rfloor = O(N \log N) \tag{4}$$

where N is the number of layers (depth) of the network. The number of parameters are $O(N \log N)$ and $O(N)$, respectively, for aggregation by concatenation and aggregation by summation. Table 1 summarizes these properties.

4 Experiments

We demonstrate the effectiveness SparseNets as a drop-in replacement (and upgrade) for state-of-the-art networks with dense feature aggregation, namely ResNets [16,17] and DenseNets [20], through image classification tasks on the CIFAR [24] and ImageNet datasets [6]. Except for the difference between the dense and sparse aggregation topologies, we set all other SparseNet hyperparameters to be the same as the corresponding ResNet or DenseNet baseline.

For some large models, image classification accuracy appears to saturate when we continue increasing model depth or internal channel counts. It is likely such saturation is not due to model capacity limits, but rather both our model and baselines reach diminishing returns given the dataset size and task complexity. We are interested not only in absolute accuracy, but also parameter-accuracy and FLOP-accuracy efficiency.

We implement our models in the PyTorch framework [29]. For optimization, we use SGD with Nesterov momentum 0.9 and weight decay 0.0001. We train all models from scratch using He *et al.*'s initialization scheme [15]. All networks were trained using NVIDIA GTX 1080 Ti GPUs. We release our implementation[1] of SparseNets, with full details of model architecture and parameter settings, for the purpose of reproducible experimental results.

[1] https://github.com/Lyken17/SparseNet.

4.1 Datasets

CIFAR. Both the CIFAR-10 and CIFAR-100 datasets [24] have 50,000 training images and 10,000 testing images with size of 32×32 pixels. CIFAR-10 (C10) and CIFAR-100 (C100) have 10 and 100 classes respectively. Our experiments use standard data augmentation, including mirroring and shifting, as done in [20]. The mark + beside C10 or C100 in results tables indicates this data augmentation scheme. As preprocessing, we normalize the data by the channel mean and standard deviation. Following the schedule from the Torch implementation of ResNet [11], our learning rate starts from 0.1 and is divided by 10 at epoch 150 and 225.

ImageNet. The ILSVRC 2012 classification dataset [6] contains 1.2 million images for training, and 50K for validation, from 1000 classes. For a fair comparison, we adopt the standard augmentation scheme for training images as in [11,16,20]. Following [16,20], we report classification errors on the validation set with single crop of size 224×224.

4.2 Results on CIFAR

Table 2 reports experimental results on CIFAR [24]. The best SparseNet closely matches the performance of the state-of-the DenseNet. We also show the results for a selection of DenseNet and SparseNet models over multiple runs on the CIFAR-100 dataset in the supplementary material. Multiple runs exhibit similar accuracies with low variance. In all of these experiments, we instantiate each SparseNet to be exactly the same as the correspondingly named DenseNet, but with sparser aggregation structure (some connections removed). The parameter k indicates feature growth rates (how many new feature channels each layer produces), which we match to the DenseNet baseline. Models whose names end with BC use the bottleneck compression structure, as in the original DenseNet paper. As SparseNet does fewer concatenations than DenseNet, the same feature growth rate produces models with fewer overall parameters. Remarkably, for many of the corresponding 100 layer models, SparseNet performs as well or better than DenseNet, while having substantially fewer parameters.

4.3 Going Deeper with Sparse Connections

Table 3 shows results of pushing architectures to extreme depth. While Table 2 explored only the SparseNet analogue of DenseNet, we now explore switching both ResNet and DenseNet to sparse aggregation structures, and denote their corresponding SparseNets by SparseNet[+] and SparseNet[$\oplus$], respectively.

Both ResNet and SparseNet[+] demonstrate better performance on CIFAR100 as their depth increases from 56 to 200 layers. The gap between the performance of ResNet and SparseNet[+] initially enlarges as depth increases. However, it narrows as network depth reaches 1001 layers, and the performance of SparseNet[+]-2000 surpasses ResNet-2000. Compared to ResNet, SparseNet[+] appears better able to scale to depths of over 1000 layers.

Similar to both ResNet and SparseNet[+], the performance of DenseNet and SparseNet[⊕] also improves as their depth increases. The performance of DenseNet is also affected by the feature growth rate. However, the parameter count of DenseNet explodes as we increase its depth to 400, even with a growth rate of 12. Bottleneck compression layers have to be adopted and the number of filters in each layer has to be significantly reduced if we want to go deeper. We experiment with DenseNets of depth greater than 1000 by adopting bottleneck compression (BC) structure and using a growth rate of 4. But, as Table 3 shows, their performance is far from satisfying. In contrast, building SparseNet[⊕] with more than 1000 layers is practical and memory-efficient. We can easily build SparseNet[⊕] with depth greater than 400 using BC structure and a growth rate of 12. At 1001 layers, it achieves far better performance than DenseNet-1001.

Table 2. CIFAR classification performance. We show classification error rate for SparseNets compared to DenseNets, ResNets, and their variants. Results marked with a $*$ are from our implementation. Datasets marked with + indicates use of standard data augmentation (translation and mirroring).

Architecture	Depth	Params	C10+	C100+
ResNet [16]	110	1.7M	6.61	-
ResNet(pre-activation) [16]	164	1.7M	5.46	24.33
ResNet(pre-activation) [16]	1001	10.2M	4.62	21.42*
Wide ResNet [40]	16	11.0M	4.81	22.07
FractalNet [26]	21	38.6M	5.52	23.30
DenseNet (k = 12) [20]	40	1.1M	5.39*	24.79*
DenseNet (k = 12) [20]	100	7.2M	4.28*	20.97*
DenseNet (k = 24) [20]	100	28.3M	4.04*	19.61*
DenseNet (k = 16, 32, 64) [20]	100	61.1M	4.31*	20.6*
DenseNet (k = 32, 64, 128) [20]	100	241.6M	N/A	N/A
DenseNet-BC (k = 24) [20]	250	15.3M	**3.65**	17.6
DenseNet-BC (k = 40) [20]	190	25.6M	3.75*	**17.53***
DenseNet-BC (k = 16, 32, 64) [20]	100	7.9M	4.02*	19.55*
DenseNet-BC (k = 32, 64,128) [20]	100	30.5M	3.92*	18.71*
SparseNet (k = 12)	40	0.8M	5.13	24.65
SparseNet (k = 24)	100	2.5M	4.64	22.41
SparseNet (k = 36)	100	5.7M	4.34	20.50
SparseNet (k = 16, 32, 64)	100	7.2M	4.11	19.49
SparseNet (k = 32, 64, 128)	100	27.7M	3.88	18.80
SparseNet-BC (k = 24)	100	1.5M	4.03	22.12
SparseNet-BC (k = 36)	100	3.3M	3.91	20.31
SparseNet-BC (k = 16, 32, 64)	100	4.4M	3.43	19.71
SparseNet-BC (k = 32, 64, 128)	100	16.7M	**3.22**	**17.71**

Table 3. Depth scalability on CIFAR. *Left:* ResNets and their sparsely aggregated analogue SparseNets[+]. *Right:* DenseNets and their corresponding sparse analogues SparseNets[⊕]. Observe that ResNet and all SparseNet variants of any depth exhibit robust performance. DenseNets suffer an efficiency drop when stretched too deep.

Model	Depth	Params	CIFAR 100+
ResNet	56	0.59M	27.00
	110	1.15M	24.70
	200	2.07M	23.10
	1001	10.33M	**21.42**
	2000	20.62M	22.76
SparseNet[+]	56	0.59M	27.70
	110	1.15M	26.10
	200	2.07M	25.77
	1001	10.33M	22.10
	2000	20.62M	**21.01**

Model	Depth	Params	CIFAR 100+
DenseNet(k=12)	40	1.10M	24.79
	100	7.20M	20.97
	400	117M	N/A
DenseNet-BC(k=24)	250	25.6M	17.6
	400	216.3M	N/A
DenseNet-BC(k=4)	400	1.10M	32.94
	1001	6.63M	28.50
SparseNet[⊕]-BC (k=12)	100	0.40M	27.99
	400	1.70M	24.41
	1001	4.62M	22.10

An important advantage of SparseNet[⊕] over DenseNet is that the number of previous layers aggregated can be bounded by a small integer even when the depth of the network is over 1000 layers. This is a consequence of the slow growth rate of the logarithm function. This feature not only permits building deeper SparseNet[⊕] variants, but allows us to explore hyperparameters of SparseNet[⊕] with more flexibility on both depth and filter count.

We also observe that SparseNet[⊕] generally has better parameter efficiency than SparseNet[+]. For example, on CIFAR-100, the error rate of SparseNet[+]-1001 and SparseNet[⊕]-1001 are (coincidently) both 22.10. However, notice that SparseNet[⊕]-1001 requires less than half the parameters of SparseNet[+]-1001. Similar trends are also seen in the comparison between SparseNet[+]-200 and SparseNet[⊕]-400. The DenseNet *vs.* ResNet advantage of preserving features via concatenation (over summation) also holds for the sparse aggregation pattern.

4.4 Efficiency of SparseNet[⊕]

Returning to Table 2, we can further comment on the efficiency of SparseNet[⊕] (denoted in Table 2 as SparseNet) in comparison to DenseNet. These results include our exploration of parameter efficiency by varying the depth and number of filters of SparseNet[⊕]. As the number of features each layer aggregates grows slowly, and is nearly a constant within a block, we also double the number of filters across blocks, following the approach of ResNets. Here, a block refers to a sequence of layers running at the same spatial resolution, between pooling and subsampling stages of the CNN pipeline.

There are two general trends in the results. First, SparseNet usually requires fewer parameters than DenseNet when they have close performance. Most notably, DenseNet-BC ($N = 190$, $k = 40$) requires 25.6 million parameters to achieve error rate 17.53% on CIFAR100+, while SparseNet-BC can achieve a similar error of 17.71% under setting ($N = 100$, $k = \{32, 64, 128\}$) with only 16.7 million parameters. The 19.71% error rate of SparseNet-BC ($N = 100$,

$k = \{16, 32, 64\}$) is close to the performance of the corresponding DenseNet-BC ($N = 100$, $k = \{16, 32, 64\}$) but requires 4.4 rather than 7.9 million parameters.

Second, when both networks have less than 15 million parameters, SparseNet always outperforms the DenseNet with similar parameter count. For example, DenseNet ($N = 100, k = 12$) and SparseNet ($N = 100, k = \{16, 32, 64\}$) both have around 7.2 million parameters, but the latter shows better performance. DenseNet ($N = 40, k = 12$) consumes around 1.1 million parameters but still has worse performance than the 0.8 million param SparseNet ($N = 40, k = 12$).

Counterexamples do exist, such as the comparison between SparseNet-BC-100-{32, 64, 128} and DenseNet-BC-250-24. The latter model, with fewer parameters, performs slightly better (17.6% vs 17.71% error) than the previous one. We argue this is an example of performance saturation considering DenseNet-BC-190-40 only has slightly higher accuracy than DenseNet-BC-250-24, with many more parameters (25.6 million *vs.* 15.3 million). These large networks may be close to saturating performance on the CIFAR-100 image classification task.

Table 4. ImageNet results. The top-1 single-crop validation error, parameters, FLOPs, and time of each model on ImageNet.

Model	Error	Params	FLOPs	Time
DenseNet-121-32 [20]	25.0*	7.98M	5.7G	19.5ms
DenseNet-169-32 [20]	23.6*	14.15M	6.76G	32.0ms
DenseNet-201-32 [20]	22.6*	20.01M	8.63G	42.6ms
DenseNet-264-32 [20]	**22.2***	27.21M	11.03G	50.4ms
SparseNet[$\oplus$]-121-32	25.6	4.51M	3.46G	13.5ms
SparseNet[$\oplus$]-169-32	24.2	6.23M	3.74G	18.8ms
SparseNet[$\oplus$]-201-32	23.1	7.22M	4.13G	22.0ms
SparseNet[$\oplus$]-201-48	**22.1**	14.91M	9.19G	43.1ms
ResNet-50	23.9	25.5M	8.20G	42.2ms
ResNet-50 Pruned [12]	**23.7**	7.47M	-	-

Note that when we double the number of filters across different blocks, the performance of SparseNets is boosted and their better parameter efficiency over DenseNets becomes more obvious. SparseNets achieve similar or better performance than DenseNets, while requiring at most half the number of parameters uniformly across all settings. These general trends are summarized in the parameter-performance plots in Fig. 2 (left).

4.5 Results on ImageNet

To demonstrate efficiency on a larger-scale dataset, we further test different configurations of SparseNet[$\oplus$] and compare them with state-of-the-art networks on ImageNet. All models are trained with the same preprocessing methods and hyperparameters. Table 4 reports ImageNet validation error.

These results reveal that the better parameter-performance efficiency exhibited by SparseNet[⊕] over DenseNet extends to ImageNet [6]: SparseNet[⊕] performs similarly to state-of-the-art DenseNets, while requiring significantly fewer parameters. For example, SparseNet-201-48 (14.91M params) yields better validation error than DenseNet-201-32 (20.01M params). SparseNet-201-32 (7.22M params) outperforms DenseNet-169-32 with just half the parameter count.

Even compared to pruned networks, SparseNets show competitive parameter efficiency. In the last row of the Table 4, we show the result of pruning ResNet-50 using deep compression [12], whose parameter-performance efficiency significantly outpaces the unpruned ResNet-50. However, our SparseNet[⊕]-201-32, trained from scratch, has even better error rate than pruned ResNet-50, with fewer parameters. See Fig. 2 (right) for a complete efficiency plot.

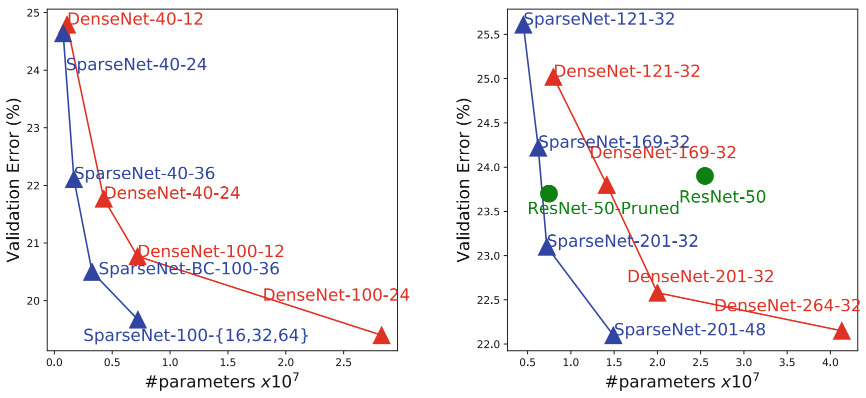

Fig. 2. Parameter efficiency. Comparison between DenseNets and SparseNets[⊕] on top-1% error and number of parameters with different configurations. *Left:* CIFAR. *Right:* ImageNet. SparseNets achieve lower error with fewer parameters.

4.6 Feature Reuse and Parameter Redundancy

The original DenseNets work [20] conducts a simple experiment to investigate how well a trained network reuses features across layers. In short, for each layer in each densely connected block, they compute the average absolute weights of the part of that layer's filters that convolves with each previous layer's feature map. The averaged absolute weights are rescaled between 0 and 1 for each layer i. The j^{th} normalized value implies the relative dependency of the features of layer i on the features of layer j, compared to other layers. These experiments are performed on a DenseNet consisting of 3 blocks with $N = 40$ and $k = 12$.

We perform a similar experiment on a SparseNet model with the same configuration. We plot results as heat maps in Fig. 3. For comparison, we also include

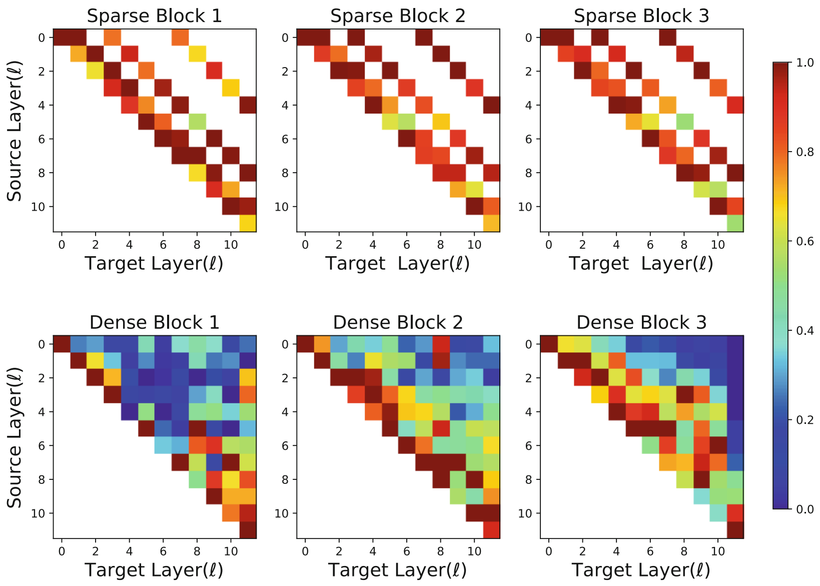

Fig. 3. The average absolute filter weights of convolutional layers in trained DenseNet and SparseNet. The color of pixel (i, j) indicates the average weights of connections from layer i to j within a block. The first row encodes the weights attached to the first input layer of a DenseNet/SparseNet block. (Color figure online)

the heat maps of the corresponding experiment on DenseNets [20]. In these heap maps, a red pixel at location (i, j) indicates layer i makes heavy use of the features of layer j; a blue pixel indicates relatively little usage. A white pixel indicates there is no direct connection between layer i and layer j. From the heat maps, we observe the following:

– Most of the non-white elements in the heat map of SparseNet are close to red, indicating that each layer takes full advantage of all the features it directly aggregates. It also indicates almost all the parameters are fully exploited, leaving little parameter redundancy. This result is not surprising considering the high observed parameter-performance efficiency of our model.
– In general, the layer coupling value at position (i, j) in DenseNet decreases as the offset between i and j gets larger. However, such a decaying trend does not appear in the heat map of SparseNet, implying that layers in SparseNets have better ability to extract useful features from long-distance connections to preceding layers.

The distribution of learned weights in Fig. 3, together with the efficiency curves in Fig. 2, serves to highlight the importance of optimizing macro-architectural design. Others have demonstrated the benefits of a range of schemes [9,19,22,30,37] for sparsifying micro-architectural network structure (parameter structure within layers or filters). Our results show similar considerations are relevant at the scale of the entire network.

5 Conclusion

We demonstrate that following a simple design rule, scaling aggregation link complexity in a logarithmic manner with network depth, yields a new state-of-the-art CNN architecture. Extensive experiments on CIFAR and ImageNet show our SparseNets offer significant efficiency improvements over the widely used ResNets and DenseNets. This increased efficiency allows SparseNets to scale robustly to great depth. While CNNs have recently moved from the 10-layer to 100-layer regime, perhaps new possibilities will emerge with straightforward and robust training of 1000-layer networks.

The performance of neural networks for visual recognition has grown with their depth as they evolved from AlexNet [25] to ResNet [16,17]. Extrapolating such trends could be cause to believe building deeper networks should further improve performance. Much effort has been devoted by researchers in computer vision and machine learning communities to train deep neural networks with more than 1000 layers, with such hope [17,21].

Though previous works and our experiments show we can train very deep neural networks with stochastic gradient descent, their test performance still usually plateaus. Even so, very deep neural networks might be suitable for other interesting tasks. One possible future direction could be solving sequential search or reasoning tasks relying on long-term dependencies. Skip connections might empower the network with backtracking ability. Sparse feature aggregation might permit building extremely deep neural networks for such tasks.

References

1. Badrinarayanan, V., Kendall, A., Cipolla, R.: SegNet: a deep convolutional encoder-decoder architecture for image segmentation. PAMI **39**, 2481–2495 (2017)
2. Chang, B., Meng, L., Haber, E., Ruthotto, L., Begert, D., Holtham, E.: Reversible architectures for arbitrarily deep residual neural networks. In: AAAI (2018)
3. Chen, L.C., Papandreou, G., Kokkinos, I., Murphy, K., Yuille, A.L.: DeepLab: semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs. arXiv:1606.00915 (2016)
4. Chen, W., Wilson, J.T., Tyree, S., Weinberger, K.Q., Chen, Y.: Compressing neural networks with the hashing trick. In: ICML (2015)
5. Chen, Y., Li, J., Xiao, H., Jin, X., Yan, S., Feng, J.: Dual path networks. In: NIPS (2017)
6. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: ImageNet: a large-scale hierarchical image database. In: CVPR (2009)
7. Girshick, R., Donahue, J., Darrell, T., Malik, J.: Rich feature hierarchies for accurate object detection and semantic segmentation. In: CVPR (2014)
8. Gomez, A.N., Ren, M., Urtasun, R., Grosse, R.B.: The reversible residual network: backpropagation without storing activations. In: NIPS (2017)

9. Gray, S., Radford, A., Kingma, D.P.: GPU kernels for block-sparse weights. Technical report, OpenAI (2017)
10. Greff, K., Srivastava, R.K., Schmidhuber, J.: Highway and residual networks learn unrolled iterative estimation. In: ICLR (2017)
11. Gross, S., Wilber, M.: Training and investigating residual nets (2016). https://github.com/facebook/fb.resnet.torch
12. Han, S., Mao, H., Dally, W.J.: Deep compression: compressing deep neural networks with pruning, trained quantization and huffman coding. In: ICLR (2016)
13. Hariharan, B., Arbelaez, P., Girshick, R., Malik, J.: Hypercolumns for object segmentation and fine-grained localization. In: CVPR (2015)
14. He, K., Gkioxari, G., Dollar, P., Girshick, R.: Mask R-CNN. In: ICCV (2017)
15. He, K., Zhang, X., Ren, S., Sun, J.: Delving deep into rectifiers: surpassing human-level performance on imagenet classification. In: ICCV (2015)
16. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: CVPR (2016)
17. He, K., Zhang, X., Ren, S., Sun, J.: Identity mappings in deep residual networks. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds.) ECCV 2016. LNCS, vol. 9908, pp. 630–645. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46493-0_38
18. Hu, H., Dey, D., Giorno, A.D., Hebert, M., Bagnell, J.A.: Log-DenseNet: How to sparsify a DenseNet. arXiv:1711.00002 (2017)
19. Huang, G., Liu, S., van der Maaten, L., Weinberger, K.Q.: CondenseNet: an efficient densenet using learned group convolutions. In: CVPR (2018)
20. Huang, G., Liu, Z., van der Maaten, L., Weinberger, K.Q.: Densely connected convolutional networks. In: CVPR (2017)
21. Huang, G., Sun, Y., Liu, Z., Sedra, D., Weinberger, K.Q.: Deep networks with stochastic depth. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds.) ECCV 2016. LNCS, vol. 9908, pp. 646–661. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46493-0_39
22. Iandola, F.N., Moskewicz, M.W., Ashraf, K., Han, S., Dally, W.J., Keutzer, K.: SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <1MB model size. arXiv:1602.07360 (2016)
23. Ioffe, S., Szegedy, C.: Batch normalization: accelerating deep network training by reducing internal covariate shift. In: ICML (2015)
24. Krizhevsky, A., Hinton, G.: Learning multiple layers of features from tiny images. Technical report, University of Toronto (2009)
25. Krizhevsky, A., Sutskever, I., Hinton, G.E.: ImageNet classification with deep convolutional neural networks. In: NIPS (2012)
26. Larsson, G., Maire, M., Shakhnarovich, G.: FractalNet: Ultra-deep neural networks without residuals. In: ICLR (2017)
27. Lee, C.Y., Xie, S., Gallagher, P., Zhang, Z., Tu, Z.: Deeply-supervised nets. In: AISTATS (2015)
28. Long, J., Shelhamer, E., Darrell, T.: Fully convolutional networks for semantic segmentation. In: CVPR (2015)
29. Paszke, A., et al.: PyTorch: tensors and dynamic neural networks in python with strong GPU acceleration, May 2017
30. Prabhu, A., Varma, G., Namboodiri, A.M.: Deep expander networks: efficient deep networks from graph theory. arXiv:1711.08757 (2017)
31. Ronneberger, O., Fischer, P., Brox, T.: U-Net: convolutional networks for biomedical image segmentation. In: Navab, N., Hornegger, J., Wells, W.M., Frangi, A.F. (eds.) MICCAI 2015. LNCS, vol. 9351, pp. 234–241. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-24574-4_28

32. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. In: ICLR (2015)
33. Srivastava, R.K., Greff, K., Schmidhuber, J.: Highway networks. arXiv:1505.00387 (2015)
34. Szegedy, C., Ioffe, S., Vanhoucke, V., Alemi, A.A.: Inception-v4, Inception-ResNet and the impact of residual connections on learning. In: AAAI (2017)
35. Szegedy, C., et al.: Going deeper with convolutions. In: CVPR (2015)
36. Wang, W., Li, X., Yang, J., Lu, T.: Mixed link networks. arXiv:1802.01808 (2018)
37. Wen, W., Wu, C., Wang, Y., Chen, Y., Li, H.: Learning structured sparsity in deep neural networks. In: NIPS (2016)
38. Xiao, L., Bahri, Y., Sohl-Dickstein, J., Schoenholz, S.S., Pennington, J.: Dynamical isometry and a mean field theory of CNNs: how to train 10,000-layer vanilla convolutional neural networks. In: ICML (2018)
39. Xie, S., Girshick, R., Dollár, P., Tu, Z., He, K.: Aggregated residual transformations for deep neural networks. In: CVPR (2017)
40. Zagoruyko, S., Komodakis, N.: Wide residual networks. In: BMVC (2016)
41. Zhao, H., Shi, J., Qi, X., Wang, X., Jia, J.: Pyramid scene parsing network. In: CVPR (2017)
42. Zoph, B., Le, Q.V.: Neural architecture search with reinforcement learning. arXiv:1611.01578 (2016)