Gated Convolutional Networks with Hybrid Connectivity for Image Classification

Chuanguang Yang,[1,2] **Zhulin An,**[1*] **Hui Zhu,**[1,2] **Xiaolong Hu,**[1,2]
Kun Zhang,[1,2] **Kaiqiang Xu,**[1,2] **Chao Li,**[1] **Yongjun Xu**[1]

[1]Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China
[2]University of Chinese Academy of Sciences, Beijing, China
{yangchuanguang, anzhulin, zhuhui, huxiaolong18g, zhangkun17g, xukaiqiang, lichao, xyj}@ict.ac.cn

Abstract

We propose a simple yet effective method to reduce the redundancy of DenseNet by substantially decreasing the number of stacked modules by replacing the original bottleneck by our SMG module, which is augmented by local residual. Furthermore, SMG module is equipped with an efficient two-stage pipeline, which aims to DenseNet-like architectures that need to integrate all previous outputs, i.e., squeezing the incoming informative but redundant features gradually by hierarchical convolutions as a hourglass shape and then exciting it by multi-kernel depthwise convolutions, the output of which would be compact and hold more informative multi-scale features. We further develop a forget and an update gate by introducing the popular attention modules to implement the effective fusion instead of a simple addition between reused and new features. Due to the **H**ybrid **C**onnectivity (nested combination of global dense and local residual) and **G**ated mechanisms, we called our network as the **HCGNet**. Experimental results on CIFAR and ImageNet datasets show that HCGNet is more prominently efficient than DenseNet, and can also significantly outperform state-of-the-art networks with less complexity. Moreover, HCGNet also shows the remarkable interpretability and robustness by network dissection and adversarial defense, respectively. On MS-COCO, HCGNet can consistently learn better features than popular backbones.

Introduction

Deep convolutional neural networks (CNNs) are becoming more and more efficient in parameter and computation without sacrificing the performance owing to novel architectures design. ResNet (He et al. 2016) introduces the residual connectivity to implement the addition of the input and output features for each micro-block. DenseNet (Huang et al. 2017) holds the dense connectivity by changing skip connections from addition to concatenation. Both of their feature aggregation connectivities can not only encourage feature reuse, but also ease the training problems. For a detailed comparison, dense connectivity is more effect for feature exploitation and exploration but exists a certain redundancy, while residual connectivity contributes to efficient feature reuse by
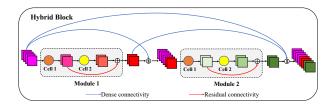
---

*Corresponding author
The codes are available at: https://github.com/winycg/HCGNet

Figure 1: The diagram of a hybrid block including $n = 2$ modules, where $n \geqslant 2$. The symbol "+" and "$\|$" denote element-wise addition and channel-wise concatenation among multiple feature maps, respectively.

parameter sharing mechanism and thus leads to low redundancy, but lacks the capability of feature preservation and exploration. To enjoy their advantages and avoid inherent limitations, many networks combine them to build a more effective aggregation topology, such as DPN (Chen et al. 2017), MixNet (Wang et al. 2018a) and AOGNet (Li, Song, and Wu 2019). Different from them, we develop a hybrid connectivity (Figure 1) with nested aggregation that facilitates feature flow by dense connectivity for global channel-wise concatenation of outputs produced by all precedent modules (blue links in Figure 1) and residual connectivity for local element-wise addition within the module (red links in Figure 1).

Our main motivation for this pattern design originates from reducing the redundancy of dense connectivity. As the depth of network linearly increases, the number of skip connections and required parameters grow by a rate of $O(n^2)$, where n denotes the number of stacked modules under dense connectivity. Meanwhile, early superfluous features which have few contributions are transferred quadratically to subsequent modules. So one simple method to reduce redundancy is to decrease the number of modules directly, but it can attenuate the representational power of features and then deteriorate the performance. Thus we develop a novel module by embedding the residual connectivity to assist feature learning within the local module. Experimentally, the number of our proposed modules under dense connectivity can be quite fewer than that of classical modules in the dense block but without sacrificing the performance.

For further adaptation with hybrid connectivity, we instantiate the basic module that includes a **squeeze cell (cell 1 in Figure 1)** for transforming the input to a compact feature map, and a **multi-scale excitation cell (cell 2 in Figure 1)** to further extract multi-scale features by multi-kernel convolutions. It is widely known that convolution builds pixel relationship in a local neighborhood, which leads to ineffective modeling of long-range dependency. To fully address this issue, we develop an update gate to model the global context features from more informative multi-scale features. Moreover, we locate a forget gate on the residual connection to capture channel-wise dependency for decaying the reused features produced by cell 1. Finally, global context features are added to the reused feature map of each spatial position to form the output, which can not only promote effective feature exploration but also retain the capability of feature re-exploitation to some extent. Moreover, both forget gate and update gate are lightweight and general plug-ins, which can be integrated into any CNNs with negligible overheads.

We perform extensive experiments across the three highly competitive image classification datasets: CIFAR-10/100 (Krizhevsky and Hinton 2009), and ImageNet (ILSVRC 2012) (Deng et al. 2009). On CIFAR datasets, HCGNets outperform state-of-the-art both human-designed and auto-searched networks but only requiring extremely fewer parameters, e.g., HCGNet-A3 obtains the better result than the most competitive NASNet-A (Zoph et al. 2018) with $4.5\times$ fewer parameters. On ImageNet datasets, it also consistently obtains the best accuracy, interpretability, robustness based on classification and transferability to object detection as well as segmentation among the widely used networks with least complexity, e.g., HCGNet-B outperforms previous SOTA AOGNet across a broad range of tasks with similar complexity.

Related Work

Improvements of ResNet and DenseNet. ResNeXt (Xie et al. 2017) outperforms ResNet with less overheads since it adopts 3×3 group convolutions in residual blocks. Afterwards, group convolutions become popular in efficient CNNs design due to the properties of lower parameter and computational cost, including our HCGNets. Wide ResNet (Zagoruyko and Komodakis 2016) show that increasing width while decreasing depth of residual networks can surpass very deep counterparts, meanwhile tackling the problems of slow training and weakened feature reuse. By representing the multi-scale features and widening the receptive fields (RF) within the residual block, Res2Net (Gao et al. 2019) outperforms the other backbones across a broad range of tasks. Multi-scale information has been widely demonstrated a effective way to improve the performance, our HCGNet also constructs the multi-scale features by multi-branch convolutions.

It is widely known that DenseNet has a certain redundancy, thus a typical practice is sparsification. Log-DenseNet (Hu et al. 2017) and SparseNet (Zhu et al. 2018) regularly conduct a sparse rather than full aggregation of all previous outputs, which change the number of connections from linear to be logarithmic in the overall topology.

Learned group convolutions are adopted in CondenseNet (Huang et al. 2018) to automatically prune unimportant channels for the incoming feature map based on channel-wise L1-norm. However, excessive sparsification affects the superiority of collective learning. Thus we only decrease the number of modules under dense connectivity to reduce redundancy, which is empirically more effective than sparsification.

Combinations of ResNet and DenseNet. To enjoy the advantages and avoid drawbacks of both two connectivities, many combinations have proposed. DPN adopts dual path architectures, which can facilitate effective feature reuse by residual path and feature exploration by dense path in parallel. MixNet blends two connectivities to implement feature aggregation with more flexible positions and sizes, further ResNet, DenseNet and DPN can be treated as particular cases of MixNet. Recently proposed AOGNet utilizes AND-OR Grammar to generate CNNs by parsing feature map as a sentence, where AND-node denotes channel-wise concatenation and OR-node denotes element-wise addition. It demonstrates that the compositional and hierarchical aggregation in AOGNet is more effective than cascade-based way in DPN. Moreover, addition and concatenation as the meta-operations are also widely applied in the field of neural architecture search, such as NASNet, PNASNet (Liu et al. 2018) and AmoebaNet (Real et al. 2019). Extensive experiments indicate that the nested way for feature aggregation in our HCGNets perform the best.

Attention Mechanisms Attention has been widely applied in computer vision, e.g., image classification (Wang et al. 2017). SENet (Hu, Shen, and Sun 2018) introduces a lightweight gate to capture channel-wise dependencies for rescaling channel features. SKNet (Li et al. 2019) further employs a dynamic kernel selection attention for weighted multi-scale features fusion, which is inspired by Inception-Nets (Szegedy et al. 2017). Beyond channel, CBAM (Woo et al. 2018) also constructs a spatial attention map to recalibrate spatial features. To capture long-range dependency, GCNet (Cao et al. 2019) simplifies non-local block (Wang et al. 2018b) to implement query-independent context modeling based on single branch information. Different from them in roles or mechanisms, we build a forget gate to capture channel-wise dependency for decaying the reused features, while an update gate fully models the global context features from multi-scale information.

Revisiting ResNet and DenseNet

We revisit the classical ResNet and DenseNet with their individual residual connectivity and dense connectivity, and further investigate their mechanisms of parameter sharing and feature learning. Finally, we analyse the overall efficiency of ResNet and DenseNet.

Parameter Sharing

Intuitively, residual connectivity implicitly accompanies a parameter sharing mechanism between the reused and newly extracted features by processing their mixed features. We

now formally describe why the parameter sharing mechanism can take place in residual connectivity but not in dense connectivity. Concretely, we use $\mathcal{F}$ to denote the bottleneck unit. Consider the input feature map $\mathbf{x}_{l-1} \in \mathbb{R}^{H \times W \times C}$ to the l-th residual block, corresponding formula is as follows:

$$\mathbf{x}_l = \mathbf{x}_{l-1} + \mathcal{F}_l(\mathbf{x}_{l-1}; W_l) = \mathbf{x}_{l-1} + \tilde{\mathbf{x}}_l \qquad (1)$$

Where $\mathbf{x}_{l-1}$ can be considered as the reused feature map, W_l and $\mathcal{F}_l(\mathbf{x}_{l-1}; W_l)$ refer to convolutional weights and newly extracted feature map, respectively. $\tilde{\mathbf{x}}_l \in \mathbb{R}^{H \times W \times C}$ represents $\mathcal{F}_l(\mathbf{x}_{l-1}; W_l)$ for simplicity. Afterwards, $\mathbf{x}_l$ becomes a new input for the next residual block to proceed the transformation:

$$\begin{aligned} \mathbf{x}_{l+1} &= \mathbf{x}_l + \mathcal{F}_{l+1}(\mathbf{x}_l; W_{l+1}) \\ &= \mathbf{x}_l + \mathcal{F}_{l+1}(\mathbf{x}_{l-1} + \tilde{\mathbf{x}}_l; W_{l+1}) \end{aligned} \qquad (2)$$

In the $l+1$-th residual block, $\mathbf{x}_{l-1}$ and $\tilde{\mathbf{x}}_l$ are shared with the same W_{l+1} and operations. Similar analysis about dense connectivity is exhibited as follows. Output of the l-th module under dense connectivity can be regarded as the concatenation of input $\mathbf{x}_{l-1} \in \mathbb{R}^{H \times W \times C}$ and newly extracted feature map $\tilde{\mathbf{x}}_l \in \mathbb{R}^{H \times W \times \tilde{C}}$ along the channels:

$$\mathbf{x}_l = \mathbf{x}_{l-1} \parallel \mathcal{F}_l(\mathbf{x}_{l-1}; W_l) = \mathbf{x}_{l-1} \parallel \tilde{\mathbf{x}}_l \qquad (3)$$

Then, the next module receives $\mathbf{x}_l \in \mathbb{R}^{H \times W \times (C + \tilde{C})}$ and conducts the following transformation:

$$\begin{aligned} \mathbf{x}_{l+1} &= \mathbf{x}_l \parallel \mathcal{F}_{l+1}(\mathbf{x}_l; W_{l+1}) \\ &= \mathbf{x}_l \parallel \mathcal{F}_{l+1}(\mathbf{x}_{l-1} \parallel \tilde{\mathbf{x}}_l; W_{l+1}) \end{aligned} \qquad (4)$$

In the $l+1$-th dense block, $\mathbf{x}_{l-1}$ and $\tilde{\mathbf{x}}_l$ are not shared with the same W_{l+1} because of the different locations of feature space between reused and newly extracted feature maps.

Feature Learning

The final output of residual block is the element-wise addition of input and newly extracted feature maps. This addition pattern facilitates efficient feature reuse without increasing the size of feature map thus reducing parameter redundancy. But one potential fact is that too many aggregations by addition may collapse the feature representation and thus impede the information flow, hence some early informative features may be lost inevitably. Moreover, parameter sharing mechanism may damage the capability of exploring new features.

Subsequently proposed DenseNet develops a global dense connectivity, where the output feature map of each preceding module flows to the all subsequent modules directly. Different from the element-wise addition, input and newly extracted feature maps are combined by concatenation along the channels. Thus dense connectivity can transfer the early feature maps to later modules, which preserves the all preceding feature information and facilitate the full exploitation of existing features. Moreover, various modules with different weights conduct a collective learning for the same features, which can promote effective feature exploration.

Overall Efficiency

It is widely known that DenseNet-100 with 0.8M parameters slightly outperforms ResNet-1001 with 10.2M parameters on CIFAR10 dataset. The explicit parameter gap is that DenseNet-100 is quite shallower than ResNet-1001 due to the more effective feature exploitation and exploration capabilities produced by collective learning, while ResNet mainly depends on increasing depth to improve the representational power of features. Empirically, DenseNet can also have extremely few number of filters in each convolutional layer due to the collective learning mechanism that further improve the efficiency. However, one potential weakness of dense connectivity is the redundancy of repeated extraction with the same features. In this case, early features flow to all subsequent layers, even if they have few contributions. By contrast, residual connectivity has a relatively low redundancy due to the parameter sharing mechanism.

Networks Architecture

Hybrid Connectivity Pattern

We develop a hybrid connectivity pattern, which can enjoy the effective feature learning and few filters of each module from global dense connectivity as well as efficient feature reuse by parameter sharing from local residual connectivity. Figure 1 illustrates this pattern within the hybrid block schematically. Note that hybrid connectivity pattern exists in the hybrid block which consists of n ($n \geqslant 2$) modules. To match the definition of growth rate in DenseNet, each module produces one feature map with k channels. The basic module consists of successive two cells, which we call them as cell 1 and cell 2, respectively. Globally, input of each module is a concatenation of all feature-maps produced by preceding modules and transferred by dense connectivity. Locally, residual connectivity provides a shortcut that allows the output of cell 1 bypassing cell 2 and then being added to the new features generated by cell 2 to form the output.

Instantiation of Basic Module

To orchestrate our hybrid connectivity, we design a basic SMG module which includes a **Squeeze cell** (cell 1), a **Multi-scale excitation cell** (cell 2) and **Gate mechanisms**. Unless specified otherwise, each convolution is bound a pre-activation, which refers to the three consecutive operations: batch normalization (BN)-rectified linear unit (ReLU)-Conv.

Squeeze Cell. This cell which locates at the beginning of SMG module is responsible for generating the compact feature map from input to improve parameter and computational efficiency for subsequent processing. 1×1 convolution is firstly adopted for changing the number of input channels $\tilde{C}$ to $\lfloor \alpha \cdot C \rfloor$, where $\alpha > 0$ can be reckoned as a width multiplier which is mostly used to reduce the number of channels, i.e. $\tilde{C} > \lfloor \alpha \cdot C \rfloor$, and C denotes the number of final output channels of squeeze cell. Then, 3×3 group convolution (GConv) with g groups proceeds to squeeze the features by reducing the number of channels from $\lfloor \alpha \cdot C \rfloor$ to C, where C needs to be divisible by g. Moreover, it can also play a down-sampling by 3×3 kernel with stride $S = 2$.

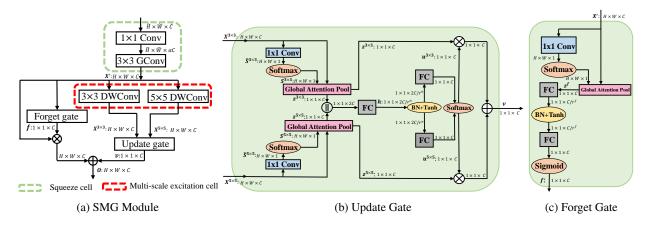(a) SMG Module   (b) Update Gate   (c) Forget Gate

Figure 2: Illustrations of SMG module, update gate and forget gate. In all figures, $\bigoplus$ and $\bigotimes$ denote broadcast element-wise addition and multiplication, respectively. We employ feature dimensions to describe the flow of feature maps for better understanding. Note that spatial size $\tilde{H} \times \tilde{W} = H \times W$ when default stride $S = 1$ of 3×3 GConv in Figure 2a.

Multi-scale Excitation Cell. Squeezed feature map enters this cell for multi-scale excitation by multi-branch convolutions with different kernel sizes. Note that the costs of parameter and computation are extremely cheap because of the few input channels, and the size of feature map throughout this cell is unchanged. To further improve efficiency, we adopt 3×3 and 5×5 depthwise convolutions (DWConv) with 1 and 2 paddings, respectively. Moreover, dilation convolution (Yu and Koltun 2016) with a kernel size of 3×3 and a dilation size of 2 is used to approximate 5×5 kernel for better trade-off between efficiency and performance. The output of this cell is two-branch feature maps produced by 3×3 and 5×5 DWConvs, respectively.

Update Gate. To capture long-range dependency, we utilize update gate to model the global context features from multi-scale information. Figure 2b shows the overall details about the update gate, which can be sequentially summarized for 3 stages: spatial attention, pooling and channel attention.

spatial attention and pooling: We perform a global context modeling for calculating spatial-wise weights of each position. For the given feature map $\mathbf{X}^{3\times3} \in \mathbb{R}^{H \times W \times C}$, a 1×1 convolutional filter shrinks it along channel dimensions to a spatial attention map $\tilde{\mathbf{S}}^{3\times3} \in \mathbb{R}^{H \times W \times 1}$, an then a softmax function normalizes it to obtain the final spatial attention map $\mathbf{S}^{3\times3} \in \mathbb{R}^{H \times W \times 1}$, each element of which is as follows:

$$\mathbf{S}^{3\times3}_{i,j,1} = \frac{e^{\tilde{\mathbf{S}}^{3\times3}_{i,j,1}}}{\sum_{x=1}^{H} \sum_{y=1}^{W} e^{\tilde{\mathbf{S}}^{3\times3}_{x,y,1}}} \tag{5}$$

We employ global attention pooling via weighted averaging with $\mathbf{S}^{3\times3}$ to shrink the global spatial information and generate the global context feature map $\mathbf{z}^{3\times3} \in \mathbb{R}^{1 \times 1 \times C}$. The c-th channel of $\mathbf{z}^{3\times3}$ is as follows:

$$\mathbf{z}^{3\times3}_c = \sum_{x=1}^{H} \sum_{y=1}^{W} \mathbf{X}^{3\times3}_{x,y,c} * \mathbf{S}^{3\times3}_{x,y,c} \tag{6}$$

Here, $*$ denotes element multiplication. Based on the above framework, $\mathbf{z}^{5\times5} \in \mathbb{R}^{1\times1\times C}$ can also be obtained by input feature map $\mathbf{X}^{5\times5} \in \mathbb{R}^{H \times W \times C}$.

channel attention: To maintain the integrity of information, we concatenate $\mathbf{z}^{3\times3}$ and $\mathbf{z}^{5\times5}$ as the input. Then it is transformed to a hidden representation $\mathbf{h} \in \mathbb{R}^{1\times1\times2*C/r^u}$, which is always a compact feature map by setting a reduction ratio r^u for better efficiency. This is achieved by a fully connected (FC) layer with non-linearity:

$$\mathbf{h} = \tanh(BN(\mathbf{W}[\mathbf{z}^{3\times3} \parallel \mathbf{z}^{5\times5}]) + \mathbf{b}) \tag{7}$$

Where BN is the batch normalization, $\mathbf{W} \in \mathbb{R}^{2*C \times 2*C/r^u}$ and $\mathbf{b} \in \mathbb{R}^{2*C/r}$ denotes the weights and biases of FC layer.

It is noteworthy that we adopt $\tanh$ rather than ReLU as our non-linearity function. For the one side, ReLU inevitably destroys feature representational power especially in low-dimensional space to a great extent, while $\tanh$ preserves information by a smoother way. For the other side, although it is widely known that $\tanh$ is more prone to cause gradient vanish as the increasing depth of CNN, this problem could not occur in our HCGNets because of the hybrid connectivity that can significantly strength the gradient back-propagation. Experimental evidence also proves that $\tanh$ is more effective than ReLU in our HCGNets.

Two-branch FC layers act on fusion representation $\mathbf{h}$ to generate two intermediate channel attention maps $\tilde{\mathbf{u}}^{3\times3} \in \mathbb{R}^{1\times1\times C}$ and $\tilde{\mathbf{u}}^{5\times5} \in \mathbb{R}^{1\times1\times C}$:

$$\tilde{\mathbf{u}}^{3\times3} = \mathbf{W}^{3\times3}\mathbf{h} + \mathbf{b}^{3\times3}, \tilde{\mathbf{u}}^{5\times5} = \mathbf{W}^{5\times5}\mathbf{h} + \mathbf{b}^{3\times3} \tag{8}$$

Where $\mathbf{W}^{3\times3}, \mathbf{W}^{5\times5} \in \mathbb{R}^{2*C/r^u \times C}$ and $\mathbf{b}^{3\times3}, \mathbf{b}^{5\times5} \in \mathbb{R}^C$ denotes the weights and biases of two FC layers. Then a simple softmax function conducts a normalization between $\tilde{\mathbf{u}}^{3\times3}$ and $\tilde{\mathbf{u}}^{5\times5}$ to produce the two final channel attention maps $\mathbf{u}^{3\times3} \in \mathbb{R}^{1\times1\times C}$ and $\mathbf{u}^{5\times5} \in \mathbb{R}^{1\times1\times C}$:

$$\mathbf{u}^{3\times3} = \frac{e^{\tilde{\mathbf{u}}^{3\times3}}}{e^{\tilde{\mathbf{u}}^{3\times3}} + e^{\tilde{\mathbf{u}}^{5\times5}}}, \mathbf{u}^{5\times5} = \frac{e^{\tilde{\mathbf{u}}^{5\times5}}}{e^{\tilde{\mathbf{u}}^{3\times3}} + e^{\tilde{\mathbf{u}}^{5\times5}}} \tag{9}$$

Figure 3: A HCGNet with three hybrid blocks, where each green box denotes SMG module.

$\mathbf{u}^{3\times3}$ and $\mathbf{u}^{5\times5}$ can be regarded as the proportions of aggregating multi-scale global context features. Weighted fusion of $\mathbf{z}^{3\times3}$ and $\mathbf{z}^{5\times5}$ is the output of update gate:

$$\mathbf{v}_c = \mathbf{u}_c^{3\times3} \cdot \mathbf{z}_c^{3\times3} + \mathbf{u}_c^{5\times5} \cdot \mathbf{z}_c^{5\times5}, \mathbf{u}_c^{3\times3} + \mathbf{u}_c^{5\times5} = 1 \quad (10)$$

Where $\mathbf{v}_c$ is the c-th channel of the output $\mathbf{v} \in \mathbb{R}^{1\times1\times C}$.

Forget Gate. To decay the reused feature map by channel-wise weights, we locate a forget gate (see Figure 2c) on the residual connection before information fusion. It can also be sequentially summarized for 3 stages: spatial attention, pooling and channel attention.

spatial attention and pooling: For the given feature map $\mathbf{X}' \in \mathbb{R}^{H\times W\times C}$, we perform the global attention pooling as same as update gate, thus a channel descriptor $\mathbf{z}^f \in \mathbb{R}^{1\times1\times C}$ can be obtained.

channel attention: To meet the requirement of weighted decay for each channel, the final output of each channel weight should be within $(0,1)$, thus we refer SE block, which stacks two continuous FC layers as a bottleneck and is ended by sigmoid function. Different from SE block, we insert a batch normalization layer for easing optimization and replace ReLU with $\tanh$ as our non-linearity. In short, the sequent transformations are as follows for the input $\mathbf{z}^f$:

$$\mathbf{f} = \sigma(\mathbf{W}_2^f(\tanh(BN(\mathbf{W}_1^f\mathbf{z}^f + \mathbf{b}_1^f))) + \mathbf{b}_2^f) \quad (11)$$

Where σ is the sigmoid function, $\mathbf{W}_1^f \in \mathbb{R}^{C\times C/r^f}$, $\mathbf{b}_1^f \in \mathbb{R}^{C/r^f}$, $\mathbf{W}_2^f \in \mathbb{R}^{C/r^f\times C}$ and $\mathbf{b}_2^f \in \mathbb{R}^C$. r^f is the bottleneck ratio and $\mathbf{f} \in \mathbb{R}^{1\times1\times C}$ is the final channel attention map.

Information Fusion. For any given feature map entering SMG module, squeeze cell firstly condenses it to a compact feature map denoted by $\mathbf{X}'$. Then $\mathbf{X}'$ enters multi-scale excitation cell and generate two-branch outputs $\mathbf{X}^{3\times3}$ and $\mathbf{X}^{5\times5}$ by 3×3 and 5×5 DWConvs, respectively. Since then, $\mathbf{X}'$ can be regarded as the reused features, while $\mathbf{X}^{3\times3}$ and $\mathbf{X}^{5\times5}$ are the newly extracted features. An update gate integrates $\mathbf{X}^{3\times3}$ and $\mathbf{X}^{5\times5}$ to model a global context feature map $\mathbf{v}$ and we aggregate it to the decayed $\mathbf{X}'$ of each spatial position by addition to build the final output $\mathbf{O} \in \mathbb{R}^{H\times W\times C}$. It can be observed that we maintain the magnitude of new features unchanged while decaying reused features, which can facilitate the effective feature exploration and retain the capability of feature re-exploitation to some extent.

Macro-architecture.

As shown in Figure 3, at the beginning of HCGNet is a stem, which is a composite function to process the initial input images. Then multiple hybrid blocks are stacked with various spatial stage. Between two adjacent hybrid blocks, we adopt

Table 1: HCGNet-B network architecture for ImageNet classification. Each row describes the stage, modules information and input resolution (IR).

Stage	Module	IR
Stem	[3×3 Conv-BN-ReLU]×3	224×224
	3×3 max pool	112×112
Hybrid Block	SMG×3 ($k=32$)	56×56
Transition	SMG×1	56×56
Hybrid Block	SMG×6 ($k=48$)	28×28
Transition	SMG×1	28×28
Hybrid Block	SMG×12 ($k=64$)	14×14
Transition	SMG×1	14×14
Hybrid Block	SMG×8 ($k=96$)	7×7
Classification	global average pool	1×1
	1000D FC, softmax	-

a transition layer to perform down-sampling and connectivity truncation. After the final hybrid block, a global average pooling attached with a softmax classifier calculates the probabilities of various categories.

Both hybrid block and transition layer adopt SMG modules but with different hyperparameter settings. We only stack one SMG module to build each transition layer and a compression factor $\theta = 0.5$ is utilized to reduce the number of channels, i.e, $C = \theta\tilde{C}$. For each SMG module, we set $g = 4$, $\alpha = 4$ and $r^u = r^f = 2$ in hybrid blocks, as well as set $g = 1$, $\alpha = 1.5$, $S = 2$ and $r^u = r^f = 4$ in transition layers. Note that we apply the standard convolutions in transition layers for best capability of feature extraction and group convolutions in hybrid blocks for better trade-off between efficiency and performance. Compared with the hybrid block, we set less multiplier α and larger reduction ratios r^u and r^f for better efficiency due to the more channels of feature maps in transition layers.

Specifically, we construct several networks to act on the image classification across the CIFAR and ImageNet datasets. For CIFAR, we adopt a 3×3 standard convolution with stride 1 as the stem that the number of output channels is twice the growth rate of the first hybrid block. And we build three networks with various model specifications: HCGNet-(8,8,8)-(k=12,24,36)(A1), HCGNet-(8,8,8)-(k=24,36,64)(A2) and HCGNet-(12,12,12)-(k=36,48,80)(A3). Formally, the first m-tuple indicates that there are m hybrid blocks, where each figure denotes the number of SMG modules in the corresponding hybrid block. The second m-tuple denotes m growth rates of m hybrid blocks, respectively. For ImageNet, the stem consists of three contiguous 3×3 Conv-BN-ReLU layers (stride 2 for the first layer) with 32, 32, 64 output channels, and attached by a 3×3 max pooling with stride 2. We construct two networks: HCGNet-(3,6,12,8)-(k=32,48,64,96)(B, as Table 1) and HCGNet-(6,12,18,14)-(k=48,56,72,112)(C).

Table 2: Comparisons of our HCGNets against state-of-the-art networks about test error rates (%) across CIFAR-10 and CIFAR-100 datasets. Note that the first and second blocks contain human-designed and auto-searched architectures, respectively.

Model	Params	FLOPs	C-10	C-100
CondenseNet-182	4.2M	0.5G	3.76	18.47
SparseNet-BC	16.7M	-	4.10	18.22
AOGNet	24.8M	3.7G	3.27	16.63
LogDenseNetV2	19.0M	11.1G	3.75	18.80
Wide ResNet-28	36.5M	5.2G	4.17	20.50
ResNeXt-29+SK	27.7M	-	3.47	17.33
Res2NeXt-29	36.9M	-	-	16.56
DenseNet-BC-190	25.6M	9.4G	3.46	17.18
DPN-28-10	47.8M	-	3.65	20.23
MixNet-190	48.5M	17.3G	3.13	16.96
PNASNet	3.2M	-	3.41	19.53
NASNet-A	3.3M	-	3.41	19.70
ENASNet	4.6M	-	3.54	19.43
AmoebaNet-A	4.6M	-	3.34	-
AmoebaNet-B	34.9M	-	2.98	17.66
NASNet-A	50.9M	-	-	16.03
ENASNet	52.7M	-	-	16.44
PNASNet	53.0M	-	-	16.70
HCGNet-A1	1.1M	0.2G	3.15	18.13
HCGNet-A2	3.1M	0.5G	2.29	16.54
HCGNet-A3	11.4M	2.0G	**2.14**	**15.96**

Table 3: Comparisons of our HCGNets against SOTA networks about Top-1 and Top-5 error rates (%) on ImageNet.

Model	Params	FLOPs	T-1	T-5
MixNet-105	11.2M	5.0G	23.3	6.7
MixNet-121	21.9M	8.3G	21.9	5.9
MixNet-141	41.1M	13.1G	20.4	5.3
DPN-68	12.8M	2.5G	23.6	6.9
DPN-92	38.0M	6.5G	20.7	5.4
DPN-98	61.6M	11.7G	20.2	5.2
DenseNet-169	14.2M	3.5G	23.8	6.9
DenseNet-201	20.0M	4.4G	22.6	6.3
DenseNet-264	33.4M	6.0G	22.2	6.1
SparseNet-201	14.9M	9.2G	22.7	-
ResNet-50	25.6M	3.9G	24.6	7.5
ResNet-50+SE	28.1M	3.9G	23.1	6.7
ResNet-50+CBAM	28.1M	3.9G	22.7	6.3
ResNet-101	44.6M	7.6G	23.4	6.9
ResNet-101+SE	49.3M	7.6G	22.4	6.2
ResNet-101+CBAM	49.3M	7.6G	21.5	5.7
ResNeXt-50	25.0M	3.8G	22.9	6.5
ResNeXt-50+SE	27.6M	3.8G	21.9	6.0
ResNeXt-101	44.2M	7.5G	21.5	5.8
ResNeXt-101+SE	49.0M	7.5G	21.2	5.7
ResNeXt-101+SK	48.9M	8.5G	20.2	-
WideResNet-18	45.6M	6.7G	25.6	8.2
WideResNet-18+SE	46.0M	6.7G	24.9	7.7
AOGNet-12M	11.9M	2.4G	22.3	6.1
AOGNet-40M	40.3M	8.9G	19.8	4.9
HCGNet-B	12.9M	2.0G	21.5	5.8
HCGNet-C	42.2M	7.1G	**19.5**	**4.8**

Experiments

Experiments on CIFAR

Dataset and training details. Both CIFAR-10 and CIFAR-100 datasets comprise 50k training images and 10k test images corresponding to 10 and 100 classes, respectively. We apply a standard data augmentation following Huang et al. (2017). We employ a stochastic gradient descent (SGD) optimizer with momentum 0.9 and batch size 128. Training is regularized by weight decay 1×10^{-4} and mixup with $\alpha = 1$ (Zhang et al. 2017). For HCGNet-A1, we train it for 1270 epochs by SGDR (Loshchilov and Hutter 2016) learning rate curve with initial learning rate 0.1, $T_0 = 10, T_{mul} = 2$. For HCGNet-A2 and A3, we train them for 1260 epochs including two continuous 630 epochs, each of them is a SGDR learning rate curve with initial learning rate 0.1, $T_0 = 10, T_{mul} = 2$.

Comparisons with Human-designed Networks. Quantitatively in Table 2, DenseNet-190 has 31 modules in each dense block, while HCGNet-A2 only has 8 modules in each hybrid block thus reduces 93% redundancy but with substantial accuracy gains. Moreover, HCGNet-A2 significantly outperforms other sparsification variants, such as LogDenseNet, SparseNet and CondenseNet, which indicate that our optimization of DenseNet is more effective than sparsification method. HCGNet-A2 using $16 \times$ fewer parameters surpasses MixNet-190, which represents the most general form of ResNet and DenseNet. It also uses $8 \times$ fewer parameters but obtains better results than concurrent AOGNet, which is the state-of-the-art human network by hierarchical and compositional feature aggregation. Consequently, our nested aggregation is the best method among other combinations and variants of ResNet and DenseNet.

Comparisons with auto-searched Networks. Notably, Our HCGNets are also more efficient than auto-searched networks. Compared with other networks with small setting, HCGNet-A2 achieves around 1% and 3% reductions on CIFAR-10 and CIFAR-100 error rates, respectively. Moreover, it is noteworthy that HCGNet-A1 can also obtain superior performance with unprecedent efficiency. For large setting, HCGNet-A3 achieves the best results with least complexity. Somewhat surprisingly, HCGNet-A3 can outperform the most competitive NASNet-A with only 22% parameters.

Experiments on ImageNet 2012

Dataset and Training Details. ImageNet 2012 dataset comprises 1.2 million training images and 50k validation

Table 4: Comparisons of HCGNet-B against other backbones on the Mask-RCNN system (He et al. 2017).

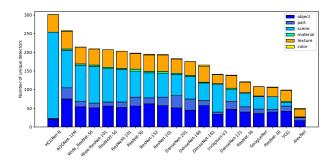| Backbone	Params	FLOPs	$AP^{bb}_{50:95}$	AP^{bb}_{50}	AP^{bb}_{75}	$AP^{m}_{50:95}$	AP^{m}_{50}	AP^{m}_{75}
ResNet-50-FPN	44.2M	275.6G	37.3	59.9	40.2	34.2	55.9	36.2
AOGNet-12M-FPN	31.2M	-	38.0	59.8	**41.3**	34.6	56.6	36.4
HCGNet-B-FPN	32.1M	230.4G	**38.3**	**60.6**	41.3	**35.2**	**57.5**	**37.1**

Figure 4: Comparisons of interpretability by network dissection (Bau et al. 2017) among popular models based on ImageNet pretrained models.

Figure 5: Comparisons of adversarial robustness by FGSM (Goodfellow, Shlens, and Szegedy 2015) attack.

images corresponding to 1000 classes. We employ the data augmentation following Huang et al. (2017). Final error rates are reported by single-crop with size 224×224 at test time on the validation set. We employ synchronous SGD with momentum 0.9 and batch size 256. Training is regularized by weight decay 4×10^{-5}, label smoothing with $\epsilon = 0.1$ (Szegedy et al. 2016), mixup with $\alpha = 0.4$ and dropout (Srivastava et al. 2014) with rate 0.1 before the final FC layer. All networks are trained for 630 epochs by SGDR learning rate curve with initial learning rate 0.1, $T_0 = 10$, $T_{mul} = 2$.

Comparisons with popular networks. As shown in Table 3, our HCGNets perform the best among all other models with less or comparable complexity in terms of top-1 and top-5 error rates. It is noteworthy that DenseNet-169 stacks 4 dense blocks with 6,12,32,32 modules, while HCGNet-B utilizes shallower design with 3,6,12,8 modules for 4 hybrid blocks, thus reducing 88% redundancy but obtaining above absolute 2.3% gain of performance. Furthermore, HCGNets yield significantly better results than the families of DenseNet, MixNet and DPN under comparable complexity. Remarkably, using considerable $4.6\times$ fewer FLOPs, HCGNet-B can also surpass SparseNet-201, which is the state-of-the-art variant of DenseNet. The family of HCGNet can consistently obtain better performance than the families of ResNet, ResNeXt, WideResNet and their attention-based variants, which represent the widely applied models in practice. Predominately, HCGNets outperform previous SOTA AOGNets across various model specifications, which show the superiority of our design.

Model Interpretability We quantify the interpretability by network dissection, which compares the number of unique detectors in the final convolutional layer. Figure 4 shows that HCGNet-B obtains the overall highest score with
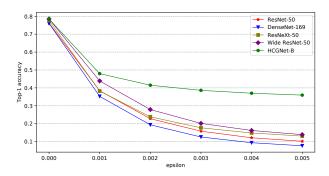
least complexity, which shows that the designs of hybrid connectivity and SMG module can not only achieve the best accuracy, but also generate the best latent representations.

Adversarial Robustness We attack HCGNet-B by popular FGSM across various perturbation energies ϵ to test the adversarial robustness against widely applied models, results of which are shown in Figure 5. HCGNet-B has a more remarkable robustness than other models in adversarial defense, especially the perturbation is relatively high.

Object Detection and Instance Segmentation

To show the transferability, we experiment HCGNet-B pretrained on ImageNet as a backbone on the Mask-RCNN system to implement object detection and instance segmentation tasks. We use COCO train2017 set to finetune the HCGNet-B by the 1x training schedule, and evaluate the performance on COCO val2017 set. We report the results by standard COCO metrics of Average Precision (AP), i.e, $AP_{50:95}$, AP_{50}, and AP_{75} for bounding box detection (AP^{bb}) and instance segmentation (AP^m) in Table 4. The results show that HCGNet-B can learn better features than SOTA ResNet and AOGNet backbones.

Conclusion.

This paper develops an efficient architecture with the innovative designs of hybrid connectivity, micro-module and attention-based forget and update gates. On CIFAR and ImageNet datasets, HCGNets outperform state-of-the-art networks with less or comparable complexity. Extensive experiments based on the ImageNet pretrained model further show the remarkable interpretability, robustness for recognition and transferability for detection. We hope our HCGNets may inspire the future study of architectural design and search.

References

Bau, D.; Zhou, B.; Khosla, A.; Oliva, A.; and Torralba, A. 2017. Network dissection: Quantifying interpretability of deep visual representations. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 6541–6549.

Cao, Y.; Xu, J.; Lin, S.; Wei, F.; and Hu, H. 2019. Gcnet: Non-local networks meet squeeze-excitation networks and beyond. *arXiv preprint arXiv:1904.11492*.

Chen, Y.; Li, J.; Xiao, H.; Jin, X.; Yan, S.; and Feng, J. 2017. Dual path networks. In *Advances in Neural Information Processing Systems*, 4467–4475.

Deng, J.; Dong, W.; Socher, R.; Li, L.-J.; Li, K.; and Fei-Fei, L. 2009. Imagenet: A large-scale hierarchical image database. In *IEEE conference on computer vision and pattern recognition*, 248–255.

Gao, S.-H.; Cheng, M.-M.; Zhao, K.; Zhang, X.-Y.; Yang, M.-H.; and Torr, P. 2019. Res2net: A new multi-scale backbone architecture. *arXiv preprint arXiv:1904.01169*.

Goodfellow, I. J.; Shlens, J.; and Szegedy, C. 2015. Explaining and harnessing adversarial examples. In *3rd International Conference on Learning Representations*.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 770–778.

He, K.; Gkioxari, G.; Dollár, P.; and Girshick, R. 2017. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, 2961–2969.

Hu, H.; Dey, D.; Del Giorno, A.; Hebert, M.; and Bagnell, J. A. 2017. Log-densenet: How to sparsify a densenet. *arXiv preprint arXiv:1711.00002*.

Hu, J.; Shen, L.; and Sun, G. 2018. Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 7132–7141.

Huang, G.; Liu, Z.; Van Der Maaten, L.; and Weinberger, K. Q. 2017. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 4700–4708.

Huang, G.; Liu, S.; Van der Maaten, L.; and Weinberger, K. Q. 2018. Condensenet: An efficient densenet using learned group convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2752–2761.

Krizhevsky, A., and Hinton, G. 2009. Learning multiple layers of features from tiny images. Technical report, Citeseer.

Li, X.; Wang, W.; Hu, X.; and Yang, J. 2019. Selective kernel networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*.

Li, X.; Song, X.; and Wu, T. 2019. Aognets: Compositional grammatical architectures for deep learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*.

Liu, C.; Zoph, B.; Neumann, M.; Shlens, J.; Hua, W.; Li, L.-J.; Fei-Fei, L.; Yuille, A.; Huang, J.; and Murphy, K. 2018.

Progressive neural architecture search. In *Proceedings of the European Conference on Computer Vision*, 19–34.

Loshchilov, I., and Hutter, F. 2016. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*.

Real, E.; Aggarwal, A.; Huang, Y.; and Le, Q. V. 2019. Regularized evolution for image classifier architecture search. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, 4780–4789.

Srivastava, N.; Hinton, G.; Krizhevsky, A.; Sutskever, I.; and Salakhutdinov, R. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research* 15(1):1929–1958.

Szegedy, C.; Vanhoucke, V.; Ioffe, S.; Shlens, J.; and Wojna, Z. 2016. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2818–2826.

Szegedy, C.; Ioffe, S.; Vanhoucke, V.; and Alemi, A. A. 2017. Inception-v4, inception-resnet and the impact of residual connections on learning. In *Thirty-First AAAI Conference on Artificial Intelligence*, 4278–4284.

Wang, F.; Jiang, M.; Qian, C.; Yang, S.; Li, C.; Zhang, H.; Wang, X.; and Tang, X. 2017. Residual attention network for image classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 3156–3164.

Wang, W.; Li, X.; Yang, J.; and Lu, T. 2018a. Mixed link networks. *arXiv preprint arXiv:1802.01808*.

Wang, X.; Girshick, R.; Gupta, A.; and He, K. 2018b. Non-local neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 7794–7803.

Woo, S.; Park, J.; Lee, J.-Y.; and So Kweon, I. 2018. Cbam: Convolutional block attention module. In *Proceedings of the European Conference on Computer Vision*, 3–19.

Xie, S.; Girshick, R.; Dollár, P.; Tu, Z.; and He, K. 2017. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 1492–1500.

Yu, F., and Koltun, V. 2016. Multi-scale context aggregation by dilated convolutions. In *International Conference on Learning Representations*.

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

Zhang, H.; Cisse, M.; Dauphin, Y. N.; and Lopez-Paz, D. 2017. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*.

Zhu, L.; Deng, R.; Maire, M.; Deng, Z.; Mori, G.; and Tan, P. 2018. Sparsely aggregated convolutional networks. In *Proceedings of the European Conference on Computer Vision*, 186–201.

Zoph, B.; Vasudevan, V.; Shlens, J.; and Le, Q. V. 2018. Learning transferable architectures for scalable image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 8697–8710.