When Residual Learning Meets Dense Aggregation: Rethinking the Aggregation of Deep Neural Networks

Zhiyu Zhu[1], **Zhen-Peng Bian**[2], **Junhui Hou**[1,*], **Yi Wang**[2] and **Lap-Pui Chau**[2]
[1]City University of Hong Kong
[2]Nanyang Technological University
zhiyuzhu2-c@my.cityu.edu.hk, {zbian1,Yi Wang}@e.ntu.edu.sg, jh.hou@cityu.edu.hk, elpchau@ntu.edu.sg

Abstract

Various architectures (such as GoogLeNets, ResNets, and DenseNets) have been proposed. However, the existing networks usually suffer from either redundancy of convolutional layers or insufficient utilization of parameters. To handle these challenging issues, we propose Micro-Dense Nets, a novel architecture with global residual learning and local micro-dense aggregations. Specifically, residual learning aims to efficiently retrieve features from different convolutional blocks, while the micro-dense aggregation is able to enhance each block and avoid redundancy of convolutional layers by lessening residual aggregations. Moreover, the proposed micro-dense architecture has two characteristics: pyramidal multi-level feature learning which can widen the deeper layer in a block progressively, and dimension cardinality adaptive convolution which can balance each layer using linearly increasing dimension cardinality. The experimental results over three datasets (i.e., CIFAR-10, CIFAR-100, and ImageNet-1K) demonstrate that the proposed Micro-Dense Net with only 4M parameters can achieve higher classification accuracy than state-of-the-art networks, while being 12.1× smaller depends on the number of parameters. In addition, our micro-dense block can be integrated with neural architecture search based models to boost their performance, validating the advantage of our architecture. We believe our design and findings will be beneficial to the DNN community.

1 Introduction

Owing to the great representation ability, deep neural networks (DNNs) have achieved great success in machine learning and computer vision communities [Gu *et al.*, 2018; Rawat and Wang, 2017]. [1] Since Alexnet [Krizhevsky *et al.*, 2012], a plain network that contains stacked convolutional layers, design of DNNs has become a hot research topic. Nowadays, the neural network architecture is going towards deeper and deeper [Simonyan and Zisserman, 2014]. However, as the depth of DNNs scales up, problems such as gradient vanishing and overfitting appear and make them difficult to train [Glorot and Bengio, 2010]. To address these problems, DNNs with different topologies have been proposed, e.g., highway networks [Srivastava *et al.*, 2015] with shortcuts followed by ResNets [He *et al.*, 2016] using identity mappings and DenseNets [Huang *et al.*, 2017] with dense concatenation. Meanwhile, some training techniques e.g., batch/group normalization [Ioffe and Szegedy, 2015; Wu and He, 2018] and drop-out layers [Srivastava *et al.*, 2014], were also proposed to deal with the optimization of DNNs. Although training very deep neural networks becomes easier with the help of shortcuts and training techniques, the performance of DNNs does not gain much from that extreme depth. [He *et al.*, 2016]

Instead of increasing the depth, GoogLeNets [Szegedy *et al.*, 2015] enlarge the width (i.e., the number of channels) of the DNN by using parallel convolutional layers with different kernel sizes. ResNeXts [Xie *et al.*, 2017] try to use cardinality to extract deep features efficiently. And

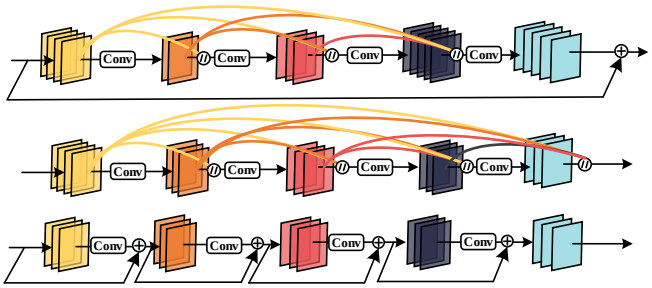

Figure 1: Qualitative comparisons of the proposed network architecture with the well-known DenseNets and ResNets. ***Top***: Our architecture, miro-dense aggregation with global residual learning. ***Middle***: Dense aggregation of DenseNets, ***Bottom***: residual learning of ResNets. Our proposed network architecture uses local dense and global identity mapping aggregations. Moreover, to make full use of parameters, the proposed micro-dense structure has two characteristics: (1) **pyramidal multi-level feature learning**, i.e., gradually increase the output channel of densely connected layers; and (2) **dimension cardinality adaptive convolution**, i.e., rise the groups of convolutional layers with the growth of width. '+' denotes the addition operation and '//' denotes the concatenation operation.

[1]Z.Zhu and Z.-P. Bian contributed to this work equally. *Corresponding author: Junhui Hou*

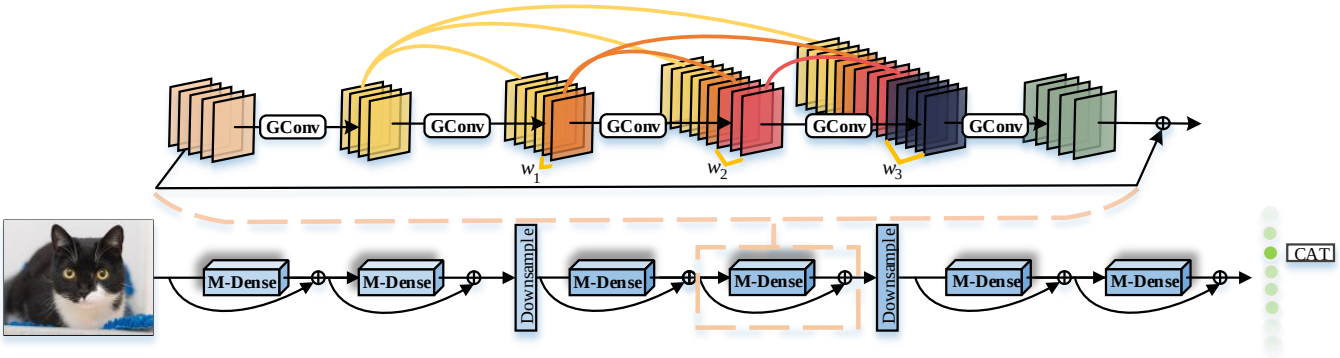

Figure 2: Illustration of our Micro-Dense Nets and micro-dense aggregations (noted as M-Dense in the figure), where $w_1 < w_2 < w_3$ indicates that both the output channels and dimension cardinalities are increasing while convolutional layer is going deeper, and GConv represents a block of GroupedConvolution-BatchNormalization-ReLU.

Res2Nets [Gao *et al.*, 2020] use features from different levels to augment the extraction ability. All of them share the same conception of using multiple branches rather than single convolutional layers in residual learning.

DenseNets [Huang *et al.*, 2017] reveal that the redundancy of convolutional layers might be the reason why the improvement stops while ResNets go deeper. Hence, the dense aggregation was proposed to handle the problem. The importance of feature reuse was also demonstrated. However, dense aggregation consumes plenty of resources on extracting accumulated features from all proceeding layers. SparseNets [Zhu *et al.*, 2018] presents a sparse connection to reduce the heavy burden of the dense aggregation. Moreover, some works try to fuse the ResNets and DenseNets together, e.g., Dualpath networks [Chen *et al.*, 2017], residual dense blocks [Zhang *et al.*, 2018b], and Mixed Link Networks (MixNets) [Wang *et al.*, 2018]. All these methods aim to make full use of residual learning and dense aggregation.

Our Solution. In this paper, by comprehensively analyzing the different types of aggregations in existing DNNs, we propose Micro-Dense Nets, which learns features in locally dense and globally residual manners. As shown in Figure 2, Micro-Dense Nets consist of several sub-convolutional blocks named micro-dense blocks, which densely connects local layers inside the block to extract features. Meanwhile, identity mapping is adopted to connect both ends of the micro-dense block. It inherits the advantages of multi-level feature learning and efficient feature storage from dense aggregations and residual learning, respectively. Moreover, the proposed micro-dense block possesses the following two characteristics: 1) pyramidal multi-level feature learning, i.e., subsequent layers are wider than preceding ones in a micro-dense block to avoid the loss of information; and 2) dimension cardinality adaptive convolution, i.e., in the proposed micro-dense block, the number of groups of convolutional layers gradually increases with the dimension of feature-maps to avoid parameter explosion, which will be aroused by the pyramidal multi-level feature learning strategy.

Our Findings. Through comprehensive experiments on ImageNet-1K and CIFAR, the proposed Micro-Dense Nets exceed state-of-the-art methods and reach 97.84% and 84.64% on CIFAR-10 and CIFAR-100, while being at most 12.1 $\times$ smaller than state-of-the-art DNNs. Moreover, we integrate our Micro-Dense Nets with the state-of-the-art neural architecture search (NAS) based model and obtain 0.5% gain in terms of the Top 1. classification accuracy on ImageNet-1K, which validates the advantage of our architecture. Last but not least, based on the experimental results, we can draw the following conclusions: *1) dense aggregation indeed boosts the feature extraction ability of DNNs, but very deep dense aggregation may degrade the performance of DNNs with the same number of parameters; and 2) although conventional multi-level features with a fixed growth rate are beneficial to DNNs, the multi-level feature learning with a varying growth rate, i.e., our pyramidal multi-level feature learning strategy, will be a better choice. Accordingly, a well designed method for avoiding parameter explosion should accompany. We believe our design and findings will be beneficial to the DNN community.*

2 Revisiting Aggregated DNNs

Aggregations play a critical role in DNNs. They could strengthen the feature extraction ability of DNNs, e.g., shortcuts existed in ResNets and dense aggregations in DenseNets. However, inappropriate introduction of aggregations may be harmful to DNNs. In the following, we will analyze the four types of aggregations adopted by existing DNNs comprehensively.

2.1 Aggregations in DNNs

Aggregations adopted in existing DNNs can be roughly classified into four categories, i.e., plain, highway connection, dense aggregation and inception. To use both features and parameters efficiently, combining diverse aggregations in multiple levels is a promising solution. We discuss such a combination with detailed analysis. In the following, we denote $\mathcal{H}(\cdot)$ as the composite function of operations including Convolution (Conv), Pooling, Batch Normalization (BN), and Recitified Linear Units (ReLU).

Plain. As a basic connection in DNNs, the plain aggregation means to simply stack different layers together. Therefore the output can be represented as

$$x^k = \mathcal{H}_{k-1}(x^{k-1}), \tag{1}$$

where x^k mean the output feature-maps of k-th layer. Limited by the simple topology, plain connections have a weak feature learning ability. Considering the fact that the topology of DNNs helps to enhance representation learning [Chen *et al.*, 2017], such a simple plain connection may even degrade the feature learning ability of networks.

Highway connection. The hghway connection [Srivastava *et al.*, 2015] expressed as:

$$x^k = x^{k-1} \times R_{k-1}(x^{k-1}) + \mathcal{H}_{k-1}(x^{k-1}) \times T_{k-1}(x^{k-1}), \quad (2)$$

where R_{k-1} and T_{k-1} are two nonlinear transforms named carry gate and transform gate, respectively. ResNets adopt identity mapping where $R_{k-1}(x^{k-1})$ and $T_{k-1}(x^{k-1})$ are set to an identity function leading to:

$$x^k = x^{k-1} + \mathcal{H}_{k-1}(x^{k-1}). \quad (3)$$

In residual learning, each layer is fed by the sum of multiple output feature-maps of preceding layers. Different from the plain network, identity mapping in ResNets could not only help the feature propagation but also alleviate the gradient vanishing problem.

Dense aggregation. Dense aggregation preserves all the output feature-maps in preceding layers. And the output of dense aggregation is outlined as:

$$x^k = \mathcal{H}_{k-1}([x^1, x^2, ..., x^{k-1}]), \quad (4)$$

where $[x^1, x^2, ..., x^{k-1}]$ refers to the concatenation of the feature-maps produced by preceding $1, 2, ..., k-1$ layers. Rather than the hard-code identity mapping in residual learning, dense aggregation learns to weighted-sum different level features. Due to the flexibility of multi-level feature learning, dense aggregations boost the feature learning ability of DNNs. Nonetheless, accumulated feature-maps are heavy burdens of Dense-aggregations which restrict effective parameter using.

Inception. The inception block uses several plain convolutional blocks to extract features separately and then concatenates them. Inception aggregation can be expressed as:

$$x^k = [\mathcal{H}^1_{k-1}(x^{k-1}), \mathcal{H}^2_{k-1}(x^{k-1}), ..., \mathcal{H}^j_{k-1}(x^{k-1})]. \quad (5)$$

Inception aggregations widen DNNs and could also combine features in different levels. However, in deep inception aggregation, each branch extract features separately, and convolutional layers could only percepte feature in one branch. Thus, deepening inception aggregation reduces the richness of features and may degenerate the performance of DNNs.

2.2 Assembling of Different Aggregations

Since network in network [Lin *et al.*, 2013] was proposed, the research on combining different kinds of aggregation in DNNs has never stopped. [Wang *et al.*, 2018] found that both ResNets and DenseNets come from the same dense topology. However, they varied in the way of aggregations. DualpathNets [Chen *et al.*, 2017] have both addition and concatenation connections, and it could save the different level features and realize residual feature learning.

Identity mapping in residual learning greatly boosts the feature learning ability of DNNs. However, as the residual learning based networks get deeper, frequently residual feature additions cause the redundancy of convolutional layers [Huang *et al.*, 2017], which will degrade its performance. Res2Nets indicate that instead of increasing the number of residual blocks in residual learning-based architectures, enhancing feature extraction ability of each block can be a promising solution. However, the convolutional layers of Res2Nets only reuse features from two levels, leading to a weaker representation ability when compared with the dense aggregation in DenseNets which could utilize features in multiple levels. Unfortunately, the dense aggregations occupy too many resources for feature-maps storage and fusion. Also widening or deepening a dense block will cause high resource consumption [Zhu *et al.*, 2018]. Additionally, a critical point neglected by most dense aggregations based methods [Huang *et al.*, 2017; Zhu *et al.*, 2018; Chen *et al.*, 2017; Wang *et al.*, 2018] is that all the dense aggregation based methods adopt a fixed channel growth rate for dense layers. Due to feature accumulation, the number of input channels, which restricts the maximum information, increases while the dense layer deepens. Thus, the fixed rate may result in losses of information. Notwithstanding this drawback, intuitively increasing the growth ratio on layer-wise is impossible for dense aggregation based methods. The reason is that it will lead to both features and parameter explosion. That is, the number of input channels (the number of parameters) grows from C_n to $\hat{C}_n$:

$$C_n = c_0 + \sum_{l=1}^{n} c_i = c_0 + n \times r_0, \quad (6)$$

$$c_i = r_0 \quad (7)$$

$$\hat{C}_n = c_0 + \textstyle\sum_{l=1}^{n} \hat{c}_i = c_0 + n \times r_0 + \hat{r} \times (n-1) \times n/2, \quad (8)$$

$$\hat{c}_i = r_0 + \hat{r} \times (i-1), \quad (9)$$

where C_n is the channel number of the fixed growth-rate r_0 and n preceding layers and $\hat{C}_n$ represents the channel number with the growth rate gradually increasing ($\hat{r}$ is increasing factor of growth rate r_0).

Remark. From the above analysis, we can conclude that: 1) there are only two operators for feature fusion: addition ('$+$' represented by residual learning) and concatenation ('[..]' represented by DenseNets). The former one could efficiently preserve features from multiple levels without consuming extra space, while the latter one reuses all output features from preceding layers to strengthen the richness of features; and 2) although multi-level feature reuse could enhance the feature learning ability of DNNs, high level features extracted by multiple layers have stronger representation abilities. In order to enhance feature extraction ability of DNNs, we should scale them up. However, ResNets are limited by the redundancy of convolutional layers and DenseNets consume too much on feature-maps reduction. Meanwhile, DenseNets fix the number of output channels which restricts the high level features.*Based on these analyses, we propose a new architecture , which can inherit the advantages of the existing DNNs well and avoid the their drawbacks such that performance is boosted.*

Layer		Input channel c_i	Output channel c_u	Cardinality
Compression	BatchNorm	c_i	c_i	-
	C1-BR	c_i	$c_0 = \lfloor c_i \times r_a \rfloor \times g_c$	-
Dense-1	C1-BR	$\hat{c}_1 = c_0$	$c_1 = c_0/g_c \times k_1$	-
	C3-BR	c_1	c_1	k_1
Dense-2	C1-BR	$\hat{c}_2 = c_0 + c_1$	$c_2 = c_0/g_c \times k_2$	-
	C3-BR	c_2	c_2	k_2
Dense-3	C1-BR	$\hat{c}_3 = c_0 + c_1 + c_2$	$c_3 = c_0/g_c \times k_3$	-
	C3-BR	c_3	c_3	k_3
Dense-n	C1-BR	$\hat{c}_n = \sum_{i=0}^{n-1} c_i$	$c_n = c_n/g_c \times k_n$	-
	C3-BR	c_n	c_n	k_n
Output Layer	C1-BR	$c_{n+1} = \sum_{i=0}^{n} c_i$	c_u	-

Table 1: The detailed architecture of a micro-dense block with n dense layers. Cs-BR indicates a stack of Convolution-BatchNorm-Relu layer with kernel of size $s \times s$. Cardinality is the number of groups and g_c is set to be 4. The micro-dense block with n dense layers is denoted as *MDConv-n*.

3 The Proposed Micro-Dense Nets

As shown in the bottom of Figure 2, our Micro-Dense Nets consist of multiple convolutional blocks named micro-dense block, which uses dense aggregations to link local layers shown in the top of Figure 2. Meanwhile, in order to strengthen feature propagation, we use identity mapping to connect both ends of a micro-dense block. In what follows, more details will be provided about the proposed Micro-Dense Nets.

3.1 Local Micro-Dense Architecture

The proposed micro-dense block is constructed by several sub-bottleneck structures. Different from the conventional dense layer, it has following two characteristics:

Pyramidal multi-level feature learning. In each micro-dense block, convolutional layers learn multi-level features with the output feature dimension gradually increasing. Although the conventional dense aggregation [Huang *et al.*, 2017] with a fixed growth rate boosts the performance of DNNs, the inconsistency between the input/output feature dimensions may degrade its representation ability. Moreover, to the best of our knowledge, all dense aggregations based architectures [Zhu *et al.*, 2018; Chen *et al.*, 2017; Wang *et al.*, 2018; Zhang *et al.*, 2018b] took it for granted that the number of output channels is fixed. In the proposed micro-dense block, this problem could be relieved, as each micro-dense block is quite shallower than the full-dense aggregation. Similar to [Han *et al.*, 2017], we increase the width of dense layers inside a dense block linearly, i.e.,

$$c_l = (l+1) \times c_0, \tag{10}$$

where c_l is the width of the l-th dense layer in a micro-dense block. Aggregating the dense layers locally inside a micro-block is able to relieve the feature explosion. Because dense aggregation only has to preserve features in one block. Nevertheless, such a linear manner also leads to parameter explosion. Since the representation learning ability of a single convolutional layer is controlled by the number of parameters, each layer should have a similar number of parameters inside a block. To this end, we introduce dimension cardinality adaptive convolution to achieve efficient parameter utilization.

Dimension cardinality adaptive convolution. Dimension cardinality indicates the number of groups in convolutional layers. Unlike the conventional convolution, it divides the input into several groups and convolves with several shallow kernels separately on these feature-maps. Thus, it could reduce the number of parameters in a single layer. As shown in Table 1, the number of groups grows up while the convolutional layers are deepening, which will greatly reduce the number of parameters. Specifically, the process is expressed as:

$$P_l = c_l^u \times (c_l^i \times w_l \times h_l + 1), \tag{11}$$

$$P_l^g = G_l \times (c_l^i/G_l) \times ((c_l^u/G_l) \times w_l \times h_l + 1) \approx P_l/G_l, \tag{12}$$

where (P_l^g / P_l) indicates the number of parameters in the convolutional layer (with / without) groups, c_i and c_u are the input and output channels, and w_k and h_k represent the width and height of the corresponding kernel. According to Eq.11, if the channels c_i and c_u grow linearly with respect to l (l is the depth of dense layers shown in Eq.10), the number of parameters grows quadratically with respect to l^2. By increasing G^k linearly as $\beta \times l$, we could alleviate the problem of parameter explosion (from quardractic to linear). In a micro-dense block, the cardinality grows linearly with

$$k_n = n + 1. \tag{13}$$

Furthermore, the group convolution requires both input and output channels to be dvisible by the group number. Thus, we design the bottleneck structure for each layer to round the channel of input feature-maps. In summary, Table 1 lists the detailed architecture of our micro-dense block.

3.2 Global Residual Learning

In order to achieve storage of efficient feature-maps , we use identity mappings to connect different blocks of neural network globally. Till now, various residual learning based methods have been proposed e.g., ResNets [He *et al.*, 2016] and PyramidNets [Han *et al.*, 2017]. Due to the gradual increment of feature-maps, PyramidNets have good performance compared to other residual learning-based methods. Thus, we use the linear increasing for pyramidal residual learning [Han *et al.*, 2017] as global aggregation. Meanwhile, the number of feature-map channels increases as:

$$\mathcal{W}_k^\alpha = \lfloor \mathcal{W}_0 + k \times \alpha/N \rfloor, \quad if \quad 0 \le k \le N, \tag{14}$$

where N denotes the total number of micro-dense blocks, α is the additional channels in Micro-Dense Nets, $\mathcal{W}_k^\alpha$ is the output channel number of the k-th micro-dense block, which is increased by a step factor of α/N, and $\mathcal{W}_0$ is the number of input channels to first block which is set it to 16 according to [Han *et al.*, 2017].

4 Experiments and Discussion

We evaluate the proposed Micro-Dense Nets on several commonly-used benchmark datasets and compare with state-of-the-art DNNs, especially residual learning and dense aggregation based methods. We also compare with state-of-the-art NAS based methods, e.g., FPNASNet [Cui *et al.*, 2019].

Stage	input	Micro-Dense Net-30α64	Micro-Dense Net-60α115
Conv1	32×32	C3-B	C3-B
Conv2	32×32	$[MDConv-3]\times10$	$[MDConv-3]\times20$
Conv3	16×16	$[MDConv-3]\times10$	$[MDConv-3]\times20$
Conv4	8×8	$[MDConv-3]\times10$	$[MDConv-3]\times20$
Output	1×1 10 / 100	8×8 global average pool full connected layer, soft max	

Table 2: The detailed architecture of Micro-Dense Nets for CIFAR classification. The first block of Conv3, Conv4 use one conolutional layer of kernel of size 4 × 4 and stride of 2 to downsample the feature-maps. α is the width factor shown in Eq. 14.

4.1 Datasets and Implementation Details

The experiments were carried on two datasets: CIFAR dataset [Krizhevsky *et al.*, 2009] and ILSVRC 2012 classification dataset (ImageNet-1K) [Deng *et al.*, 2009]. Specifically, CIFAR contains colored nature images of resolution 32× 32 with two sub-datasets: CIFAR-10 from 10 classes object images and CIFAR-100 from 100 classes. The training and test sets of both CIFAR-10 and CIFAR-100 are 50,000 and 10,000 images, respectively. We report the final test error at the end of training on the 10K images from the test set. The ILSVRC 2012 classification dataset (ImageNet-1K) contains 1,000 classes images with 1.2 million images for training, and 50,000 for validation. The classification error was calculated on the validation set. We used auto-augmentation [Cubuk *et al.*, 2019] to train neural networks.

All the networks were trained using stochastic gradient descent(SGD). On CIFAR, the batchsize being 128 For ImageNet-1K, we train neural network with batchsize 400. The learning rate was initialized to 0, and gradually changed as:

$$lr = \ lr_{top} \times (1 + cos(\pi \times i/N_a))/2 \quad \forall i \in [0, N_a] \ , \quad (15)$$

$$lr_{top} = \begin{cases} lr_{max} \times i/N_w & \forall i \in [0, N_w] \\ lr_{max} & \forall i \in (N_w, N_a] \end{cases} , \quad (16)$$

where i is the iteration index, N_w is the number of iterations to warm up the learning rate, lr_{max} is the maximum learning rate which was set to 0.1 for CIFAR and 0.2 for ImageNet-1K, and N_a is the total number of iterations. Meanwhile, the weight initialization was used [He *et al.*, 2015], and the weight decaies of the optimizers were set to 10^{-4} for CIFAR and 10^{-5} for ImageNet-1K. Nesterov momentums [Sutskever *et al.*, 2013] of optimizers were 0.9 without dampening.

4.2 Experiments on CIFAR

Generally, the existing DNNs can be categorized into high efficiency and lightweight models, e.g., MobileNets and ShuffleNets [Zhang *et al.*, 2018a], and heavy and complex models for high accuracy, e.g., PyramidNets [Han *et al.*, 2017]. The proposed network could also be scaled into various levels. We designed two Micro-Dense Nets for CIFAR classification, namely Micro-Dense Net 30α64 and Micro-Dense Net 60α115, and their architectures are shown in Table 2.

From Tables 3 and 4, we can see that the proposed Micro-Dense Nets, which have the fewest number of parameters, reach the lowest error rates on both CIFAR-10/100.

Model	#Params(M)	Ratio-to-ours	CIFAR-10	CIFAR-100
GoogLeNet[Szegedy *et al.*, 2015]	3.5	5.0×	7.06	27.10
MobileNet [Sandler *et al.*, 2018]	2.2	3.1×	4.13	-
ShuffleNet [Zhang *et al.*, 2018a]	2.5	3.6×	5.83	-
ResNet [He *et al.*, 2016]	1.7	2.4×	6.43	-
ResNet+ [Wang *et al.*, 2019]	1.7	2.4×	6.43	-
Network in network [Lin *et al.*, 2013]	1.7	2.4×	5.46	24.33
FPNASNet [Cui *et al.*, 2019]	1.7	2.4×	3.99	-
ResNet with Stochastic Depth[ECCV-2016]	1.7	2.4×	5.25	24.98
PyramidNet(α=48) [Han *et al.*, 2017]	1.7	2.4×	4.58	23.12
Highway network [Srivastava *et al.*, 2015]	1.7	2.3×	7.54	-
MixNet [Wang *et al.*, 2018]	1.5	2.1×	4.19	21.12
DenseNet [Huang *et al.*, 2017]	1.0	1.4×	5.24	24.33
Micro-Dense Net-30α64	0.7	1.0×	3.41	20.85
Micro-Dense Net-30α64+	0.7	1.0×	3.28	19.47

Table 3: Error rate (%) of different lightweight methods. '+' denotes the method using dynamic regularization [Wang *et al.*, 2019]. The lowest (the best) and second lowest error rates are highlighted with red and blue, respectively.

Model	#Params(M)	Ratio-to-ours	CIFAR-10	CIFAR-100
MixNet [Wang *et al.*, 2018]	48.5	12.1×	3.13	16.96
Wide ResNet [Cubuk *et al.*, 2019]	36.5	9.1×	2.6	17.1
Res2NeXt-29 [Gao *et al.*, 2020]	36.7	9.2×	-	16.79
Res2NeXt-29-SE [Gao *et al.*, 2020]	36.9	9.2×	-	16.56
PyramidNet(α=270) [Han *et al.*, 2017]	27.0	6.8×	3.48	17.01
DenseNet [Huang *et al.*, 2017]	25.6	6.4×	3.46	17.18
SparseNet [Zhu *et al.*, 2018]	16.7	4.2×	3.22	17.71
DenseNet [Huang *et al.*, 2017]	15.3	3.8×	3.62	17.60
Wide ResNet [Tan *et al.*, 2019]	11.0	2.8×	4.27	20.43
FPNASNet [Cui *et al.*, 2019]	5.7	1.4×	3.01	-
NnasNet-A [Zoph *et al.*, 2018]	3.3	0.8×	2.65	-
Micro-Dense Net-60α115	4.0	1.0×	2.58	16.92
Micro-Dense Net-60α115+	4.0	1.0×	2.16	15.36

Table 4: Error rate (%) of different heavy models. '+' denotes the method using dynamic regularization. The lowest and second lowest error rates are highlighted with red and blue, respectively.

Specifically, compared with manually designed DNNs, e.g., ResNets and MixedNet, Micro-Dense Nets reach a compatible performance with up to 12.1 × smaller than state-of-the-art DNNs. Compared with the deep-wise convolution proposed by MobileNets, our cardinality adaptive convolution could further improve the network performance. Moreover, owning to the sufficient utilization of features, the proposed Micro-Dense Nets even achieve compitable performance with the state-of-the-art NAS based method, e.g., FPNASNet.

4.3 Experiments on ImageNet-1K

NAS has greatly boosted the performance of DNNs on ImageNet-1K. However, due to the limitation of the searching space, it is still hard for NAS to find the optimal neural network topology. Here, we integrated the proposed micro-dense architecture with the state-of-the-art classification model EfficientNet-B0 [Tan and Le, 2019]. Table 5 shows the top-1 and top-5 test accuracies on the ImageNet-1K dataset. We replaced the 8-th,9-th,10-th *MBConv* modules in EfficientNet-B0 with our micro-dense block under the same number of parameters and FLOPs, and the resulting new model is denoted by Micro-Dense-EfficientNet. It is also worth noting that the other part of Micro-Dense-EfficientNet are the same as the original EfficientNet-B0 one which are not optimized for *MDConv-n*. The *MBConv* blocks of EfficientNet-B0 are obtained from NAS algorithm [Tan *et al.*, 2019].

As listed in Table 5, experimental results reveal that our Micro-Dense-EfficientNet could also perform on par with NAS methods. That is, Micro-Dense-EfficientNet boosts the performance of EfficientNet-B0 by 0.5% point on Top-1 accuracy, which validates the advantage of our architecture. The activation mappings (thorugh [Selvaraju *et al.*, 2017])

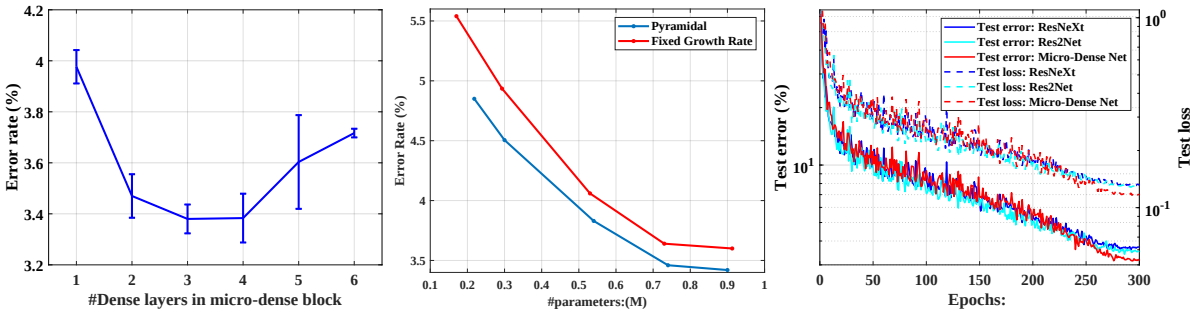

Figure 4: ***Left***: Error rates of networks with 0.7M parameters but varies with different numbers of dense layers. ***Middle***: Error rates of network with (fixed number / gradually increasing) channels. ***Right***: Error rate of the training process with different networks. We carried out all these experiments on CIFAR-10.

Model	Params(M)	FLOPs(B)	Top-1 Acc.	Top-5 Acc.
ResNet-50	26.0	4.1	76.0	93.0
DenseNet-169	14.0	3.5	76.2	93.2
NASNet-A (4 @ 1056)	5.3	0.6	74.0	91.6
EfficientNet-B0	5.3	0.4	76.3	93.2
EfficientNet-B0*	5.3	0.4	76.5	93.0
Micro-Dense-EfficientNet	5.3	0.4	76.8	93.2

Table 5: Accuracy (%) of state-of-the-art ImageNet-1K classification models including both the manually designed and NAS based methods. '*' denotes our implementation.

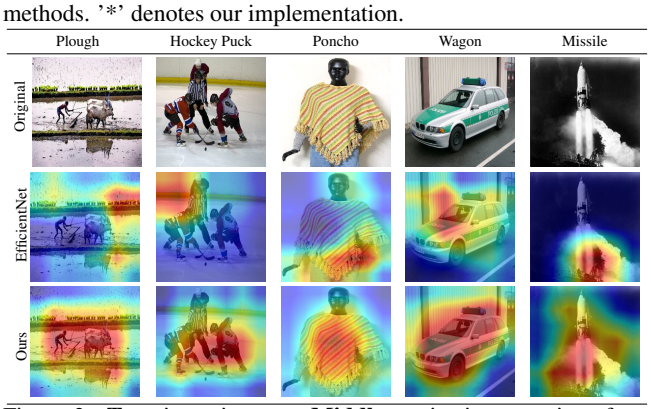

Figure 3: ***Top***: input images. ***Middle***: activation mappings from EfficientNet. ***Bottom***: activation mappings from Micro-Dense-EfficientNet. They represent the attentions of DNNs.

in Figure 3 also show that micro-dense architecture could help DNNs to get more accurate concentrations on the targets. Moreover, it is expected that replacing searching space from *MBcov* to *MDConv-n* could further rise up the frontier of feature learning.

4.4 Architecture Analysis

We carry out more experiments to investigate the proposed micro-dense architecture.

Micro-dense vs. dense aggregations. The dense aggregation could fully exploit features from different levels, but it also consume much resource on preserving and fusing accumulated feature-maps from proceeding layers. Thus, our micro-dense blocks with different branches but same number of parameters and depths are evaluated in this section. The result is shown in the left of Figure 4. We can observe that: 1) the local dense aggregation indeed boosts the performance of DNNs as the error rate declines with the number of dense layers increasing at the beginning stage; and 2) very deep dense aggregation is not necessary for the micro-dense block, and it may even degrade the performance of DNNs.

Model	CIFAR-10	CIFAR-100
ResNeXt	3.70	20.95
Res2Net	3.49	20.45
Micro-Dense Net	3.35	18.49

Table 6: Average error rate (%) of state-of-the-art models: Res2Net, ResNeXt with Micro-Dense Net. All the methods are with same number of parameters 1.8M and FLOPs 0.26G. The only variable is architecture of DNNs.

Pyramidal vs. fixed growth rate multi-level feature learning. In the micro-dense block, pyramidal multi-level feature learning with varying dimension cardinality is proposed to extract features efficiently. To validate pyramidal multi-level feature learning, We compare Micro-Dense Nets with ones trained with a fixed growth rate. The experimental results in the middle of Figure 4 shows that pyramidal multi-level feature learning strengthens the representation learning ability of DNNs.

Micro-dense vs. residual aggregations. To investigate the effectiveness of micro-dense aggregation in residual learning, we compare Micro-Dense Nets with state-of-the-art residual learning based methods Res2Next and Res2Net with same number of parameters and FLOPs. Note we only replaced each ResNeXt block with Res2Net or micro-dense block and they were trained under exact same conditions. From Table 6, we can see that Micro-Dense outperforms Res2Next and Res2Net, indicating the feature learning ability is boosted. Meanwhile, the test errors and losses are also compared in the right of Figure 4, and our Micro-Dense Net achieve the lowest errors and loss on both CIFAR-10/100.

5 Conclusion

In this paper, we proposed a new DNN architecture, namely Micro-Dense Nets, which takes advantages of identity mapping and dense aggregation to fuse features from different levels. Moreover, Micro-Dense Nets are characterized as pyramidal multi-level feature learning and dimension cardinality adaptive convolution. Extensive experimental results demonstrate that Micro-Dense Nets achieve state-of-the-art performance over CIFAR-10/100 and ImageNet-1K. In addition, Micro-Dense Nets require much fewer parameters and computational resources than ResNets and DenseNets. Last but not least, Micro-Dense Net could also be integrated with NAS based models to boost their performance, which forcefully validates the advantage of our design.

References

[Chen *et al.*, 2017] Yunpeng Chen, Jianan Li, Huaxin Xiao, Xiaojie Jin, Shuicheng Yan, and Jiashi Feng. Dual path networks. In *NIPS*, 2017.

[Cubuk *et al.*, 2019] Ekin D Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V Le. Autoaugment: Learning augmentation strategies from data. In *CVPR*, 2019.

[Cui *et al.*, 2019] Jiequan Cui, Pengguang Chen, Ruiyu Li, Shu Liu, Xiaoyong Shen, and Jiaya Jia. Fast and practical neural architecture search. In *ICCV*, 2019.

[Deng *et al.*, 2009] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009.

[Gao *et al.*, 2020] Shang-Hua Gao, Ming-Ming Cheng, Kai Zhao, Xin-Yu Zhang, Ming-Hsuan Yang, and Philip Torr. Res2net: A new multi-scale backbone architecture. *IEEE TPAMI*, 2020.

[Glorot and Bengio, 2010] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. *JMLR*, 9:249–256, 2010.

[Gu *et al.*, 2018] Jiuxiang Gu, Zhenhua Wang, et al. Recent advances in convolutional neural networks. *Pattern Recognition*, 77:354–377, 2018.

[Han *et al.*, 2017] Dongyoon Han, Jiwhan Kim, and Junmo Kim. Deep pyramidal residual networks. In *CVPR*, 2017.

[He *et al.*, 2015] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *ICCV*, 2015.

[He *et al.*, 2016] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[Huang *et al.*, 2017] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *CVPR*, 2017.

[Huang *et al.*, 2019] Yanping Huang, Youlong Cheng, et al. Gpipe: Efficient training of giant neural networks using pipeline parallelism. In *NIPS*, 2019.

[Ioffe and Szegedy, 2015] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015.

[Krizhevsky *et al.*, 2009] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

[Krizhevsky *et al.*, 2012] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012.

[Lin *et al.*, 2013] Min Lin, Qiang Chen, and Shuicheng Yan. Network in network. *arXiv:1312.4400*, 2013.

[Rawat and Wang, 2017] Waseem Rawat and Zenghui Wang. Deep convolutional neural networks for image classification: A comprehensive review. *Neural computation*, 29(9):2352–2449, 2017.

[Sandler *et al.*, 2018] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *CVPR*, 2018.

[Selvaraju *et al.*, 2017] Ramprasaath R Selvaraju, Michael Cogswell, et al. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE international conference on computer vision*, pages 618–626, 2017.

[Simonyan and Zisserman, 2014] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv:1409.1556*, 2014.

[Srivastava *et al.*, 2014] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *JMLR*, 15(1):1929–1958, 2014.

[Srivastava *et al.*, 2015] Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. Highway networks. *arXiv:1505.00387*, 2015.

[Sutskever *et al.*, 2013] Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. In *ICML*, 2013.

[Szegedy *et al.*, 2015] Christian Szegedy, Wei Liu, and orthers. Going deeper with convolutions. In *CVPR*, 2015.

[Tan and Le, 2019] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *ICML*, 2019.

[Tan *et al.*, 2019] Mingxing Tan, Bo Chen, Ruoming Pang, Vijay Vasudevan, Mark Sandler, Andrew Howard, and Quoc V Le. Mnasnet: Platform-aware neural architecture search for mobile. In *CVPR*, 2019.

[Wang *et al.*, 2018] Wenhai Wang, Xiang Li, Tong Lu, and Jian Yang. Mixed link networks. In *IJCAI*, 2018.

[Wang *et al.*, 2019] Yi Wang, Zhen-Peng Bian, Junhui Hou, and Lap-Pui Chau. Convolutional neural networks with dynamic regularization. *arXiv:1909.11862*, 2019.

[Wu and He, 2018] Yuxin Wu and Kaiming He. Group normalization. In *ECCV*, 2018.

[Xie *et al.*, 2017] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *CVPR*, 2017.

[Zhang *et al.*, 2018a] Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In *CVPR*, 2018.

[Zhang *et al.*, 2018b] Yulun Zhang, Yapeng Tian, Yu Kong, Bineng Zhong, and Yun Fu. Residual dense network for image super-resolution. In *CVPR*, 2018.

[Zhu *et al.*, 2018] Ligeng Zhu, Ruizhi Deng, Michael Maire, Zhiwei Deng, Greg Mori, and Ping Tan. Sparsely aggregated convolutional networks. In *ECCV*, 2018.

[Zoph *et al.*, 2018] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V Le. Learning transferable architectures for scalable image recognition. In *CVPR*, 2018.