GM-Net: Learning Features with More Efficiency

Yujia Chen
School of Computer & Communication Engineering
University of Science and Technology Beijing
Beijing, China
cyj.issac@gmail.com

Ce Li
China University of Mining & Technology, Beijing
University of Chinese Academy of Sciences
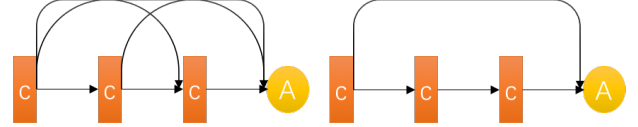*Beijing, China*
licekong@gmail.com

Figure 1. **Left:** BU_A with densely connections, where C stands for convolution unit and A stands for adaption unit. **Right:** BU_B with straight mapping.

Abstract—Deep Convolutional Neural Networks (CNNs) are capable of learning unprecedentedly effective features from images. Some researchers have struggled to enhance the parameters' efficiency using grouped convolution. However, the relation between the optimal number of convolutional groups and the recognition performance remains an open problem. In this paper, we propose a series of Basic Units (BUs) and a two-level merging strategy to construct deep CNNs, referred to as a joint Grouped Merging Net (GM-Net), which can produce joint grouped and reused deep features while maintaining the feature discriminability for classification tasks. Our GM-Net architectures with the proposed BU_A (dense connection) and BU_B (straight mapping) lead to significant reduction in the number of network parameters and obtain performance improvement in image classification tasks. Extensive experiments are conducted to validate the superior performance of the GM-Net than the state-of-the-arts on the benchmark datasets, e.g., MNIST, CIFAR-10, CIFAR-100 and SVHN.

Keywords-deep learning; image classification; grouped merging net; basic unit.

I. INTRODUCTION

Convolution Neural Networks (CNNs) has drawn lots of attention recently due to strong power of extracting structural and semantic features. Effective learning strategies enable CNNs, even a shallow one, to represent any bounded polynomial under certain conditions[1], which has brought remarkable performance improvement in vision tasks.

After LeNet[2] demonstrated the effectiveness (5 layers), the performance of Alexnet[3] (8 layers) and VGGnet[4] (19 layers) show that deeper is better. ResNet[5] and Highway Networks[6] have made it to more than 100 layers. However, deeper model is not necessarily better than shallower one. Information may lose sharply as going back through the network, thus layers in the beginning seldom get optimized, and these redundant layers will eventually become a hindrance.Wu *et.al* [7] introduced a shallower but wider model that gain better results without going very deep. Yet, it is not a once-for-all solution because it still has too much parameters that influence the speed and overfitting risk.

Although grouped convolution[3] was first used in Alexnet to compensate for the hardware deficiency, its utilization of making the connection sparse leading to more useful feature is verified by many scholars. Besides, we find that an appropriate merging strategy is very important while applying grouped convolution contiguously,

because each group has limited learning power and non-negligible information loss. Furthermore, outputs with different convolution groups focus on various contents, so that the combination of different convolution group numbers will produce more powerful feature maps.

In this paper, we propose an architecture with well-designed basic units (BUs) (Figure 1), slightly different with basic form, detailed in Section III. As shown in Figure 2, we define four units as BU_i (input), BU_f (full), BU_s (single) and BU_e (end). After BU_i, the input will flow through BU_f and BU_s separately in order to get representations with different structural and semantic information. Then, these two flows are merged and fed into BU_e, leading to capture relations between channels. Particularly, our model aims to extract better feature via deliberately-designed feature reuse instead of pursuing being deeper. In addition, to reduce abundant parameters, we apply grouped convolution in almost each convolutional layer.

BU, composed of 3 convolution units and 1 adaption unit, is the crux of our models. Two kinds of BUs are proposed as BU_A and BU_B (Figure 1). Enlightened by[8], BU_A applies densely connection, which connects each layer to others. However, it originally aims to alleviate gradient vanishing, strengthen feature propagation and reduce parameter number with concatenation operation. We take this advantage mainly by replacing concatenation with summation operation, since simply stacking different layers together can't learn features with high-level information sufficiently, demonstrated in Section IV. Moreover, the negative influence of more parameters with summation operation is mitigated by grouped convolution. Different with BU_A, the BU_B applies straight mapping from beginning to the end. We observe that two BUs have similar performance and BU_A is slightly better on more complex datasets (Table II), and evaluate two models on four benchmarks: MNIST[9], CIFAR-10[10], CIFAR-100[10] and SVHN[11]. With only 29 layers and
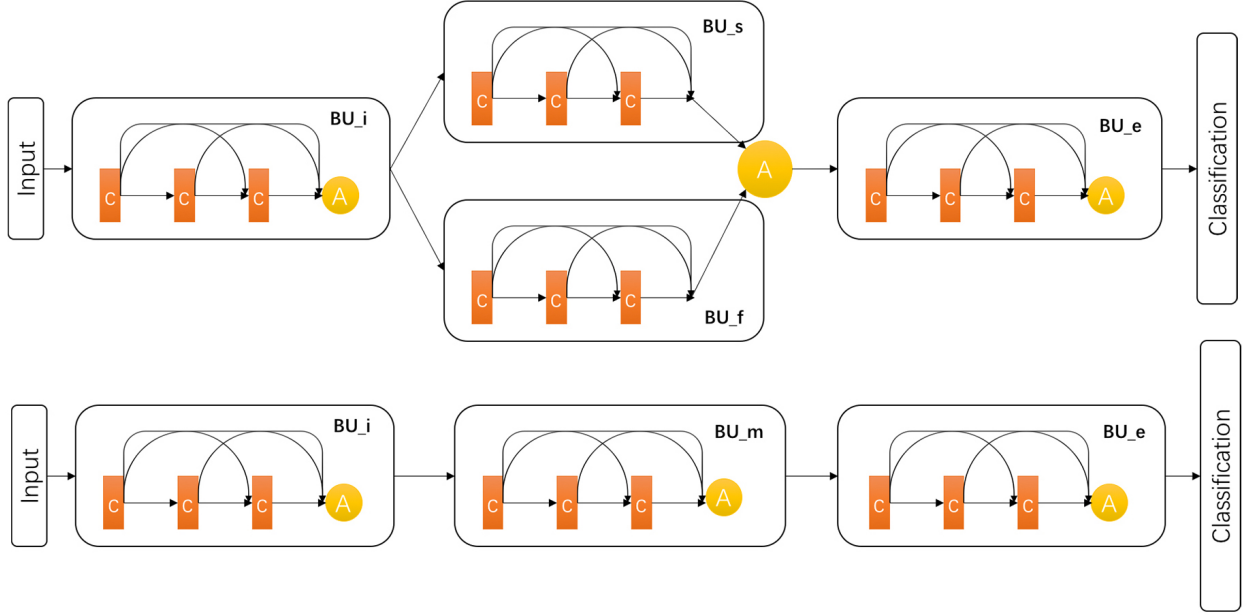
Figure 2. Our proposed structures. **Top:** GM-Net with BU_A. **Bottom:** Baseline model with BU_A.

2 paths, they gains competitive results with state-of-the-arts, which demonstrates that the extracted feature are informative enough to represent the contents, even if our model contains far less parameters.

The paper is organized as follows. Section II reviews related works. Section III describes our proposed model. Experiments are in Section IV and conclusion in Section V.

II. RELATED WORK

Overfitting: Overfitting has long been a major issue in computer vision tasks, since millions of parameters tend to remember each training data[12]. Dropout[13] provides an effective regularization that prevents co-adaptions on training data by randomly dropping out neurons. It can be explained as an ensemble of sparse neural networks to reduce overfitting risks. Similarly, stochastic depth[14] and drop path[15] utilize the idea of only training part of the network to optimize models. We attempt to make parameters sparse and compact by dropping connections with special basic units. Sparse models reduce overfitting generally, but underfitting will occur if it is too sparse. This problem is handled in our models with a unique merging strategy.

Feature Reuse: Combining low-level features that contain detailed information and high-level features that contain semantic information together has demonstrated its practicability in recent works. Residual representation and identity mapping in ResNet[16] value in both forward and backward propagation, and residual learning is a good reformulation or preconditioning that simplify optimization and fully take advantages of all the knowledges flowing in forward. And identity mapping provides a simpler method. DenseNet[8] uses different level features by concatenating thin feature maps, which also reduce parameters dramat-

ically. All methods suggest that finding an effective way of feature reuse will boost model's accuracy.

Grouped Convolution: Ascribe to the computing limitation, AlexNet[3] has to split its feature maps into two groups, which enlightens scholars nowadays to find high-efficient architectures. Grouped convolution hasn't got much attention until ResNeXt[17] pointed out the significance of cardinality, explained as the number of convolutional groups. By increasing cardinality, ResNeXt[17] increases its accuracy when maintaining model's complexity and parameter numbers. Xception[18] proposed depthwise separable convolution, or channel-wise convolution, where each channel is deemed as a convolution group. As achieving higher performance, an open question is still left behind: What is the optimal number of convolutional groups?

III. APPROACH

A. Architecture Overview

As shown in Figure 2, images are first fed into input basic unit (denoted as BU_i), and then flow through full-convolution unit (denoted as BU_f) and single-channel convolution unit (denoted as BU_s) separately. Finally, with end-flow unit (denoted as BU_e) merging two paths together, output is computed on high-level semantic features. Our baseline model only replaces BU_f and BU_s with a middle unit (denoted as BU_m), which has the same settings with BU_e. Table I describes the two proposed models.

Generally, GM-Net contains two paths in the middle. After splitting channels into different groups, three convolution units are performed contiguously. In Caffe[19], grouped convolution is realized by matching input groups with output groups in order, so it can also be regarded as each single group flows through three convolution units

BU	Output	BU_A (B)		Baseline	
BU_i	16 × 16	3 × 3 × 64		3 × 3 × 64	
		$\begin{array}{l}1 \times 1 \times 64 \\ 3 \times 3 \times 128\end{array}$	× 3, g = 8	$\begin{array}{l}1 \times 1 \times 64 \\ 3 \times 3 \times 128\end{array}$	× 3, g = 8
		3 × 3 × 128, g = 4		3 × 3 × 128, g = 4	
		2 × 2 AvePool, stride2		2 × 2 AvePool, stride2	
BU_f	16 × 16	$\begin{array}{l}1 \times 1 \times 128 \\ 3 \times 3 \times 256\end{array}$	× 3, g = 1	—	
BU_s	16 × 16	1 × 1 × 256, g = 1			
		3 × 3 × 256, ×3, g = 256			
AU	8 × 8	3 × 3 × 256, g = 4			
		2 × 2 AvePool, stride2			
BU_m	8 × 8	—		$\begin{array}{l}1 \times 1 \times 128 \\ 3 \times 3 \times 256\end{array}$	× 3, g = 8
				3 × 3 × 256, g = 4	
				2 × 2 AvePool, stride2	
BU_e	8 × 8	$\begin{array}{l}1 \times 1 \times 192 \\ 3 \times 3 \times 384\end{array}$	× 3, g = 8	$\begin{array}{l}1 \times 1 \times 192 \\ 3 \times 3 \times 384\end{array}$	× 3, g = 8
		1 × 1 × 384, g = 4		1 × 1 × 384, g = 4	
	1 × 1	Global AvePool, 10d-fc, softmax		Global AvePool, 10d-fc, softmax	
#Params	-	1.5M		0.7 M	

uninterruptedly. In this way, each BU has paths of the same number with group numbers. The method also selects the best local feature of the same location with different filters. Similarly, we realize it by squeezing the main flow into some groups and apply apt merging strategy. Besides, sparse connection caused by grouped convolution ensures that no redundant information is learned.

Merging is another key of our model. Limited by the channel numbers of each group, the learning power has its upper bound. Consequently, contiguous grouped convolution can't be performed in a long sequence. That's why we merge all paths with an adaption unit after only three convolution units. More details are presented in Section III-C.

BU_A and BU_B is also designed specifically to reinforce feature reuse. BU_A contains densely connected mapping to reuse features from all previous stages, reducing the information loss incurred by small number of parameters. And BU_B with a straight mapping follows the idea in[16], enabling each unit to learn in a sequence. Also, both of two methods provides effective means in backward propagation.

B. Basic Unit Design

Convolution unit is composed of a bottleneck where a 1×1 convolution is applied before a 3×3 convolution. In convolution unit. Adaption unit consists of a pooling layer with stride 2 after a 3×3 or 1×1 convolution layer. To obtain different representative features, we use standard convolution in BU_f and channel-wise grouped convolution in BU_s. We empirically demonstrated the effectiveness of this method in Section IV-B. The number of parameters of a grouped convolution can be computed as:

$$k \times k \times \frac{c}{g} \times \frac{o}{g} \times g \qquad (1)$$

where k denotes kernel size, c denotes the number of input channels, o denotes the number of output channels, and g is the group number. In this way, parameters are g times smaller, and more compact features can be learned.

C. Merging Strategy

Merging is applied densely in our model to compensate for the disadvantages of grouped convolution. Our merging strategy has two levels. In operation level, 1×1 convolution inside a convolution unit has two functions: reducing channels and merging information. So we don't use grouped convolution on it. Thus, merging is applied inside every convolution unit. In unit level, BU_A or BU_B merges features from different layers to reinforce detailed information, demonstrated in Section IV-B.

D. Implementation Details

For each convolution unit, 1×1 convolution reduce the number of channels with the factor of 0.5. All convolution operation is composed of conv-BN-ReLU. We use 8 groups for 3×3 convolution and 4 groups for adaption unit. Besides, dropout is used in BU_i and BU_e to reduce

Method	Depth	#Params	C10	C10+	C100	C100+	SVHN	MNIST
Network in Network[20]	-	-	10.41	8.81	35.68	-	2.35	0.45
DropConnet[21]	-	-	-	9.32	-	-	1.94	0.23
Highway Network[6]	19	2.3M	-	7.72	-	32.39	-	0.45
ResNet[5]	110	1.7M	-	6.61	-	-	-	-
Stochastic Depth[14]	110	1.7M	11.66	5.23	37.8	24.58	1.75	-
Fractal Net[15]	21	38.6M	10.18	5.22	35.34	23.3	2.01	-
Wide ResNet[22]	16	8.9M	-	4.97	-	22.89	-	-
Fitnet4-LSUV[23]	17	2.5M	-	6.06	-	27.66	-	-
Tree+Max-Avg[24]	-	-	7.62	6.05	32.37	-	1.69	0.31
DenseNet (growth rate =12)[8]	40	1M	7.00	5.24	27.55	24.42	1.79	-
Baseline	23	0.7M	7.13	5.98	27.59	25.81	2.39	0.3
BU_A	**29**	**1.5M**	**6.81**	**5.15**	**27.09**	**24.97**	**1.88**	**0.21**
BU_B	**29**	**1.5M**	**6.77**	**5.62**	**27.08**	**25.22**	**1.92**	**0.22**

overfitting with keep probability of 0.8. In experiment settings, we use Nesterov optimization with momentum 0.9. Base learning rate is 0.1 or 0.01 (MNIST) and is reduced 10 times after half and 3/4 of total epochs. SVHN and MNIST datasets are trained 200 epochs and 300 epochs for other datasets.

IV. EXPERIMENTS

In this section, we first conduct experiments on MNIST[9], CIFAR-10[10], CIFAR-100[10] and SVHN[11] datasets in Section IV-A. We also conduct ablation experiments, without data augmentation or other tricks, to further analyze the practicability of our model with CIFAR-10 dataset in Section IV-B.

A. Experiments on Different Datasets

MNIST. It contains 60000 hand-written digits for training and 10000 for testing. The images are collected from 250 writers and have shapes of $1 \times 28 \times 28$ where 1 denotes one channel.

CIFAR. It consists of 60000 32×32 colored natural scene images, 10000 of which are used for testing. Specifically, CIFAR-10 (C10) contains 10 classes, with 5000 training images and 1000 testing images per class, while CIFAR-100 (C100) has 100 classes, 500 training images and 100 testing images for each classes. Channel means are computed and subtracted in preprocessing. We also apply standard augmentation[6], [14], [15], [22], [8], [23] (marked as C10+ and C100+), leading into higher accuracy.

SVHN. It is a real-world dataset obtained from house numbers in Google Street View images. It consists 10 classes, where 73257 digits for training, 26032 digits for testing, and 531131 additional. All digits have been resized to 32-by-32 pixels. The task is to classify the central digit into a correct class.

Result Analysis. In our method, the efficiency of parameters is the key. We observe that even without large number of parameters, our models obtain accuracy that are comparable with many state-of-the-art methods (Table II). Our baseline model contains only 0.7M parameters, but performs almost the same as Fractal Net[15] which has **40 times** more parameters. This fully demonstrates that our models extract most useful information without overfitting. Moreover, with alike parameters, BU_A and BU_B outperform well-designed architectures like Highway Network[6] and Fitnet4-LSUV[23] by a large margin on CIFAR-10 and CIFAR-100. We believe it is the better output representations that accounts for the result.

On real-world SVHN dataset with more complex background, though our models obtain good result, there is still a lot space to improve on. For example, increasing parameters can help grasp more crucial information. On MNIST dataset, GM-Net outperforms lots of well-designed architectures and achieves state-of-the-art accuracy. Since it is a comparatively simpler dataset, useful information is easier to extract and utilize. We ascribe the results to the informative representation of features.

Besides, BU_A and BU_B perform similar, while B_A is slightly better on more complex datasets like CIFAR-100 and SVHN. Because CIFAR-100 has fewer images for each class and SVHN contains many ambiguous digits, dense connections can act as good form of feature reuse to effectively compensate for information loss during forward propagation.

B. Experiments on Architecture

1) Study on Grouped Convolution: As original input images and bottom convolution layers contain highly correlated adjacent pixels, the reduction of connections in bottom layers can lead to sparse representation, forcing to abandon redundant information. Besides, making parameters compact on top layers may also lead to more representative features. We compare putting blocks with grouped convolution at different locations (Table III). The result shows that model without grouped convolution has

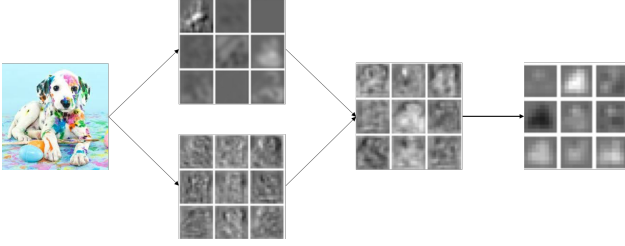| Settings	Accuracy on CIFAR-10
No groups	92.55%
Groups conv on block1	93.03%
Group conv on block3	93.01%
Both block1 and block3 (**ours**)	**93.19%**
8+2	92.86%
8+4 (**ours**)	**92.87%**
8+4&16+8	92.35%
4+4&4+4	92.01%

Figure 3. Feature maps visualization. The features on the top are extracted from the end of BU_s, the bottom ones are from the end of BU_f, the third ones are from the adaption unit after BU_s and BU_f, and the last ones are from the last convolution layer. Features become more discriminable with the flow as expected.

the lowest accuracy. As expected, setting grouped convolution on block1 gains an increase of 0.02% than block3. The low image resolution of datasets may explain the little gap there is less need to reduce correlation on images with low resolution. Last, if we split the channels into groups for both block1 and block3, accuracy will also increase by 0.64%. Thanks to merging strategy, information is efficiently utilized with the breakthrough in the limitations of group's learning ability.

How many groups should be used still remains an open question. We try two methods on baseline model keeping group number and keeping channel numbers (Table III). For the former one, we set 8 for group numbers of convolution units, and 2 and 4 for adaption unit in both BU_i and BU_m (marked as 8+2 or 8+4). For the latter method, channel number is fixed at 8 for each group in convolution unit and 16 in adaption unit (marked as 8+4 & 16+8). A setting where 4 groups used in both convolution and adaption units is also presented as a comparison.

The fact that our setting outperforms others verifies the advantages of the merging strategy. The increase of channel number from the latter block blender flows together, grasping more information. If we decrease the group number in adaption unit to 2, performances are similar while number of parameters vary sharply. The comparison between the first setting and the last setting also bespeaks that our grouped convolution helped grasp useful features instead of causing information loss.

2) Comparison on Feature Reuse Method: We can see from Table II that GM-Net outperform our baseline by
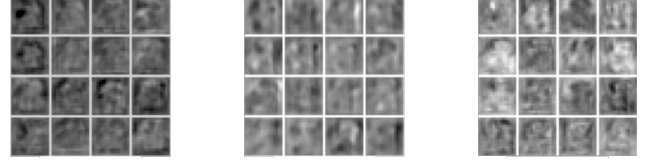

Figure 4. Features with different merging strategy experimented with baseline model. Features are extracted from the adaption unit after BU_m. **Left:** operation-level merging is taken out, resulting in an accuracy drop by 0.75% and darker feature maps. **Middle:** unit-level merging is taken out, resulting in an accuracy drop by 0.4% and blurred feature maps. **Right:** features with our merging strategy, showing shaper objects and informative representation.

around 0.35%-0.83%, verifying that combining features that propagated from different convolution groups can effectively increase the accuracy. Figure 3 and Figure 4 show the function of our merging strategy. We can observe from Figure 3 that features after BU_s are sparser and concentrate only on a specific part on each feature map, while features after BU_f contains a lot more redundant information. By combining them together after an adaption unit, we find the part of the object is more highlighted and the motif shows more discriminability, which leads to a more informative classification layer afterwards.

We also conduct ablation experiments to see how merging strategy influence the features. While accuracy drops by 0.75% on CIFAR-10, darker feature maps in Figure 4 (Left) shows the significant information loss without operation-level merging. If we only take out unit-level merging, result drops by 0.4%. Feature maps is lighter but more blurred. This may match the learning limitation with groups. With our two-level yet simple merging strategy, feature maps become sharper and still informative enough.

Finally, we plot test error for model with sum merging and concatenate merging (Figure 5) to verify superior performance of sum operation than concatenation. Accuracy drops by 1.1% with the latter method. Specifically, the basic unit for concatenate merging is like [8] but shorter. To keep the parameters in a same level, we set growth rate as 20, 40, 60 for BU_i, BU_f and BU_e. The number can be easily computed with Equation (1). BU_s is performed as described in Section III-B. Furthermore, no grouped convolution is used in the latter method, because it has already contained a similar idea of compacting parameters.

V. CONCLUSION

With the study on sparse issue of grouped convolution for applications in computer vision in this paper, we propose a joint Grouped Merging Network (GM-Net) with two basic unit forms, BU_A (dense connection) and BU_B (straight mapping), and a simple two-level merging strategy to compensate for the reduction of parameters and learn features with more efficiency. Joint grouped and reused semantic features are produced and demonstrates superior efficiency on utilizing parameters than the state-of-the-arts. The improvement of experimental result is obtained in four image classification tasks and ascribed to
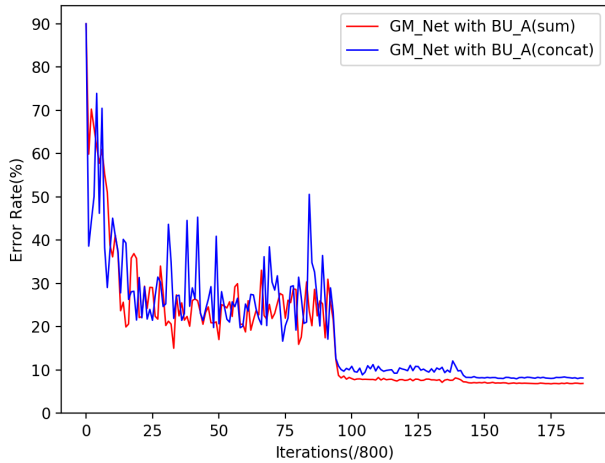
Figure 5. CIFAR-10 test error rate of GM-Net with two kinds of merging methods.

the better feature representation. In the future, we will keep making CNNs lighter and learning more representative features.

ACKNOWLEDGMENT

REFERENCES

[1] A. Andoni. Learning polynomials with neural networks. *Proceedings of International Conference on Machine Learning*, 2014.

[2] Yann Lecun, Leon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[3] A. Krizhevsky, I. Sutskever, and G.E. Hinton. Imagenet classification with deep convolutional neural networks. *Proceedings of Neural Information Processing Systems*, 2012.

[4] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael S Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

[5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *Proceedings of Computer Vision and Pattern Recognition*, pages 770–778, 2015.

[6] Rupesh Kumar Srivastava, Klaus Greff, and Jurgen Schmidhuber. Training very deep networks. *Proceedings of Neural Information Processing Systems*, pages 2377–2385, 2015.

[7] Z. Wu, C. Shen, and A. Hengel. Wider or deeper: Revisiting the resnet model for visual recognition. *arXiv*, 1611.10080, 2016.

[8] G. Huang, Z. Liu, and K. Q. Weinberger. Densely connected convolutional networks. *arXiv*, 1608.06993, 2016.

[9] LeCun Y. The mnist database of handwritten digits. *http://yann. lecun. com/exdb/mnist/*, 1998.

[10] A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images. *Technical Report*, 2009.

[11] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A.Y. Ng. Reading digits in natural images with unsupervised feature learning. *Proceedings of Neural Information Processing Systems Workshop*, 2011.

[12] C. Zhang, S. Bengio, and M. Hardt. Understanding deep learning requires rethinking generalization. *arXiv*, 1611.03530, 2016.

[13] N. Srivastava, G. Hinton, and A. Krizhevsky. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[14] Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Q Weinberger. Deep networks with stochastic depth. *Proceedings of European Conference on Computer Vision*, 2016.

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

[16] K. He, X. Zhang, and S. Ren. Identity mappings in deep residual networks. *Procceddings of European Conference on Computer Vision*, pages 630–645, 2016.

[17] S. Xie, R. Girshick, and Dollar P. Aggregated residual transformations for deep neural networks. *arXiv*, 1611.05431, 2016.

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

[19] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv*, 1408.5093, 2014.

[20] M. Lin, Q. Chen, and S. Yan. Network in network. *Proceedings of International Conference on Learning Representations*, 2014.

[21] L. Wan, M. Zeiler, and S. Zhang. Regularization of neural networks using dropconnect. *Proceedings of International Conference on Machine Learning*, pages 1058–1066, 2013.

[22] S. Zagoruyko and N. Komodakis. Wide residual networks. *arXiv*, 1605.07146, 2016.

[23] D. Xie, J. Xiong, and S. Pu. All you need is beyond a good init: Exploring better solution for training extremely deep convolutional neural networks with orthonormality and modulation. *arXiv*, 1703.01827, 2017.

[24] C.Y. Lee, P.W. Gallagher, and Z. Tu. Generalizing pooling functions in convolutional neural networks: Mixed, gated, and tree. *International Conference on Artificial Intelligence and Statistics*, 2016.