Reconciling Feature-Reuse and Overfitting in DenseNet with Specialized Dropout

Kun Wan, Shu Yang, Boyuan Feng, Yufei Ding
Department of Computer Science
University of California, Santa Barbara
Santa Barbara, USA
{kun, shu, boyuan, yufeiding}@cs.ucsb.edu

Lingwei Xie
School of Infomatics
Xiamen University
Xiamen, China
xielingwei@stu.xmu.edu.cn

Abstract—**Recently convolutional neural networks (CNNs) achieve great accuracy in visual recognition tasks. DenseNets become one of the most popular CNN models due to its effectiveness in the feature-reuse. However, like other CNN models, DenseNets also face the overfitting problem if not more severe. Existing dropout methods can be applied but not effective. In particular, the property of the feature-reuse in DenseNets will be impeded, and the dropout effect will be weakened by the spatial correlation inside feature maps. To address these problems, we craft the design of a specialized dropout method from three aspects, the dropout location, the dropout granularity, and the dropout probability. The insights attained here could potentially be applied as a general approach for boosting the accuracy of other CNN models with similar shortcut connections. Experimental results show that DenseNets with our specialized dropout method yield better accuracies compared to vanilla DenseNets and state-of-the-art CNN models, and such accuracy boost increases with the model depth.**

I. Introduction

Recent years have seen a rapid development of deep neural network in the computer vision area, especially for visual object recognition tasks. From AlexNet, the winner of ImageNet Large Scale Visual Recognition Challenge (ILSVRC) [12], to later VGG network [15] and GoogLeNet [19], CNNs have shown significant success with its shocking improvement of accuracy.

Researchers gradually realized that the depth of the network always plays an important role in the final accuracy of one model due to the increased expressiveness [3, 4, 19]. However, simply increasing the depth does not always help because of the induced vanishing gradient problem [1]. ResNet [5] has been proposed to promote the flow of information across layers without attenuation by introducing *identity skip-connections*, which sums together the input and the output of several convolutional layers. In 2016, Densely connected network (DenseNet) [9] came out, which replaces the simple summation in ResNet with concatenation after realizing the summation may also impede the information flow.

Despite the information flow, DenseNet still suffers from the overfitting problem, especially when the network goes deeper. Standard dropout [17] has been used to combat with such problem, but can not work effectively on DenseNet. The reasons are twofold: 1) The feature-reuse will be weakened by the standard dropout as it makes features dropped at previous layers no longer be used at later layers. 2) The standard dropout method does not interact well with convolutional layers because of the spatial correlation inside feature maps [20]. Since the *dense connectivity* increases the number of feature maps tremendously — especially at deep layers — the effectiveness of standard dropout would further be reduced.

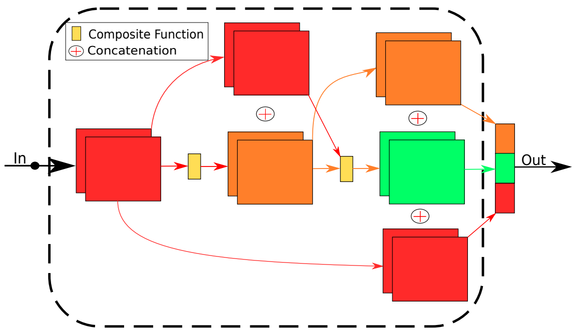
In this paper, we design a specialized dropout method to solve these problems. In particular, three aspects of dropout design are addressed: 1) Where to put dropout layers in the network? 2) What is the best dropout granularity? 3) How to assign appropriate dropout (or survival) probabilities for different layers? Meanwhile, we show that the idea to re-design the dropout method from the three aspects also applies to other CNN models like ResNet. The contributions of the paper can be summarized as follows:

- First, we propose a new structure named pre-dropout to solve the possible feature-reuse obstruction when applying the standard dropout method on DenseNets.
- Second, we are the first to show that the channel-wise dropout (compared to the layer-wise and the unit-wise) fit best for CNNs through both detailed analysis and experimental results.
- Third, we propose three distinct probability schedules, and via experiments we find out the best one for DenseNets.
- Last, we provide some insights for future researchers by showing potential aspects and options worth considering when applying the dropout method on CNN models.

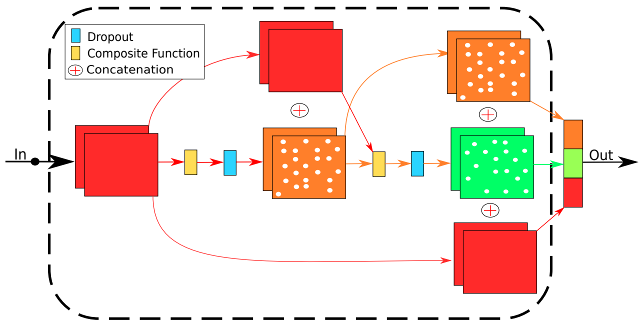
Experiments in our paper suggest that DenseNets with our proposed specialized dropout method outperform other comparable DenseNets and state-of-art CNN models in terms of the accuracy, and by following the same idea dropout methods designed for other CNN models can also achieve consistent improvements over the standard dropout method.

II. Related Work

In this section, first we will review some basic ideas behind DenseNet, which is also the foundation for our method. Then the standard dropout method along with some variants will be introduced as counterparts of our proposed dropout approach.



(a) One *dense block* in the DenseNet



(b) DenseNet with the standard dropout

Fig. 1: Examples of the *dense block* and the DenseNet with the standard dropout method. White spots denote dropped units. The right figure shows that the same feature maps (orange color) after the dropout layer will be directly sent to later layers, which makes the dropped features always unavailable to later layers.

A. DenseNet

DenseNet was first proposed by [9], which features a special structure—*dense blocks*. Figure 1a gives an example of the *dense block*. One *dense block* consists of several convolutional layers, each of which output k feature maps, where k is referred to as the *growth rate* in the network. The most important property of DenseNet is that for each convolutional layer inside the *dense block*, the input is the concatenation of all feature maps from the preceding layers within the same block, which is also known as the *dense connectivity*. With *dense connectivity* previous output features can be reused at later layers.

$$\mathrm{X}_\ell = \mathrm{L}_\ell([\mathrm{X}_0, \mathrm{X}_1, \cdots, \mathrm{X}_{\ell-1}]) \tag{1}$$

Equation 1 shows such kind of relationship clearly. Here $\mathrm{X_i}$ represents the output at layer i. $[,]$ denotes concatenation operation. L is a composite function of a batch normalization (BN) [11], a rectified linear unit (ReLU) [2] and a 3×3 convolutional layer.

With the help of the feature-reuse, DenseNet achieves a better performance and turns out to be more robust [8]. However, the *dense connectivity* will cost a large amount of parameters. To ease the consumption of parameters, a variant structure, named DenseNet-BC, came out. In this structure, a 1×1 layer is added before each 3×3 layer to reduce the depth of the input to $4k$.

B. Dropout

The standard dropout method [7, 17] discourages the co-adaptation between units by randomly dropping out unit values. Stochastic Depth [10] extends the idea to the layer level by randomly dropping out whole layers. Other alternative examples include DropConnect [21] which generalizes the dropout by dropping individual connections instead of units, and Swapout [16] which skips layers randomly at the unit level.

These previous works mainly focus on one aspect of dropout method, i.e., the dropout granularity. We are the first who gives a thorough study of the dropout design from all three aspects: the dropout location, the dropout granularity and the dropout probability. In our experiments, we will not only give the overall accuracy improvements, but also the breakdown along the three aspects. Meanwhile, notice that all methods above will impede the feature-reuse in DenseNets since dropped features in previous layers will never be available to later layers. Figure 1b shows an example of the feature-reuse obstruction when applying the standard dropout method on DenseNet.

III. Model Description

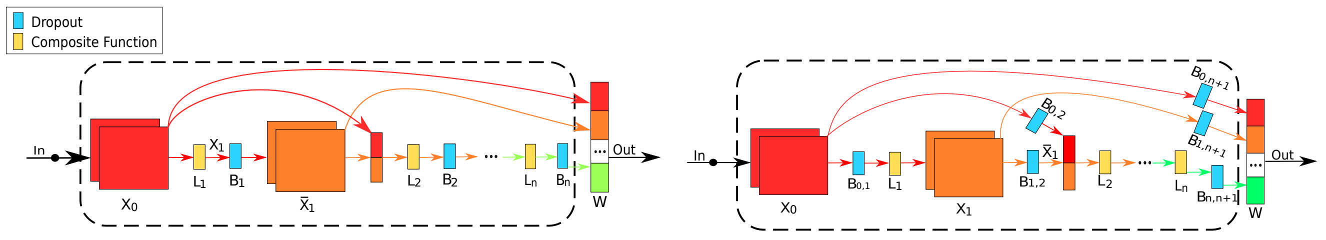
As previously mentioned, the standard dropout method will have some limitations on DenseNets: 1. It can impede the feature-reuse; 2. The dropout effect will be weakened by the spatial correlation.

To solve these problems and further improve the model generalization ability, we propose the following structures.

A. Pre-dropout Structure

The pre-dropout structure aims at solving the possible feature-reuse obstruction when applying dropout methods on DenseNets. As mentioned in Section II-B, due to the *dense connectivity*, the standard dropout will make features dropped at previous layers no longer be used at later layers, which will impede the effect of the feature-reuse in DenseNets. To retain the feature-reuse, we come up with a simple yet novel idea named pre-dropout, which instead places dropout layers before composite functions so that complete outputs from previous convolutional layers can be transferred directly to later layers before the dropout is applied. Meanwhile one extra benefit from the pre-dropout is that we can stimulate much more feature-reuse patterns in the network because of different dropout patterns applied at different layers. Figure 2 illustrates the differences between the standard dropout and the pre-dropout. Next, we will explain these two benefits in details.

Suppose the input to Figure 2a is $\mathrm{X}_0^{standard}$, then it would also be used as the input to the composite function L_1. The



(a) Standard dropout (b) Pre-dropout

Fig. 2: Examples illustrating data flows in one *dense block* of the standard dropout method and the pre-dropout method. The blue boxes represent dropout layers, while the yellow boxes represent composite functions. X_0, X_1 and $\overline{\mathrm{X}}_1$ are the data tensors. W stands for the final output of the *dense block*.

calculation of B_1 can be seen as element-wisely multiplying $\mathrm{X}_1^{standard}$ with Θ_1, a tensor of random variables following $Bernoulli(p_1)$. Thus we can get

$$\overline{\mathrm{X}}_1^{standard} = \Theta_1 \odot \mathrm{X}_1^{standard} = \Theta_1 \odot \mathrm{L}_1(\mathrm{X}_0^{standard}) \quad (2)$$

where $\odot$ represents element-wise multiplication. Then $\mathrm{X}_0^{standard}$ and $\overline{\mathrm{X}}_1^{standard}$ will be concatenated together as the input to L_2, i.e., $\mathrm{X}_0^{standard} \oplus \overline{\mathrm{X}}_1^{standard}$, where $\oplus$ denotes concatenation, so on and so forth. Finally, the output of the *dense block* can be written as

$$\mathrm{W}^{standard} = \mathrm{X}_0^{standard} \oplus \overline{\mathrm{X}}_1^{standard} \oplus \cdots \oplus \overline{\mathrm{X}}_\mathrm{n}^{standard} \quad (3)$$

Similarly we can get the formulas for Figure 2b, which shows the data flow in a *dense block* with the pre-dropout method,

$$\overline{\mathrm{X}}_1^{pre} = \Theta_{1,2} \odot \mathrm{X}_1^{pre} = \Theta_{1,2} \odot \mathrm{L}_1(\Theta_{0,1} \odot \mathrm{X}_0^{pre}) \quad (4)$$

$$\mathrm{W}^{pre} = (\Theta_{0,\mathrm{n+1}} \odot \mathrm{X}_0^{pre}) \oplus (\Theta_{1,\mathrm{n+1}} \odot \mathrm{X}_1^{pre}) \oplus \cdots \oplus (\Theta_{\mathrm{n,n+1}} \odot \mathrm{X}_\mathrm{n}^{pre}) \quad (5)$$

where $\Theta_{\mathrm{i,j}}$ represents the tensor of the dropout layer connecting from the output of the layer i to the input of the layer j, in which random variables follow $Bernoulli(p_{i,j})$.

Comparing Equation 2 with 4, we can find that the pre-dropout method will allow a better feature-reuse than the standard dropout method. For instance, the output of the standard dropout would become zero and remains the same as the input to all following layers when Θ_1 equals to zero. However the pre-dropout solves this problem. In the pre-dropout method, X_1^{pre} would be multiplied by different independent tensors $\Theta_{1,2}, \Theta_{1,3}, \cdots, \Theta_{1,\mathrm{n+1}}$, such that for X_1^{pre} even if $\Theta_{1,2}$ makes it become zero at L_2, we still have a chance to reuse this feature at later layers.

Meanwhile we also notice that the pre-dropout method can stimulate much more feature-reuse patterns in the network. For example, in Figure 2a, once $\mathrm{X}_1^{standard}$ goes through the dropout layer B_1, the same output will always be reused. Whereas in the pre-dropout method, every time before X_1^{pre} is utilized as the input to the next layer, it will be multiplied by a different tensor. For example, in Figure 2b, the contributions of X_1^{pre} are actually two distinct features, since $\Theta_{1,2}$ and $\Theta_{1,\mathrm{n+1}}$ are independent.

Note that the similar feature-reuse obstruction also exists when applying the standard dropout on other CNN models with shortcut connections, such as ResNet [5], Wide-ResNet [22] and RoR [23]. Thus the pre-dropout method could work for those networks as well.

B. Channel-wise Dropout

When designing our specialized dropout method, the dropout granularity is also an important aspect to be considered. Figure 3 shows three different types of granularity: the unit-wise, the channel-wise and the layer-wise.

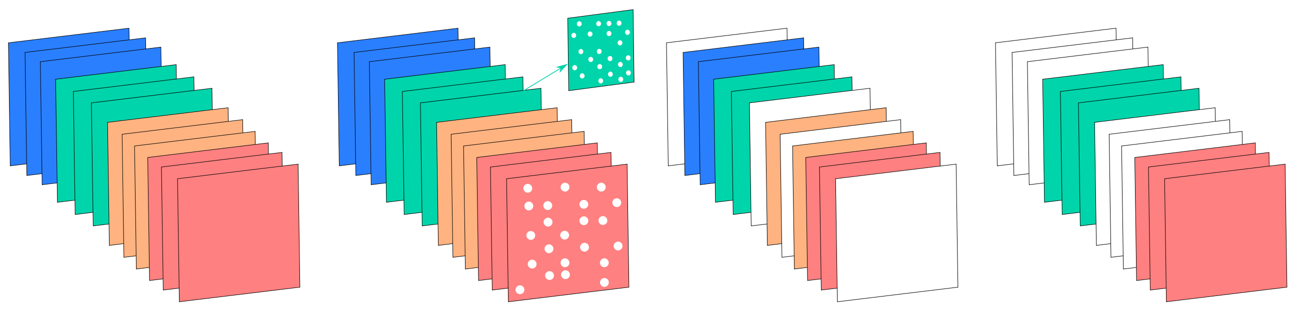
The standard dropout is one kind of unit-wise method, which has been proved useful when applied on fully connected layers [17]. It helps improve the generalization capability by breaking the strong dependence among neurons. However, when applying the same method on convolutional layers, the effectiveness of the standard unit-wise method will be hampered due to the strong spatial correlation among neurons within a feature map — although dropping one neuron can stop other neurons from replying on that particular one, they will still be able to learn from the correlated neurons in the same feature map.

To cope with the spatial correlation, we need the dropout at better granularity. The layer-wise method drops the entire layers, which refer to the outputs from previous layers inside the input tensor. However, the layer-wise dropout is prone to discard too many useful features. The channel-wise method, which will drop a entire feature map at a given probability, strikes a nice trade-off between the two granularity above. Our experiments also confirm that the channel-wise method works the best for regularizing DenseNets.

Meanwhile since the spatial correlation exists in all types of convolutional layers, our analysis above should also work for other CNN models whenever a dropout method is applied.

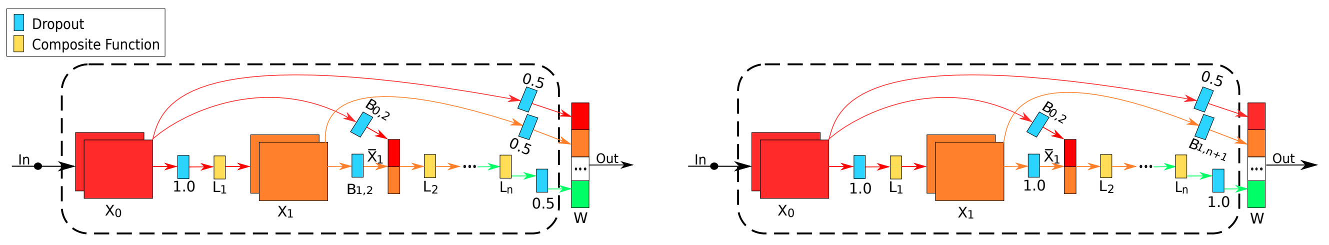
C. Probability Schedule

The channel-wise dropout can still be improved when applied on DenseNets. Notice that the naive channel-wise dropout cannot add various degrees of noises to different layers due to the deterministic survival probability, and since in DenseNets the *dense connectivity* makes the sizes of inputs



(a) Original input (b) Unit-wise dropout (c) Channel-wise dropout (d) Layer-wise dropout

Fig. 3: Different dropout granularity, white color represents dropout.



(a) Schedule v_1 (b) Schedule v_3

Fig. 4: Different probability schedules. The numbers besides blue boxes represent survival probabilities. Left figure shows the linearly decaying schedule v_1 which applies linearly decaying probabilities on different layers of DenseNets. Right figure shows v_3 schedule which applies various probabilities on different portions of the input to a convolutional layer depending on distances between layers generating those portions and the input layer.

at different layers quite different, and such variation seems to be helpful. We believe the model can benefit from such kind of noises. Thus, in our design, to further promote the model generalization ability, besides the pre-dropout and the channel-wise dropout, we also apply the stochastic probability schedule to introduce various degrees of noises to different layers in DenseNets. In the experiments, we have compared the stochastic probability schedule with the deterministic probability schedule, and results show that the stochastic method gets a better accuracy on DenseNets. Since in CNNs the degree of the feature correlation and the importance of features are generally different at different layer depths, such probability schedule would always be desired.

Once we adopt the stochastic method, one natural question arises: *How can we assign probabilities for different convolutional layers in order to achieve the best accuracy?* Actually it is hard to find out the best probability for each layer. However, based on some observations we are still able to design some useful probability schedules. Figure 4 gives examples on the different schedules below. Same as before, we have done some experiments on them and adopt the best one in our design.

One observation is that in the shallow layers of DenseNets, the number of feature maps used as inputs is quite limited and as the layers go deeper the number will become larger. Meanwhile in CNN high-level features are prone to be repeated. Thus intuitively we propose a linearly decaying probability schedule. We refer to it as v_1. For this probability schedule, in each *dense block*, the starting survival probability at the first convolutional layer is 1.0 while the last one is 0.5. Recall that in Section III-A we use $\Theta_{\mathrm{i,j}}$ to represent the tensor of the dropout layer connecting from the output of the layer i to the input of the layer j, in which random variables follow $Bernoulli(p_{i,j})$. So the schedule v_1 will have the following properties,

1) For fixed j and $\forall i$, $p_{i,j} = \mathrm{C}$, C is a constant. Particularly, $p_{i,1} = 1.0$ and $p_{i,n+1} = 0.5$;
2) For fixed i, $p_{i,j}$ is monotonically decreasing with j.

To the best of our knowledge, the dropout will add per unit/neuron different levels of stochasticity depending on the survival probability and the maximum randomness is reached when the probability is 0.5. Meanwhile the sizes of inputs in DenseNets will gradually increase. So to reduce the total randomness in the model, we design another schedule v_2, which is also a reverse version of v_1, i.e., the starting probability for the first layer is 0.5 whereas the last one is 1.0. Similarly, in this schedule, we will have,

1) For fixed j and $\forall i$, $p_{i,j} = \mathrm{C}$, C is a constant. Particularly, $p_{i,1} = 0.5$ and $p_{i,n+1} = 1.0$;
2) For fixed i, $p_{i,j}$ is monotonically increasing with j.

Additionally, we observe that in DenseNets deeper layers tend to rely on high-level features more than low-level features, such phenomenon is also mentioned in [9]. Based on that, schedule v_3 is proposed. For this schedule, we decide survival probabilities for different layers' outputs based on their distances to the convolutional layer, i.e., the most recent output from previous layers will be assigned with the highest probability while the earliest one gets the lowest. In our implementation, for the input to the last layer in one *dense block* we assign the most recent output in it with the probability 1.0 and the least with 0.5. Then based on the number of previous layers concatenated, we can calculate the probability difference between two adjacent layers' outputs, which is further used to decide probabilities for other portions of the input. The corresponding properties of v_3 can be summarized as,

1) $p_{i,j} \propto \frac{1}{d(i,j)}$, where $d(i,j)$ denotes the distance between layer i and j;
2) For fixed j, $p_{i,j}$ is monotonically increasing with i. Particularly, when $i = j - 1$, $p_{i,j} = 1.0$, and $p_{0,n+1} = 0.5$;
3) For fixed i, $p_{i,j}$ is monotonically decreasing with j. Particularly, when $j = i + 1$, $p_{i,j} = 1.0$.

In conclusion, by using v_3 for inputs to deep layers in DenseNets low-level features from shallow layers will always be dropped with higher probabilities whereas high-level features can be kept with a good chance. Meanwhile, survival probabilities for the output of one layer will become smaller as layers go deeper, which is also intuitive as outputs from earlier layers have been used for more times so there exists higher probability that such outputs will not be used again later.

Also notice that the idea to apply different dropout probabilities at different layers can also be extended to other networks since the variation of input sizes at different layers are quite common in CNN models.

IV. Experiments

A. Datasets and Experimental Settings

In our experiments, we mainly use two datasets: CIFAR10 and CIFAR100, containing 50k training images and 10k test images, a perfect size for a normal-sized model to overfit. Meanwhile, we apply a data augmentation on them which includes the horizontal flipping and the translation.

When implementing our models, we try to retain the same configurations of original DenseNets though further hyper-parameter tuning might generate better results since the dropout will slow down the convergence [17]. Briefly four DenseNet structures are chosen: DenseNet-40, DenseNet-BC-76, DenseNet-BC-100 and DenseNet-BC-147. The number at the rear of the structure name represents the depth of the network. The growth rate k for all structures is 12. We adopt 300 training epochs with the batch size 64. The initial learning rate is 0.1 and is divided by 10 at 50% and 75% of the total number of training epochs. During the training stage a fixed survival probability 0.5 is used for non-stochastic dropout methods. The test error is reported after every epoch and all results in our experiments represent test errors after the final epoch.

B. Overall Effectiveness of Specialized Dropout

As an important part of our work, we want to know how DenseNets with our specialized dropout method compare with other models. In this section, we choose three different DenseNet structures and test on CIFAR10 and CIFAR100 datasets.

Note that our specialized dropout will not incur additional parameters in the network. From results in Table I, we can see that DenseNets with our specialized dropout method get the best accuracy on both datasets. In particular, the specialized dropout can consistently have good improvements over the standard dropout method and DenseNet-BC-100 with our specialized dropout, which only contains 0.8M parameters, can outperform a 1001-layer ResNet model. Notice that the data augmentation (which is always applied) already imposes some generalization power to the model, which could make other regularization methods less effective.

Further from the results, our specialized dropout method gives better accuracy improvements on larger dataset CIFAR100, and the improvement of the method also increases with the depth of the network.

C. Experiments on Pre-dropout Structure

In order to show the effectiveness of the pre-dropout structure, we compare the unit-wise pre-dropout method with the standard dropout on two DenseNet structures. Experiments are conducted on CIFAR10 dataset. Results are shown in Table II.

In Table II, the pre-dropout structure achieves better accuracy than the standard dropout on both of the two DenseNet structures. Such results coincide with our analysis that the pre-dropout can incur a better feature-reuse and stimulate more feature patterns in the network. Meanwhile according to the analysis in Section III-B one factor that could potentially disadvantage the pre-dropout here is that in the standard dropout method, even if the dropped units are no longer available in the later layers, the correlated units can compensate for them.

D. Effectiveness of Channel-wise Dropout

In this section, we use the DenseNet-BC-76 structure to compare the three dropout granularity on CIFAR10 and CIFAR100 datasets respectively. The results are shown in Table III.

Table III shows that the channel-wise dropout can achieve the best accuracy on both datasets. So in our specialized dropout method, we adopt the channel-wise dropout granularity. Also from Table III, the layer-wise dropout always has the worst performances. The reason for that could be the layer-wise dropout discards too many useful features at one time, and as a result a accuracy degradation is observed.

E. Experiments on Different Probability Schedules

In Section III-C, we argue that DenseNets would benefit from the variation of noises at different layers. In order to validate this idea, we compare the three stochastic probability schedules proposed to the standard version with a uniform survival probability (0.5) on DenseNet-BC-76. The empirical

TABLE I: Test errors (%) of various network structures and methods

Structure and method	Depth	Params	C10	C100
FitNet [14]	19	-	8.39	35.04
Deeply Supervised Net [13]	-	-	7.97	34.57
Highway Network [18]	-	-	7.72	32.39
ResNet v1 with Stochastic Depth [10]	110	1.7M	5.23	24.58
ResNet v2 [6]	164	1.7M	5.46	24.33
ResNet v2 [6]	1001	10.2M	4.92	22.71
Swapout v2 $W \times 2$ [16]	20	1.09M	5.68	25.86
Swapout v2 $W \times 4$ [16]	32	7.46M	4.76	22.72
DenseNet-BC	76	0.5M	5.21	24.09
DenseNet-BC(standard dropout)	76	0.5M	5.56	24.75
DenseNet-BC(our specialized dropout)	76	0.5M	4.94	23.90
DenseNet-BC	100	0.8M	4.73	23.22
DenseNet-BC(standard dropout)	100	0.8M	5.01	23.80
DenseNet-BC(our specialized dropout)	100	0.8M	4.51	22.33
DenseNet-BC	148	1.5M	4.31	20.76
DenseNet-BC(standard dropout)	148	1.5M	4.60	22.28
DenseNet-BC(our specialized dropout)	148	1.5M	**3.90**	**19.75**

TABLE II: Unit-wise pre-dropout vs standard dropout

r

	Test error (%)	
Structure	standard dropout	Pre-dropout
DenseNet-40	5.83	**5.75**
DenseNet-BC-76	5.56	**5.25**

TABLE III: Comparisons of different dropout granularity

	Test error (%)	
Method	C10	C100
Unit-wise dropout	5.25	24.52
Layer-wise dropout	5.46	25.28
Channel-wise dropout	**5.09**	**24.36**

study indicates that v_3 is always the best among the three schedules, whereas v_2 is the worst. Thus, we pick the schedule v_3 in our final specialized dropout method.

Table IV gives an example result for DenseNet-BC-76 on

TABLE IV: Comparisons of various probability schedules

Method	Error (%)
Channel-wise dropout (uniform 0.5)	5.09
Channel-wise dropout with v_1	5.02
Channel-wise dropout with v_2	5.13
Channel-wise dropout with v_3	**4.94**

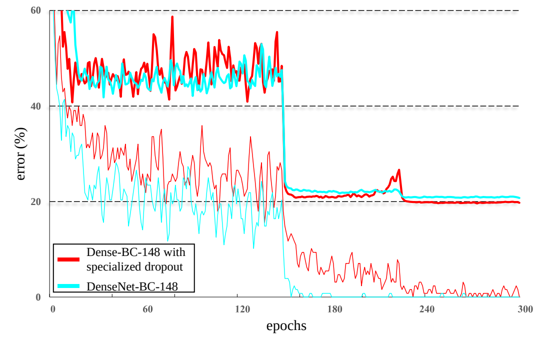


Fig. 5: Training progress on CIFAR100. Thin curves denote the training errors, and bold curves denote the test errors.

CIFAR10 dataset. Recall that the number of feature maps to shallow layers in DenseNets is very limited and the schedule v_2 applies lower survival probabilities on these layers. From the results we can see although v_2 reduces the total randomness in the model, a loss of relatively larger quantity of low-level features can still hurt the accuracy.

Furthermore, we notice that the effect of the schedule v_1 also exists in v_3, i.e., relatively higher survival probabilities are assigned for shallow layers and lower ones for deep layers. In addition, v_3 can also help deep layers rely more on the high-level features, which could be the reason making v_3 better than v_1.

F. Regularization Effect

We also want to figure out how the specialized dropout method contributes to the model generalization ability. To analyze this, we compare the training/test errors during the

TABLE V: Effectiveness of the specialized dropout method on other state-of-the-art CNN models. The model names in bold denote models with the specialized dropout method. Results are reported as test errors.

Model	Depth	C10	C100
AlexNet	8	10.32	38.71
AlexNet	8	**9.78**	**35.42**
VGG-16	16	8.39	35.04
VGG-16	16	**7.25**	**33.71**
ResNet v1	110	6.41	27.22
ResNet v1	110	**5.45**	**25.46**
ResNet v2	164	5.46	24.33
ResNet v2	164	**4.38**	**23.36**

training procedures of a normal DenseNet model and the one with our specialized dropout method. Figure 5 shows such comparison on DenseNet-BC-148. As shown in the figure, the specialized dropout version reaches a slightly higher training error at convergence, but produces a lower test error. This phenomenon indicates that the improvement of the accuracy comes from the stronger regularization effect brought by the specialized dropout method, which also supports the point that the specialized dropout method can actually improve the model generalization ability.

G. Specialized Dropout on Other CNN Models

Following the idea of designing a specialized dropout method for DenseNets, we also want to explore whether such idea could also apply to other state-of-the-art CNN models. Here we choose AlexNet, VGG-16 and ResNet to conduct the experiments. Similar to the DenseNets, we design the specialized dropout method for each model from three aspects. We apply the pre-dropout structure and the channel-wise granularity for all specialized dropout methods and decide the survival probability for each layer by the size of the input. In order to reduce the total randomness in a model, the largest input will have the survival probability of 1.0 while the smallest one corresponds to the probability of 0.5. Layers between the two will follow a linear increasing/decreasing schedule to assign the survival probabilities. The results are shown in Table V.

From results in Table V we can see that all models with the specialized dropout method outperform its original counterparts, which indicates that our idea to design a specialized dropout method will also work in other CNN models. The effectiveness of our idea is also validated by such results.

V. Conclusion

In this paper, first we show problems of applying the standard dropout method on DenseNets. To deal with these problems, we come up with a new pre-dropout structure and adopt the channel-wise dropout granularity. Specifically, we put the dropout layers before convolutional layers to reinforce the feature-reuse inside the model. Meanwhile we randomly zero out some feature maps in the inputs of convolutional layers to break the dependence among them. Besides, to further promote the model generalization ability we introduce stochastic probability schedules to add various degrees of noises to different layers in DenseNets. Experiments show that in terms of the accuracy DenseNets with our specialized dropout method outperform all other CNN models and the idea to design a specialized dropout method also works in other CNN models.

References

[1] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 249–256, 2010.

[2] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pages 315–323, 2011.

[3] Johan Håstad. *Computational Limitations of Small-depth Circuits*. MIT Press, Cambridge, MA, USA, 1987.

[4] Johan Håstad and Mikael Goldmann. On the power of small-depth threshold circuits. *Computational Complexity*, 1(2):113–129, 1991.

[5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *European Conference on Computer Vision*, pages 630–645. Springer, 2016.

[7] Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012.

[8] Gao Huang, Danlu Chen, Tianhong Li, Felix Wu, Laurens van der Maaten, and Kilian Q Weinberger. Multi-scale dense convolutional networks for efficient prediction. *arXiv preprint arXiv:1703.09844*, 2017.

[9] Gao Huang, Zhuang Liu, Kilian Q Weinberger, and Laurens van der Maaten. Densely connected convolutional networks. *arXiv preprint arXiv:1608.06993*, 2016.

[10] Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Q Weinberger. Deep networks with stochastic depth. In *European Conference on Computer Vision*, pages 646–661. Springer, 2016.

[11] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, pages 448–456, 2015.

[12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural

networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[13] Chen-Yu Lee, Saining Xie, Patrick Gallagher, Zhengyou Zhang, and Zhuowen Tu. Deeply-supervised nets. In *Artificial Intelligence and Statistics*, pages 562–570, 2015.

[14] Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. Fitnets: Hints for thin deep nets. *arXiv preprint arXiv:1412.6550*, 2014.

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

[16] Saurabh Singh, Derek Hoiem, and David Forsyth. Swapout: Learning an ensemble of deep architectures. In *Advances in neural information processing systems*, pages 28–36, 2016.

[17] Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of machine learning research*, 15(1):1929–1958, 2014.

[18] Rupesh K Srivastava, Klaus Greff, and Jürgen Schmidhuber. Training very deep networks. In *Advances in neural information processing systems*, pages 2377–2385, 2015.

[19] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[20] Jonathan Tompson, Ross Goroshin, Arjun Jain, Yann LeCun, and Christoph Bregler. Efficient object localization using convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 648–656, 2015.

[21] Li Wan, Matthew Zeiler, Sixin Zhang, Yann Le Cun, and Rob Fergus. Regularization of neural networks using dropconnect. In *International Conference on Machine Learning*, pages 1058–1066, 2013.

[22] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016.

[23] Ke Zhang, Miao Sun, Xu Han, Xingfang Yuan, Liru Guo, and Tao Liu. Residual networks of residual networks: multilevel residual networks. *IEEE Transactions on Circuits and Systems for Video Technology*, 2017.