Groupout: A Way to Regularize Deep Convolutional Neural Network

Eunbyung Park
Department of Computer Science
University of North Carolina at Chapel Hill
eunbyung@cs.unc.edu

Abstract

Groupout is a new technique of regularization for deep convolutional neural network. It is known that the dropout technique works very well by randomly omitting half of hidden units on each training case. However, sometimes it performs poorly when we apply this technique to convolutional neural network. Inspired by the fact that all hidden units in a convolutional feature map are activated by specific patterns, I group the all hidden units in a specific feature map and drop all of them instead of dropping out individual units. Using CIFAR datasets, I demonstrate that proposed method shows reasonable amount of improvement in terms of prediction errors.

1 Introduction and Motivation

Convolutional neural network(CNN)s has been remarkably successful across almost every computer vision tasks, such as classification [7], detection [3], segmentation [10], and so on. Its mainly because of good representation (or image features) learnt by CNN and the key enabling factors are the increase of computational power and big labeled datasets [2], which allow us to scale up the networks to tens of millions of parameters [7, 12]. In addition, there is easily accessible public software [6] so that many researchers can easily develop and train their own CNN architectures for different purposes.

One of the challenging problem of training deep neural network is to avoid overfitting to datasets. Recently, dropout technique was proposed for regularizing deep feed-forward neural network [5]. On each presentation of each training case, each hidden unit is randomly omitted from the network with a probability of 0.5, so a hidden unit cannot rely on other hidden units being present. In other words, we can force each hidden units to be independent each other. In addition, the dropout procedure is known as a very smart way of doing model averaging with neural networks because it approximates the average outputs of many different separate networks.

However, applying dropout technique to convolutional layers is not commonly recommended when it comes to training deep and large network. It usually gives us poor performance than the model trained without dropout. There are a few of works that argued that dropout over convolutional layers also gives us additional performance improvement. However, its experiments were limited over relatively small size datasets and networks [11]. What people usually have been doing when they try to train large and deep CNNs is to apply dropout to the last two or three fully connected layers [7]. It turns out that this method achieved state-of-art results for most of recent CNN architectures.

While I have been working on this idea, same work has been done by other people and they uploaded the paper on arXiv on 16th Nov 2014 [13]. It suggested exactly same idea(but in different context), which is called 'spatial dropout'. Even if the idea of this project is no longer novel, it is still worth writing a report because I have different interpretation and performed different experiments.

For convolutional layers, what we've done so far in terms of regularization is to do weight decay method. It seems that the weight decay has prevented convolutional filters from growing too much. Another regularization methods for deep CNNs is data augmentation. Horizontal flipping and cropping random patches out of an image lead us to have bigger dataset, which also gives us substantial performance improvements. Form my understanding, however, it is a trick that makes the network try to overfit over augmented dataset. Although we can think of this as regularization, it has a problem that training procedure gets slower according to the size of augmented dataset.

In this project, I propose a new way to regularize deep CNNs. I drop out all hidden units in a convolutional feature map together by given probability, instead of dropping out each hidden unit individually. By doing this we introduce additional constraints into convolutional layers so that each feature map tries to be independent each other. In same point of view in dropout technique, each feature map wont be able to rely on other feature maps to explain training examples. In other words, each feature map would like to keep as much information as possible in the process of training.

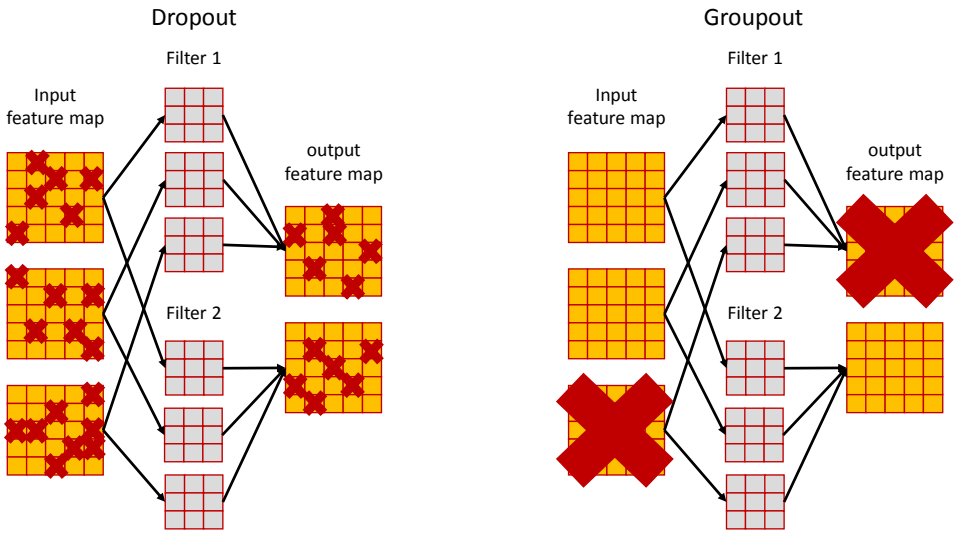

Figure 1: Dropout vs Groupout over convolutional layers

1.1 Problem of dropout over convolutional layer

Recently, there has been many works that support the idea of proposed technique. It turns out that a feature map, which are a group of hidden units activated by a convolutional filter, has high activation values when only similar input pattern occurs [15, 3]. In addition, this contains spatial information, which means that a hidden unit located at specific position in a feature map gives us high activation value when the pattern showed up at the specific position in an input image. For example, let's assume that there is dog head detector feature map. If a dog was on left bottom side of an image, hidden units on left bottom side of a feature map will have high activation values. In short, each hidden unit in a feature map has same functionality.

Hence, to dropout individual hidden units in a feature map seems to have no strong justification. In this way we are just introducing another data augmentation, such as noisy versions of an input image. It gave us additional performance improvement over small dataset and relatively shallow network, but it performed poorly over large dataset and deep CNN. I guess the reason why this hurt performance is because additional noisy versions of input image make dataset way larger. Therefore, the network were not large and deep enough to be trained over large augmented dataset. As input images pass through convolutional layers with dropout they became too noisy or they lost important information most of time in the process of training

1.2 Grandmother convolutional filter

Neuroscientists have believed that there are grandmother cells(GMC) in our brain. They only respond to very specific and complex visual stimuli, such as the face of ones grandmother. In computer vision community, they have been trying to find good mid-level patches and encoding images in terms of them. The problem of finding good mid-level patches is often posed as a search for a set of high-recall discriminative templates, which is somewhat related to find GMC cell. This makes it interesting to investigate the nature of CNN features such as last or last two convolutional layers. [1] recently showed that there are a few of GMC filters by finding a feature map that fire strongly on all images of one object and nothing else.

As [1] has shown, there are only few of GMC like filters in current pre-trained deep CNNs. In the paper, they concluded that there were GMC filters for only bicycle, person, car, and cat. It would give better performance if we could have more GMC like filters. In addition, it is widely accepted view that the more sparsity on your representation the better accuracy we can get at least for the object classification or detection. The groupout technique proposed in this project will help to find more GMC like convolutional filters. Each feature maps under groupout regularization will be trying not to rely on each other, and it will eventually end up with having highly GMC like feature representations.

2 Description of Groupout

In this section, I present concrete description of dropout and groupout over convolutional layers. For general fully connected layers, you can find it in the dropout paper [11].

2.1 Dropout over convolutional layer

The convolution of the input map x with k filters **f** and biases **b** are defined as following.

$$y_{i''j''k} = b_k + \sum_{i'=1}^{H'} \sum_{j'=1}^{W'} \sum_{d=1}^{D} f_{i'j'd} \times x_{i''+i',j''+j',d}$$

,where $\mathbf{x} \in \mathbb{R}^{H \times W \times D}$ is input map, where H is length of height, W is length of width, and D is the number of channels. $\mathbf{y} \in \mathbb{R}^{H'' \times W'' \times K}$ is output map, where H'', W'' is size of map after convolution operations. $\mathbf{f} \in \mathbb{R}^{H' \times W' \times D \times K}$ is convolutional filter, where H', W' is size of filter and K is the number of filters. If we apply dropout to individual hidden units in a feature map, we have

$$y_{i''j''k} = b_k + \sum_{i'=1}^{H'} \sum_{j'=1}^{W'} \sum_{d=1}^{D} f_{i'j'd} \times x_{i''+i',j''+j',d} \times \delta_{i''+i',j''+j',d}$$

,where $\delta \in \{0,1\}^{H \times W \times D}$, which each element of δ is independently and identically distributed bernoulli trials. In practice, we can implement this as simple matrix multiplication with binary mask matrix, which is same size as input maps.

2.2 Groupout of hidden units

The convolution with the groupout method is following.

$$y_{i''j''k} = b_k + \sum_{i'=1}^{H'} \sum_{j'=1}^{W'} \sum_{d=1}^{D} f_{i'j'd} \times x_{i''+i',j''+j',d} \times \delta_d$$

, where $\delta \in \{0,1\}^{D}$. In this case, all elements in a input feature map will be multiplied by same δ_d.

Table 1: Model description

Model	Description
A	Dropout over input, conv(1,2,3) and fc(4,5) layers
B	Dropout over fc(4,5) layers
C	Groupout over conv(2,3) and fc(4,5) layers
D	Groupout over conv(2,3) and fc(4,5) layers, bigger model
E	Dropout over input layer, groupout over conv(2,3) and fc(4,5) layers
F	Dropout over input layer, groupout over conv(2,3) and fc(4,5) layers, bigger model

2.3 Groupout of weights

Inspired by dropconnect [14], which is a variant of dropout technique that randomly omits the weights instead of hidden units. We can also apply our grouping argument to convolutional filters. Basically convolutional filters for each layer is 4 dimentional tensor and there are K convolutional filters, where each filter consists of D matrices, where dimension of each matrix is $H' \times W'$. So, instead of dropping out individual elements in a $H' \times W'$ matrix, we drop an entire matrix.

2.4 Training and testing networks with dropout and groupout

We can use the standard, stochastic gradient descent procedure for training the network with dropout and groupout. In practice, when we are training networks, we multiplied outgoing weight by $1/p$ if a unit is dropped with probability p. During test time, we can simply use a single neural net without dropout and it is known that it approximates average the predictions of exponentially many network with dropout. you can find more details behind this in dropout paper [11].

3 Experiment

3.1 Experimantal setup

Experiments were performed over CIFAR-10 datasets. It consist of 32x32 color images(3 input channels) and 10 categories. Some of paper mentioned that data pre-processing(ZCA whitening and global contrast normalization) gave additional performance gain, but I didnt preprocess data. All the networks are trained with 0.9 momentum and 0.001 weight decay I trained network 100000 iteration with batch size 100(200 epochs). I adjusted learning late at 50000 iteration from base learning rate 0.001 to 0.0001. I couldnt explore hyper-parameter space because of the time constraint. Finally I ran all the training on caffe framework [6].

I followed standard CNN architecture used in [11]. It uses three convolutional layers each followed by a max-pooling layer. The convolutional layers have 96, 128, and 256 filters(bigger version has 96, 256 and 384 filters) respectively. Each convolutional layer has a 5x5 receptive filed applied with a stride of 1 pixel. Each max pooling layer pools 3x3 regions at strides of 2 pixels, which cause 2 times down-sampling. The convolutional layers are followed by two fully connected hidden layers having 2048 units each. Dropout was applied to all the layers of the network with the probability of p = (0.1,0.25,0.25,0.5,0.5,0.5) for the different layers of the network(going from input layer to last fully connected layer).

3.1.1 Results

I exhibit experimental results. They are self-explnatory.

3.1.2 Discussion

As you can see, usually groupout gave 3 4 percent improvement in terms of accuracy. Even though the experiments are limited in small CIFAR dataset, we can still say that it is not neglectable improvement. Note that the accuracy I got from the experiments is not as high as ones in the dropout

Table 2: Accuracy on testset

Model	CIFAR10	CIFAR100
A	0.829	0.520
B	0.818	0.552
C	0.858	0.593
D	0.856	0.595
E	0.852	0.593
F	0.857	0.594

paper. It is because I didn't perform any data-preprocessing and data augmentation. In general, those are ginving additional improvements so we can expect that the accuracy number described here would increase. In addition, because of time constraint, I couldn't explore many hyperparameter choices, so it also might give us more improvement.

This methods are orthogonal to recent works that have achieved state-of-art results on CIFAR dataset [8, 9, 4]. So, it is easy to combine groupout to existing technuque and it also might give more improvement.

4 Conclusion

In this project, I propose new regularization technique for deep CNNs. Experimental results have shown that it outperforms original dropout method.

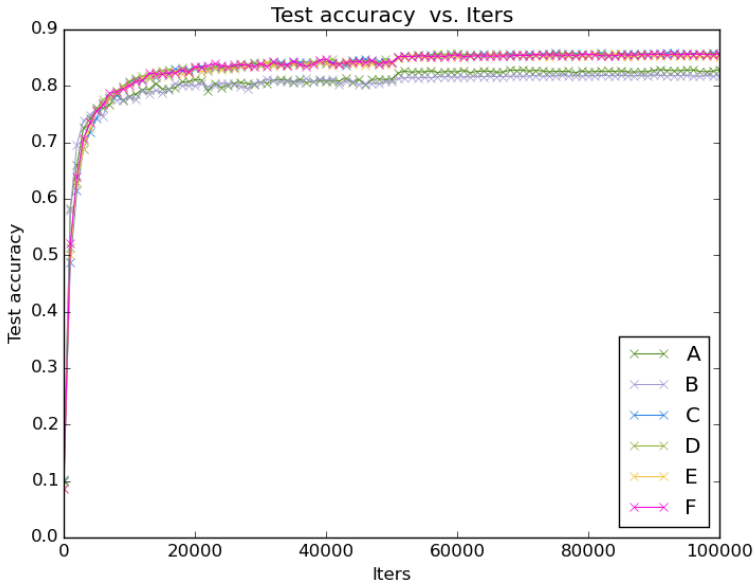

Figure 2: accuracy vs iteration on CIFAR10

References

[1] Pulkit Agrawal, Ross Girshick, and Jitendra Malik. Analyzing the performance of multilayer neural networks for object recognition. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2014.

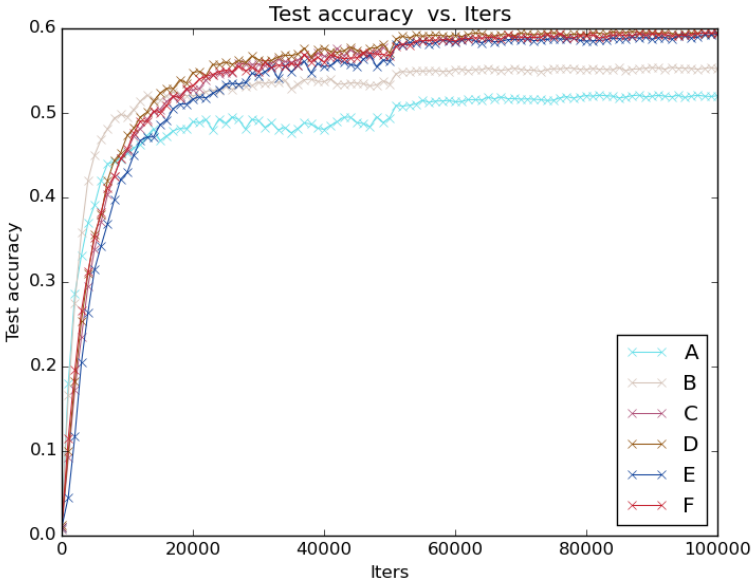

Figure 3: accuracy vs iteration on CIFAR100

[2] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.

[3] Ross B. Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. *CoRR*, abs/1311.2524, 2013.

[4] Ian J. Goodfellow, David Warde-Farley, Mehdi Mirza, Aaron C. Courville, and Yoshua Bengio. Maxout networks. In *Proceedings of the 30th International Conference on Machine Learning, ICML 2013, Atlanta, GA, USA, 16-21 June 2013*, pages 1319–1327, 2013.

[5] Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *CoRR*, abs/1207.0580, 2012.

[6] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014.

[7] Alex Krizhevsky, Sutskever Ilya, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C.J.C. Burges, L. Bottou, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.

[8] Chen-Yu Lee, Saining Xie, Patrick Gallagher, Zhengyou Zhang, and Zhouwen Tu. Deeply-supervised nets. *CoRR*, abs/1409.5185, 2014.

[9] Min Lin, Qiang Chen, and Shuicheng Yan. Network in network. *CoRR*, abs/1312.4400, 2013.

[10] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. *CoRR*, abs/1411.4038, 2014.

[11] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014.

[12] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. *CoRR*, abs/1409.4842, 2014.

[13] Jonathan Tompson, Ross Goroshin, Arjun Jain, Yann LeCun, and Christoph Bregler. Efficient object localization using convolutional networks. *CoRR*, abs/1411.4280, 2014.

[14] Li Wan, Matthew Zeiler, Sixin Zhang, Yann L. Cun, and Rob Fergus. Regularization of neural networks using dropconnect. In Sanjoy Dasgupta and David Mcallester, editors, *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, volume 28. JMLR Workshop and Conference Proceedings, May 2013.

[15] Matthew D. Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. *CoRR*, abs/1311.2901, 2013.