Comparing Regulation of Dropout, DropConnect and DropBlock

Guanyu Song
University of Toronto

Jie Yang
University of Toronto

Wenzhi Lin
University of Toronto

Abstract

Dropout method is one of the most commonly used regulation for reducing overfitting of neural networks. In this paper we look into the performance of CNN with Dropout and its generalizations, DropConnect and DropBlock. We found that while all three methods can significantly reduce overfitting, generally it is more effective to implement DropBlock on a convolutional layer than Dropout and DropConnect on a fully connected layer.

1 Introduction

Large neural networks with comparatively small data set often leads to overfitting. Many regulation methods such as l_1 and l_2 regularizations, data augmentations, and early stopping are introduced to solve the problem. It has been shown that large networks trained with regulations performs way better than small networks trained without regulations **[3]**.

One of the most effective methods is Dropout. During the forward propagation, a portion of the output neurons are chosen and deleted in each layer. The error is then backpropagated only through the remaining activations. DropConnect **[1]** works similarly by deleting weights instead of output neurons. The intuition behind is that it trains the weights separately with certain features omitted to prevent the network from memorizing the training data set.

Dropout and DropConnect are known to work best with fully connected layers. For convolutional layers, information can still flow through even when some activations or weights are deleted. In this case, we can use the DropBlock **[2]** method, where features in a block, i.e., a contiguous region of a feature map, are dropped together.

2 Related works

Dropout: Dropout is a regularization method to reduce the overfitting in neural networks. Individual nodes are dropped out of the network with probability p or kept with probability 1-p at each training stage, resulting in a smaller network. The incoming and outgoing edges to a dropped-out node are likewise eliminated. Mendenhall and Meiiler demonstrate the power of Dropout to improve model performance in many deep learning domains using ANN models with Dropout.[3]

DropConnect: DropConnect is the generalization of Dropout in which each connection, rather than each output unit, can be dropped with probability p. DropConnect is similar to Dropout as it introduces dynamic sparsity within the model, but differs in that the sparsity is on the weights W, rather than the output vectors of a layer [1]. The author of "The Regularization of Neural Networks using DropConnect "trains and evaluates the datasets Minist, NORB, CIFAR-10, and SVHN by using the CNN model introduced with DropConnect. The experiment shows that DropConnect generally outperforms DropOut in large neural models.

Preprint. Under review.

DropBlock: The paper "Drop-Block: A regularization method for convolutional networks" introduces the regularization method DropBlock, a simple method similar to DropOut. The main difference between it and DropOut is that it drops contiguous regions in the layer's feature map instead of dropping independent random units. The observation of the CNN model introduced with DropBlock successfully proved that DropBlock is a more effective regularization tool than dropout, making the model more robust.

3 Method / algorithm

3.1 Dataset

The CIFAR-10 dataset consists of 60000 32x32 colour images in 10 classes, with 6000 images per class. We use the 50000-image training set for training split the 10000-image test set into validation set and test set, each containing 5000 images.

3.2 Neural Network Architect

In our model[1], we use three convolutional layers for feature extracting. Each layer contains a 2d convolution operation, a Relu activation, and a max-pooling. Followed by the feature extraction, the output is reshaped and sent to a fully-connected layer with Relu activation function and another fully-connected layer with no activation. The network is illustrated in the **Figure 4** in Appendix. Cross-entropy loss is used to evaluate our model performance and ADAM optimizer is used to train the model.

3.3 Regulation Implementation

For Dropout, we use the PyTorch implementation [2], which randomly zeroes elements of the input tensor using Bernoulli distribution. Furthermore, the outputs are scaled by a factor of $\frac{1}{1-p}$ during training. We apply Dropout to the output of **fc1**, the first fully connected layer in our network.

For easier implementation, we use a different approach than the original paper **[3]**, where $\mathbf{o}$ is the output of **fc1**, $\mathbf{M}$ is the mask matrix, $\mathbf{W}$ is the weight matrix of **fc1**, and $\mathbf{u}$ is the input of **fc1**. In our implementation, instead of masking weights of **fc1**, we construct a new linear layer **wd1** between the CNN and **fc1**. The weights in **wd1** are randomly zeroed out during training, and do not update during backpropagation (**wd1** is a constant matrix). In other words, $\mathbf{o} = \mathbf{W}(\mathbf{M}\mathbf{W_1})\mathbf{u}$, where $\mathbf{W_1}$ is the weight matrix of **wd1** and is constant. The implementation is modified based on the torchnlp.nn.WeightDrop [3].

For DropBlock, we use the existing implementation on Github by Miguel Varela Ramos [4]. Similar to Dropout, DropBlock randomly applies a mask $\mathbf{M}$ using Bernoulli distribution of γ, where $\gamma = \frac{drop\_prob}{block\_size^3}$ [5]. For each zero position $\mathbf{M_{ij}}$, a spatial square mask, a.k.a the block, is created with the center being $\mathbf{M_{ij}}$, the width, height being the block size and set all the values of $\mathbf{M}$ within the square to be zero. $\mathbf{M}$ is then applied to activations of a layer $\mathbf{o}$ and the result is normalized with the ratio of $\frac{entries\_M}{unmasked\_entries\_M}$. Furthermore, $drop\_prob$ is scheduled: $drop\_prob$ is gradually increased from 0 to the target value with a linear scheme. We apply dropblock on the output of **conv2**, the second convolutional layer in our network. The complete algorithm from the paper for DropBlock is here:

[1] This model is modified based on the starter code provided by Wandb, `https://wandb.ai/authors/ayusht/reports/Implementing-Dropout-in-PyTorch-With-Example--VmlldzoxNTgwOTE`

[2] `https://pytorch.org/docs/stable/generated/torch.nn.Dropout.html`

[3] `https://pytorchnlp.readthedocs.io/en/latest/_modules/torchnlp/nn/weight_drop.html`

[4] `https://github.com/miguelvr/dropblock`

[5] The calculation of γ is slightly different than the original paper [2], but plays the same role of controlling the number of features to drop.

Algorithm 1 DropBlock

Input:output activations of a layer (A), $block\_size$, γ, $mode$
if $mode$ == *Inference* **then**
    return A
end if
Randomly sample mask M: $M_{i,j} \sim Bernoulli(\gamma)$
For each zero position $M_{i,j}$, create a spatial square mask with the center being $M_{i,j}$, the width, height being $block\_size$ and set all the values of M in the square to be zero (see Figure 2).
Apply the mask: $A = A \times M$
Normalize the features: $A = A \times$ **count**(M)/**count_ones**(M)

Figure 1: Complete algorithm for DropBlock **[2]**

3.4 Procedure

Training and testing for the model with different and no regulation methods will be conducted and analyzed. The training and validation curves of each model will be compared. The effect of hyperparameters will also be explored.

4 Experiments and Discussion

4.1 Performance across Methods

In this experiment we compare the performance of model with different regulations. We set $batch\_size = 64, learning\_rate = 0.0003, epochs = 45, drop\_rate = 0.5$. The complete result of validation and test loss and accuracy are shown in Table 1 in the Appendix.

The behaviour of each regulation method is shown in the diagrams in **Figure 2**. To summarize, DropBlock > DropConnect > Dropout > Unregulated, ranking from slowest to fastest training loss and accuracy convergence; DropBlock > DropConnect > Dropout > Unregulated, ranking from best to worst validation loss and accuracy.

Therefore, it's safe to conclude that the regulation methods effectively reduce overfitting by preventing the model from memorizing the training data. More effective regulation methods not only keep the validation loss lower, but also keep it more steady. The strength of the methods goes: DropBlock> DropConnect > Dropout, ranking from most effective to the least.

Both Dropout and DropConnect allows us to train different variations of the network at the same time and average out the results, thus help us improve the generality. The reason why DropConnect performs better than Dropout in our case may be that DropConnect leads to more possible variation for the model, which creates more noise and better helps the model to generalize.

DropBlock outperforms both DropConnect and Dropout in this case likely because it filters the information flow in a more structural way and is applied to a more critical layer. More details will be discussed in secion 4.3.

When the experiment is conducted with different hyperparameters, the general shape of the curves remain the same, that is, stronger regulation methods leads to slower training convergence and better validation performance. However, sometimes stronger regulation have similar lowest validation loss with weakly regulated model, despite of the fact that the validation curve of the weak one rises earlier and more rapidly than the strong one (see **Figure 5** in Appendix). In this case, regulation methods still have their value as they have more stable validation curve and require less data to determine the stopping point of training.

4.2 Performance with Different Batch Size and Learning Rate

Some results of validation performance of different models with various batch size is shown in Appendix A Table 2. The general trend for performance is as expected, that all models favour small

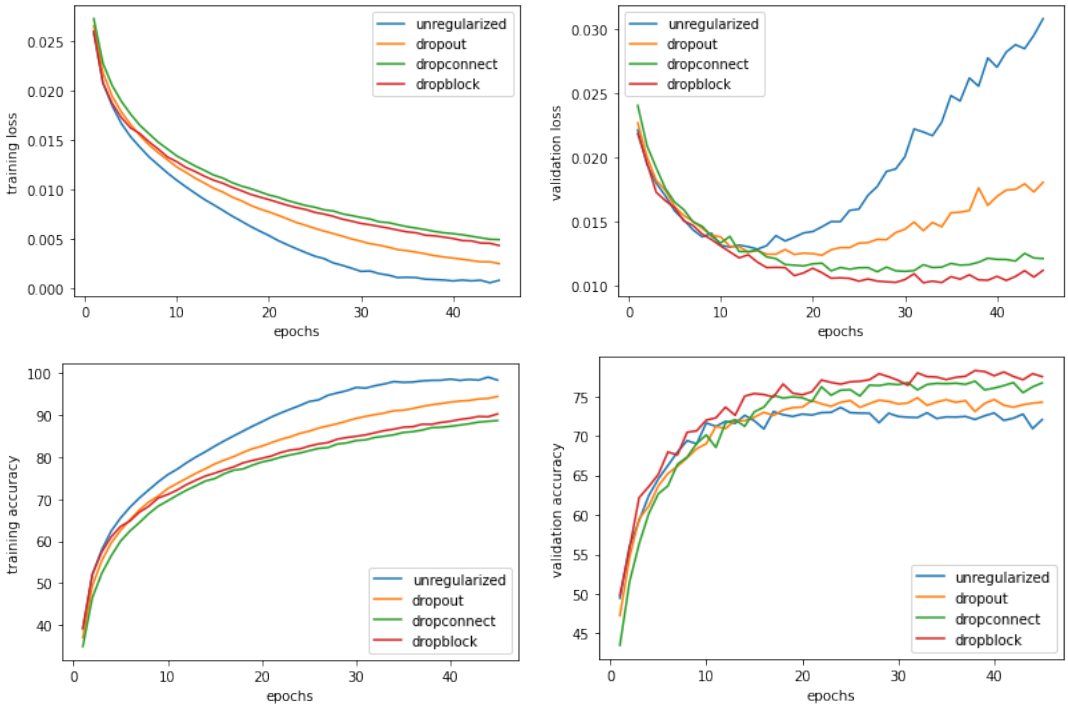

Figure 2: The training loss, validation loss, training accuracy, and validation accuracy curves for unregulated model, model with dropout, dropconnect, and dropblock.

batches. Training with larger batch size tends to converge slower, and have less overfitting compared to smaller batch size.

Some results of validation loss of different models with various learning rate is shown in Appendix A Table 3. The performance of each model rises first and then falls as the learning rate increases. Training with smaller learning rate tends to converge slower, and regulated model tends to favour smaller learning rate. It may because the larger fluctuation brought by large learning rate may accidentally result in a better performance for unregulated model, while it has a smaller impact on regulated models as their performance is more steady.

4.3 Dropout on Convolutional Layer

It is known that dropout method works best on fully connected layer. So what happens when it is applied to a convolutional layer? Surprisingly, our model performs better when Dropout is applied to convolutional layer compared to fully connected layer. With $batch\_size = 64, learning\_rate = 0.0003, epochs = 45, drop\_rate = 5$, the validation loss with Dropout on **fc1** is 0.0130 while with Dropout on **conv2** is 0.0114. The experiment is conducted for several times and it all shows similar results. The reason may be in our model the convolution layer plays a more important role and its weights are more sensitive. So regulations applied to the convolutional layer introduces more noise, which reduces the overfitting more effectively.

However, when both applying to **conv2**, DropBlock outperforms Dropout (see **Figure 3**). This is because although both methods drops the same number of features, DropBlock does it more structurally. In the convolutional layer, features in the same neighbourhood carries partially overlapped information, so when randomly dropped like Dropout does, information still flows through the features near the dropped one. Meanwhile, when DropBlock drops a feature, it also drops (almost, depending on the block size) all features correlated to it.

Furthermore, the drop rate for DropBlock is scheduled that it gradually increase to the target value, while the drop rate for Dropout is fixed. Smaller drop rate at the beginning helps the model to reach a low validation loss at first, so while slowing down the overall convergence it doesn't hold up the validation curve drop at the beginning.

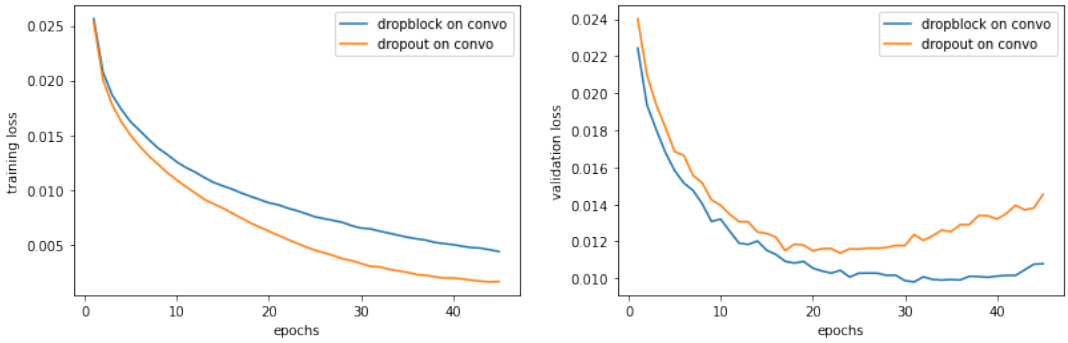

Figure 3: The training loss and validation loss curve for dropout on **conv2** and dropblock on **conv2**. $batch\_size = 64, learning\_rate = 0.0003, epochs = 45, drop\_rate = 0.5$. DropBlock reduces overfitting further than Dropout.

4.4 Exploration: Combination of Regulations

We conducted the same experiment as 4.1 on a slightly modified network to see if the same results come out. The new network is the same as before except we added Batch Norm regulation for each convolutional layer. The training and validation curves are shown in **Figure 6** in Appendix A. It turns out that with the batch norm regulation, the difference brought by different dropping methods is even more obvious. The model with batch norm regulation also outperforms the one without.

We also tried to combine different regulations together, for example, DropBlock on **conv2** and Dropout on **fc1** (**Figure 7**). The result is that different regulations in different layers work independently and the effects of them add up to further reduce overfitting. It is also found that multiple layers of Dropout/DropConnect/DropBlock of small dropping rate work better than single layer of them of large dropping rate, which can be explained by that more layers leads to more variations of the net.

4.5 Summary and Further Discussion

In this project we demonstrated the strength of Dropout, DropConnect, and DropBlock regulation. We analyzed the reasons behind their effectiveness and the role of hyperparameters. The conclusion is, in general, while all methods significantly reduce overfitting, DropBlock outperforms DropConnect and DropConnect outperforms Dropout.

Limited by time and computational resources, we only ran each of our experiment once instead of running multiple times and average them. Randomness of the results may cause unwanted bias, especially when analyzing the hyperparameters. In addition, we realize that our implementation to calculate the loss is to some extent dependent on the batch size of the data. Because of that, when testing different batch sizes, we compared validation and test accuracy instead of loss, which is a better measurement. By the limitation of time and resource, we did not implement the original version of DropConnect, where the DropConnect layer also participates in weights updating. Going through the fixed sparse matrix in our implementation may increase the correlation of the gradient, which may slow convergence.

The power of DropBlock shows the effects of scheduling. Applying Dropout and DropConnect with drop rate that changes along the properties (eg., magnitude, distribution) of the weights could be a good direction for further improvements.

References

References follow the acknowledgments. Use unnumbered first-level heading for the references. Any choice of citation style is acceptable as long as you are consistent. It is permissible to reduce the font size to `small` (9 point) when listing the references. Note that the Reference section does not count towards the page limit.

[1] Wan, L., Zeiler, M., Zhang, S., Lecun, Y., Fergus, R. (n.d.). Regularization of Neural Networks using DropConnect. Retrieved April 20, 2022, from http://proceedings.mlr.press/v28/wan13.pdf.

[2] Golnaz, G., Google, B., Tsung-Yi, L., Quoc, V., Le Google Brain. (n.d.). DropBlock: A regularization method for convolutional networks. Retrieved April 20, 2022, from https://proceedings.neurips.cc/paper/2018/file/7edcfb2d8f6a659ef4cd1e6c9b6d7079-Paper.pdf

[3] Mendenhall, J., Meiler, J. (2016). Improving quantitative structure–activity relationship models using Artificial Neural Networks trained with dropout. Journal of Computer-Aided Molecular Design, 30(2), 177–189. https://doi.org/10.1007/s10822-016-9895-2.

A Appendix

A.1 Contributions

Guanyu Song:

- Constructed the network, the training / testing process and analysis tools
- Implemented DropConnect and DropBlock with Jie Yang
- Designed and set up the experiments and ran them
- Organized and analyzed the experimental results
- Wrote section 3 and 4 of the report
- Styling of the final report

Jie Yang:

- Define the first version of implementations for DropBlock and DropConnect
- Found useful data set interface together with Guanyu Song
- Running experiments
- Generated the reference, organized and wrote the related work section for the report
- Provided illustrations for the network
- Organize meetings and communications

Wenzhi Lin:

- Formulated the introduction section of the report
- Data processing
- Ran the experiments
- Put together the experimental results
- Helped with the understanding and implementation of the regulation methods
- Helped with analysis of the experimental results

A.2 Tables

Regulation Method	Best Val Loss	Best Val Acc	Test Loss	Test Acc
None	0.0128	73.58%	0.0133	72.3%
Dropout	0.0124	74.8%	0.0130	73.6%
DropConnect	0.0111	76.9%	0.0117	75.4%
DropBlock	0.0102	78.24%	0.0102	77.7%

Table 1: Complete result for 4.1

Batch Size	Unregularized	Dropout	DropConnect	DropBlock
32	73.84%	74.36%	76.64%	79.62%
64	73.58%	74.8%	76.9%	78.24%
128	73.54%	74.26%	75.28%	76.7%
256	72.16%	73.76%	75.24%	76.2%
512	71.92%	73.52%	74.74%	77.08%

Table 2: Best validation accuracy of different models with various batch size. $learning\_rate = 0.0003, drop\_rate = 0.5, epochs = 45$.

Learning rate	Unregularized	Dropout	DropConnect	Dropblock
0.0001	0.0132	0.0125	0.0117	0.0110
0.0003	0.0128	0.0124	0.0111	0.0102
0.0005	0.0102	0.0119	0.0114	0.0100
0.001	0.0121	0.0119	0.0114	0.0102
0.002	0.0132	0.0128	0.0123	0.0118

Table 3: Lowest validation loss with different learning rate

A.3 Figures

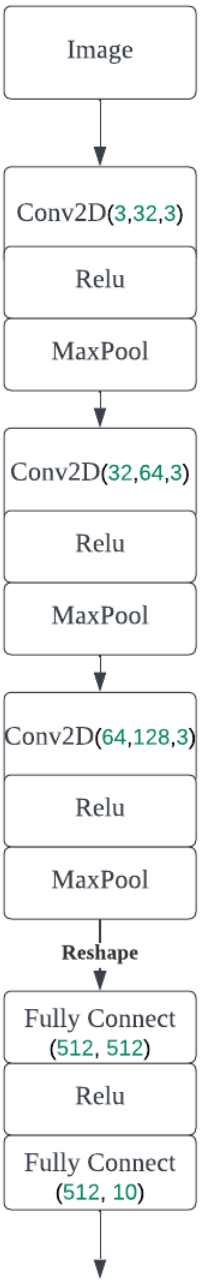

Figure 4: The neural network used in our experiment

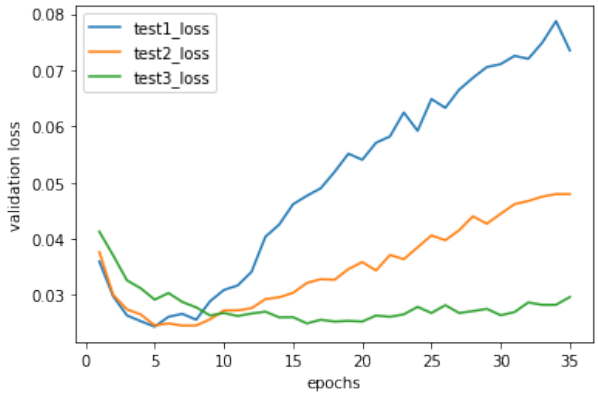

Figure 5: All three curves here have similar lowest validation loss, while the extents of their overfitting do differ significantly.

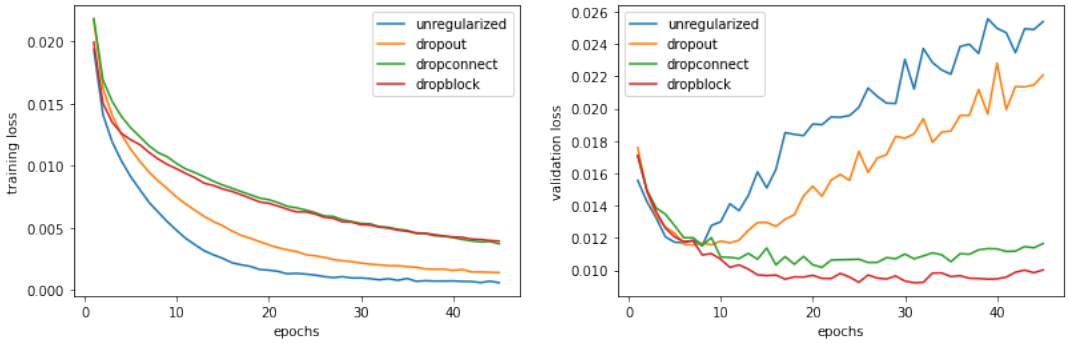

Figure 6: training and validation loss curves with Batch Normalized network. $batch_norm = 64, learning\_rate = 0.0003, epochs = 60, drop\_rate = 0.5$

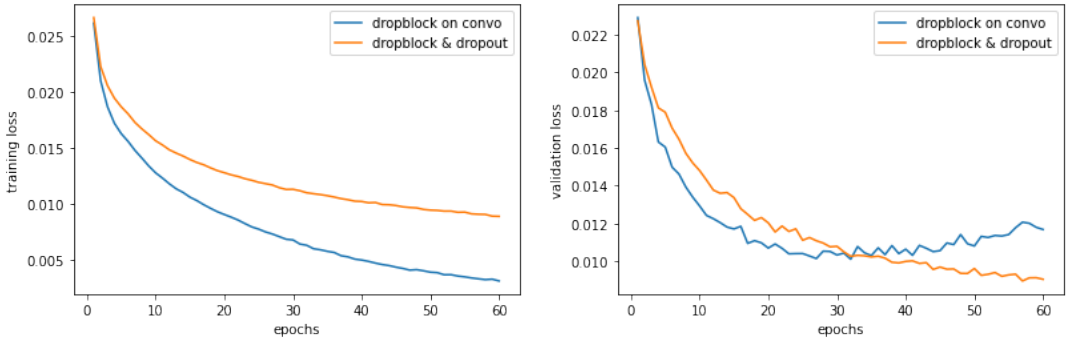

Figure 7: It is shown that combining DropBlock and Dropout can reduce the overfitting and improve the performance. $batch_norm = 64, learning\_rate = 0.0003, epochs = 60, drop\_rate = 0.5$.