Combining Batch Normalization and Dropout in the Training of Deep Neural Network

Aveek Choudhury, Harshita Ved, Aishwarya Vantipuli

April 21, 2021

Abstract

Deep Neural networks (DNN) are revolutionizing the industry and daily life today, taking artificial intelligence to the next level. To achieve state-of-the-art efficiency, DNNs in AI require millions of data entries and complicated model training. Finding ways to improve the quality of DNN training has hence become a major challenge. Our work in this project deals with implementation of a novel independent-Component (IC) layer before each weight layer in order to make the inputs independent as suggested by Chen et al., 2019. The research is focused on the brilliant idea that whitening inputs will speed up convergence of the network. Through this work, we reproduce a part of the original work by implementing the IC layer for residual networks and validate that the proposed whitening method indeed improves the training process and performance of our residual nets on the CIFAR-10 dataset.

1 Related Work

The original work that we have tried to replicate was done by a team of researchers from Tencent Technology, the Chinese University of Hong Kong, and Nankai University who proposed a novel method to address the issue of improving training deep neural networks (DNN). Their work is published through the paper "Rethinking the Usage of Batch Normalization and Dropout in the Training of Deep Neural Networks" (Chen et al., 2019) which proposes a marriage of 2 commonly used techniques – Batch Normalisation and Dropout, to achieve input whitening that neither method could otherwise achieve independently.

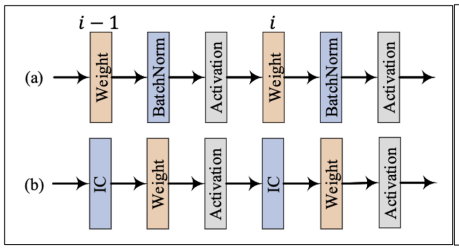
1.1 Scientific Context

The benefits of whitening the inputs of neural networks have already been established and advocated in prior work (Le Cun et al., 1991). More recent studies (Moreno-Bote et al., 2014; Beaulieu-Laroche et al., 2018) have established the finding of positive linear dependence between representation power of a neural system and the number of independent neurons. Generation of independent

components and whitening of inputs have previous been explored using methods like Independent Component Analysis (ICA) (Oja & Nordhausen, 2006) and Zero-phase Component Analysis (ZCA) (Huang et al., 2018). However, these methods involve calculations that are computationally expensive and thus, become increasingly difficult to use with deep neural networks.

In their pursuit of computationally efficient method for whitening inputs of every layer of the neural network, the authors of the paper discover that independent activations for neurons in intermediate layers could be constructed using a combination of BatchNorm and Dropout. Intuitively, BatchNorm performs a task like ZCA – normalizes net activations to zero mean and unit variance. On the other hand, with the usage of Dropout, neurons are allowed to either output their value with a probability p or zero, made feasible by introducing random gates for neurons in a layer. The two methods are combined in an Independent-Component (IC) layer, the main contribution of the authors, and applied to each weighted layer in the network. Due to these effects of BatchNorm and Dropout, the application of the IC layer disentangles each pair of neurons in a layer continuously.

As represented in Fig. 1(b), the IC layer is attached to each weight layer in the ResNet. In their study the authors explore the improvement in training process contributed by the IC layers in three different ordering patterns – a) ReLU-IC-Conv2D, b) IC-Conv2D-ReLU, and c) Conv2D-ReLU-IC each of which has a short connection as in Fig. 2(b).



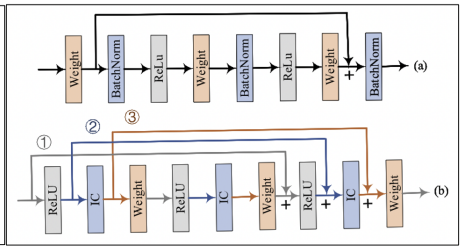


Figure 1 (a) The common practice of performing the whitening operation, or named as batch normalization, between the weight layer and the activation layer. (b) Our proposal to place the IC layer right before the weight layer.

Figure 2 (a) The classical ResNet architecture, where '+' denotes summation. (b) Three proposed ResNet architectures reformulated with the IC layer.

1.2 Original Results

Our work focuses on the replication of results obtained by the authors in training four different networks with and without the IC layers on the CIFAR-10 dataset. The four network variants chosen to evaluate the impact of IC layer on the training process are – ResNet-110, ResNet-164, ResNet-110 (Bottleneck) and ResNet-164 (Bottleneck). The authors presented consistently better training performance using the IC layers on CIFAR-10, which is more apparent in the residual block variants. Of the various IC layer orderings, the ordering of Conv2D-ReLU-IC performs the best across network variants.

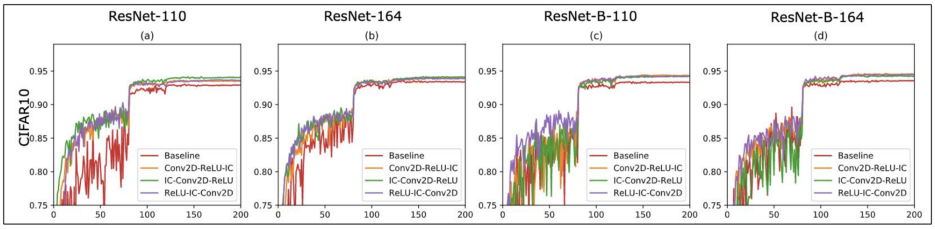


Figure 3 The testing accuracy of implementing ResNet and ResNet-B with the IC layer on the CIFAR10 dataset with respect to the training epochs. (a) ResNet110 on CIFAR 10. (b) ResNet164 on CIFAR 10. (c) ResNet-B 110 on CIFAR 10. (d) ResNet-B 164 on CIFAR10.

2 Implementation Details

The network architecture variants described in the original paper have been implemented in PyTorch. The ResNet-110 and ResNet-164 architectures are implemented with the first layer of 3 x 3 convolutions followed by $6n$ stacks of two-weighted 3 x 3 convolution layers in each residual block for feature maps of 32 x 32, 16 x 16, 8 x 8 with $n = 18$ and $n = 27$ respectively. A stride of 2 and 16, 32, 64 number of feature maps are used in the ResNets. The bottleneck versions of the two networks use three-weight layers in each block $\begin{bmatrix} 1x1, & 16 \\ 3x3, & 16 \\ 1x1, & 64 \end{bmatrix}$ with $n = 12$ and $n = 18$ respectively. These residual and bottleneck blocks are followed by average pooling and a 10-way fully connected layer. To ensure fair comparison of training performance improvement, the weighted layers in the ResNets without IC layers are followed by BatchNorm and ReLU activation as is the standard.

The IC layer is implemented as a sequential model with 2 layers for BatchNorm and Dropout. The IC layer was attached to each weighted layer in the Residual and Bottleneck blocks in the 3 ordering variations as specified before.

2.1 Experimental Setup

Our training process resembles the one used by the authors in their original work barring a few changes. Dropout probability was set to $p = 0.05$ across all experiments. Owing to computational limitations, we made small modifications to the training process – a) training for 100 epochs instead of 200, and b) using a fixed learning rate of 0.001 instead of the learning rate update rule specified. The model was trained on the CIFAR-10 training set of 50k images spread across 10 classes using a mini-batch of 64 images and evaluated against the accuracy measured on the test set comprising of 10k images.

2.2 Results

Though our networks fail to perform as well as the ones in the paper, possibly due to fact that we did not apply the learning rate update rule and trained for only 100 epochs instead of 200, the use of IC layers in the network do provide an improvement over the baseline networks as seen from the accuracies reported over the test set as the training progresses. Our results depict a similar trend as

the authors in terms of a faster convergence speed and better convergence limit. This indicates a more stable training performance with the usage of IC layers as compared to the baseline ResNet architectures. The performance improvement is most clearly visible in the training of the ResNet-110 architecture where the IC variants outperform the baseline version by a huge margin.

Table 1 The testing results of implementing ResNet and ResNet-B with the IC layer on the CIFAR10 dataset at the end of training.

Model	Depth	Layers in Residual Unit	CIFAR10
ResNet	110	2×{ReLU-IC-Conv2D}	0.8226
		2×{IC-Conv2D-ReLU}	0.8429
		2×{Conv2D-ReLU-IC}	0.8449
		Baseline	0.6454
	164	2×{ReLU-IC-Conv2D}	0.8076
		2×{IC-Conv2D-ReLU}	0.8279
		2×{Conv2D-ReLU-IC}	0.8402
		Baseline	0.8183
ResNet-B	110	3×{ReLU-IC-Conv2D}	0.8251
		3×{IC-Conv2D-ReLU}	0.8355
		3×{Conv2D-ReLU-IC}	0.8389
		Baseline	0.8159
	164	3×{ReLU-IC-Conv2D}	0.8218
		3×{IC-Conv2D-ReLU}	0.8455
		3×{Conv2D-ReLU-IC}	0.8498
		Baseline	0.8254

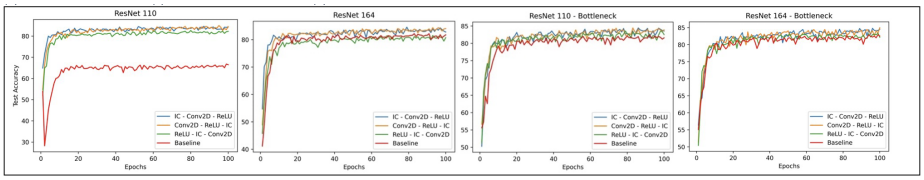


Figure 4 The testing accuracy of implementing ResNet and ResNet-B with the IC layer on the CIFAR10 dataset with respect to the training epochs. (a) ResNet110 on CIFAR 10. (b) ResNet164 on CIFAR 10. (c) ResNet-B 110 on CIFAR 10. (d) ResNet-B 164 on CIFAR10.

3 Conclusion

Through the experiments conducted during this project, we were able to validate that the implementation of the IC layer in the ResNet architecture does in fact lead to a better training process. The inclusion of the IC layer before the weight layers to transfer activations into independent components results in faster convergence speed and better generalization performance on the CIFAR-10 dataset as per our experiments. This method shows promise in the training of neural networks, especially the deeper versions where incorporating other computationally expensive whitening methods are challenging. Future work, as described by the authors, would involve incorporating more advanced normalization methods into the IC layer, as well as constructing independent components using advanced statistical techniques.

References

[1] G. Chen, P. Chen, Y. Shi, C.-Y. Hsieh, B. Liao and S. Zhang, "Rethinking the usage of batch normalization and dropout in the training of deep neural networks", arXiv:1905.05928, 2019, [online].
[2] Le Cun, Y., Kanter, I., and Solla, S. A. Eigenvalues of covariance matrices: Application to neural-network learning. Physical Review Letters, 66(18):2396, 1991.
[3] Ioffe, S. and Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. arXiv preprint arXiv:1502.03167, 2015.
[4] Huang, L., Yang, D., Lang, B., and Deng, J. Decorrelated batch normalization. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 791–800, 2018.
[5] Oja, H. and Nordhausen, K. Independent component analysis. Encyclopedia of Environmetrics, 3, 2006.