Boolean Structured Convolutional Deep Learning Network (BSconvnet)

Sing Kuang Tan
Email: singkuangtan@gmail.com

May 28, 2023

Abstract

In this paper, I am going to propose a new Boolean Structured Convolutional Deep Learning Network (BSconvnet) built on top of BSnet, based on the concept of monotone multi-layer Boolean algebra. I have shown that this network has achieved significant improvement in accuracy over an ordinary Relu Convolutional Deep Learning Network with much lesser number of parameters on the CIFAR10 dataset.

1 Introduction

My previous model BSnet trained on MNIST dataset based on fully connected deep learning network seems like a toy problem. Because MNIST dataset is so simple that a Support Vector Machine (SVM) can solve it. So I experimented with current model BSconvnet on CIFAR10 dataset, that is a small version of Imagenet dataset, which can prove that my BSconvnet can solve real world problem.

2 Hypotheses

Hypothesis 1: My BSnet will overfit CIFAR10 dataset due to its fully connected deep learning network design.

I did experiments with my fully conected BSnet on CIFAR10. The accuracy is much lower than an ordinary convolutional network. My conclusion is that the high dimensions of fully connected network with large number of parameters will easily overfit the CIFAR10 dataset. CIFAR10 is unlike MNIST dataset, where specific location pattern of an image is important for classification, using convolutional network will ignore the exact spatial locations but concentrate on the patch patterns of image, will classify the dataset with higher accuracy.

Hypothesis 2: My BSconvnet has better performance than an ordinary convolutional Deep Learning Network.

Look at the leaderboard table of accuracies on CIFAR10 using different models (https://paperswithcode.com/sota/image-classification-on-cifar-10), my model achieves the same accuracy using much lesser parameters. So my model is more efficient per parameter.

Hypothesis 3: My BSconvnet will not work without batch normalization.

I removed the batch normalization from BSconvnet. The trained model accuracy is very low. I think batch normalization is needed to prevent the gradients and weights to become skewed (in Singular Value Decomposition explanation, skewed means that the largest eigenvalue and the lowest eigenvalue differs by a lot), and this makes training possible.

Hypothesis 4: My BSconvnet will not work if the negation branch is taken out of the network.

Note that when I took negation branch out of the BSconvnet, the non-negative weights constraint still exist. I did an experiment to remove the negation branch. The trained accuracy of the model is very low. It shows that the network is unable to learn without the negation branch. I think without the non-negative constraint, there are natural negations in the input values of each layer, and therefore a normal convolutional network is still able to learn a dataset.

3 My Model

I developed a Boolean Structured Convolutional network (BSconvnet). Every complex convolutional layer of the input will first concatenate the positive and negative vector of previous layer input. Then it will pass through a convolutional layer where all weights are constrained to be always positive or zero, and apply a Relu activation function to the output of convolutional layer. The concatenation of positive and negative vector represents the "Not" gates, and the convolutional layer represents the "And" and "Or" gates. Look at [2] https://vixra.org/abs/2112.0151.

It will go through 1 layer of (height=5,width=1) complex convolutional layer, followed by 1 layer of (height=1,width=5) complex convolutional layer. Then it will go through 1 layer of (height=1,width=5) complex convolutional layer, followed by 1 layer of (height=5,width=1) complex convolutional layer. This 4 types of complex convolutional layers will repeat another time, so in total there are 8 complex convolutional layers. Next a complex max pooling of (height=4, width=4)is applied to the previous output. Note that the convolutions are using padding=same, which means that the width and height of output are the same as the width and height of input.

Next a (1,8) complex convolutional layer followed by a shift in previous output by 0.5, followed by a (8,1) complex convolutional layer then another shift in previous output by 0.5. The previous two complex convolutional layers are applied to reduce the dimensions of previous layer output. The shift by 0.5 is to make the output of previous layer into a new output with range more suitable for the next layer to process. This two complex convolutional layers make the input to the complex fully connected layer smaller in dimensions, reducing the total parameter count by a lot. Note that these convolutions are using padding=valid, which means that the width and height of output from the convolution will be smaller than the width and height of the input.

Then a series of 2 complex fully connected layer is applied and lastly a sigmoid fully connected layer of 10 outputs to represent the output probability prediction of each class in CIFAR10 dataset.

Input is $32 \times 32 \times 3 = 3072$ dimensions and output is 10 classes.

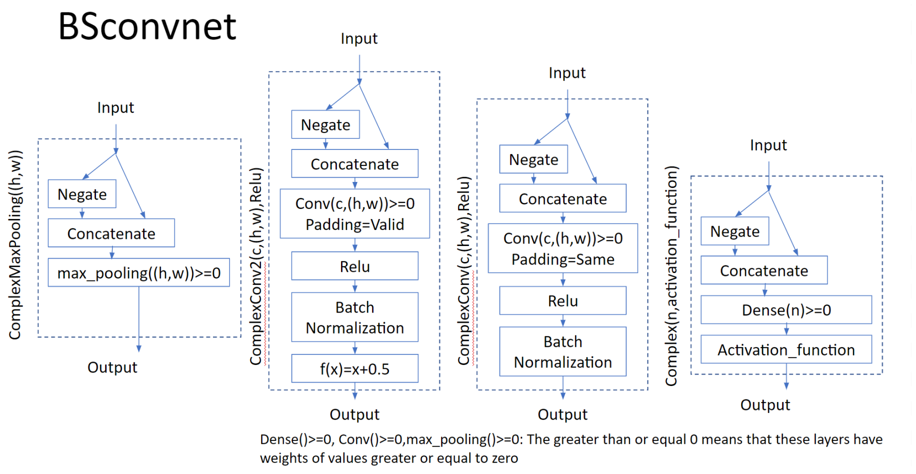

BSconvnet
Input
ComplexMaxPooling((h,w))
Negate
Concatenate
max_pooling((h,w))>=0
Output
Input
ComplexConv2(c,(h,w),Relu)
Negate
Concatenate
Conv(c,(h,w))>=0 Padding=Valid
Relu
Batch Normalization
f(x)=x+0.5
Output
Input
ComplexConv(c,(h,w),Relu)
Negate
Concatenate
Conv(c,(h,w))>=0 Padding=Same
Relu
Batch Normalization
Output
Input
Complex(n,activation_function)
Negate
Concatenate
Dense(n)>=0
Activation_function
Output
Dense()>=0, Conv()>=0,max_pooling()>=0: The greater than or equal 0 means that these layers have weights of values greater or equal to zero

Figure 1: Network Diagram of BSconvnet (Basic Blocks)

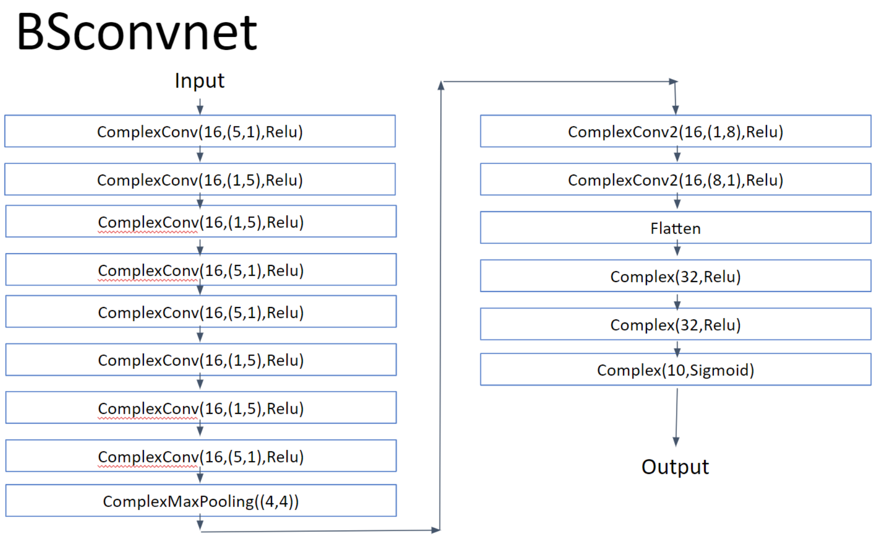

BSconvnet
Input
ComplexConv(16,(5,1),Relu)
ComplexConv(16,(1,5),Relu)
ComplexConv(16,(1,5),Relu)
ComplexConv(16,(5,1),Relu)
ComplexConv(16,(5,1),Relu)
ComplexConv(16,(1,5),Relu)
ComplexConv(16,(1,5),Relu)
ComplexConv(16,(5,1),Relu)
ComplexMaxPooling((4,4))
ComplexConv2(16,(1,8),Relu)
ComplexConv2(16,(8,1),Relu)
Flatten
Complex(32,Relu)
Complex(32,Relu)
Complex(10,Sigmoid)
Output

Figure 2: Network Diagram of BSconvnet (Overall Model)

The figures of the BSconvnet are shown in Figure 1 and Figure 2.

Each complex convolutional layer logic is first the concatenation of the input and negated input. Then it passes through a 2D convolution layer with weights constrainted to be always positive or zero. Next a batch normalization is applied so that the output will not be skewed for the next layer. As for the complex fully connected layers, they are the same as BSnet [3], with concatenation of input and negated input followed by a fully connected layer with weights constrainted to be always positive or zero. Note that complex fully connected layers do not need batch normalization layers as the output of these fully connected layers are not very skewed.

I also added a very minor modification to the weights of each neuron. The weights of each neuron is set to zero for weights smaller than zero, then the non-zero weights are set to a mean value of those weights greater than zero. This modification is very minor and is not necessary for good performance. This is to make the gradient descent easier with equal value weights.

It may seems redundant to add a negation input in my complex layers. For inference, the negation input can be combined with non-negated input to form an ordinary deep learning layer. However for training, the negated branch actually helps the deep learning weights to switch between negated input and positive input easily, without it the deep learning weights got stuck in the local minima.

4 Experiment Results

The Figures 3 and 4 below shows the validation and training curves of the BSconvnet networks.

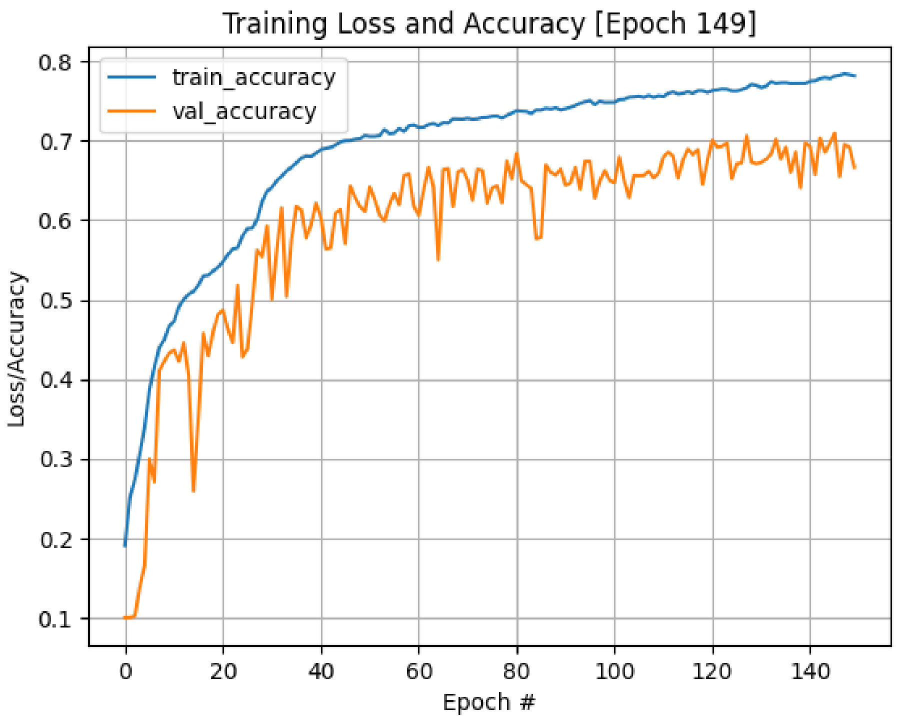


Figure 3: Validation and Training Accuracies of BSconvnet of first pass training at learning rate=1

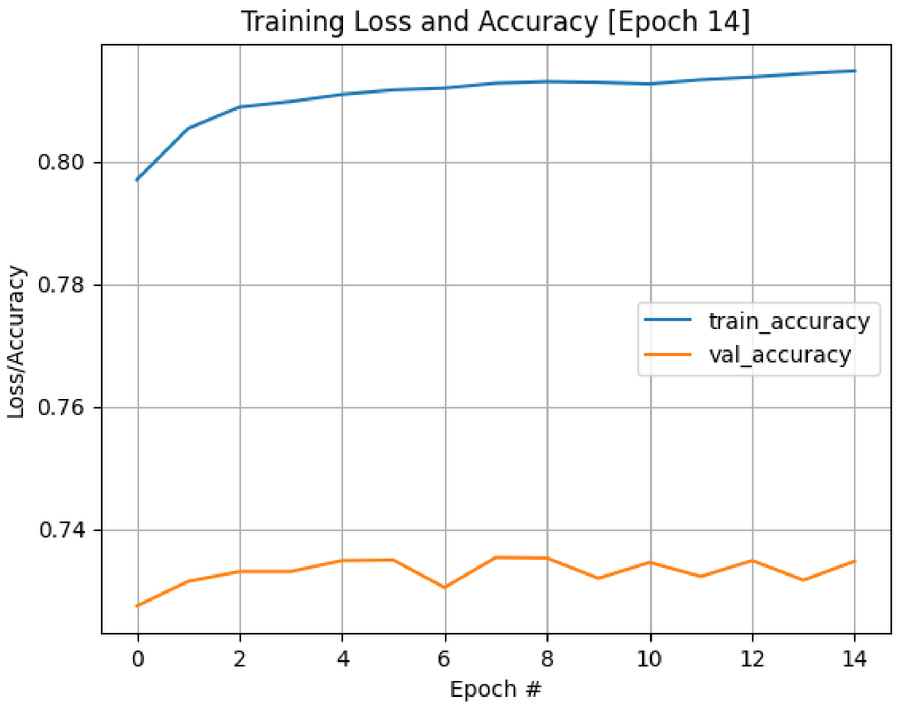
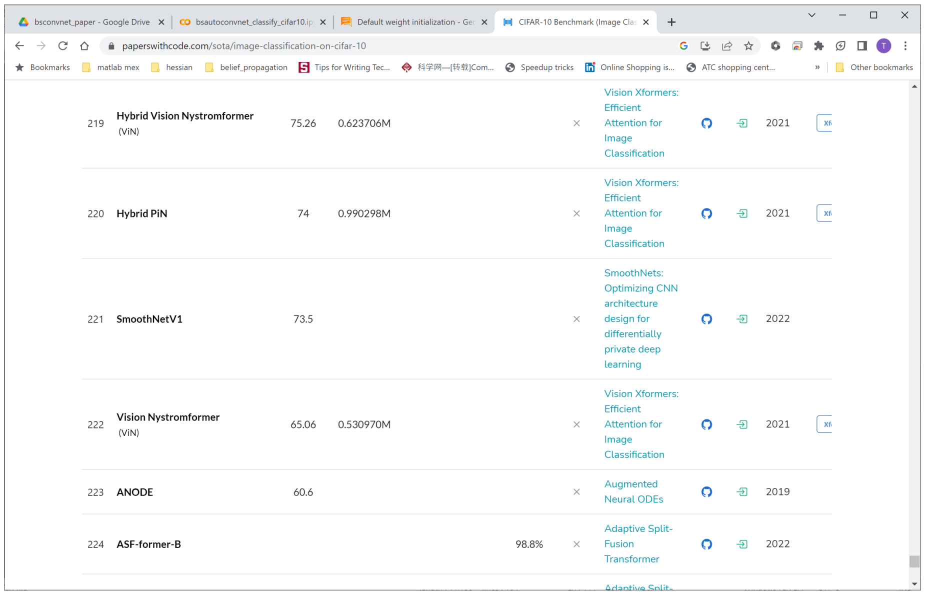


Figure 4: Validation and Training Accuracies of BSconvnet of second pass training at learning rate=0.1

My BSconvnet will have higher accuracy (73.5%) than ordinary network (see accuracies Figures 3 and 4 for CIFAR10). Note that the training is carried out in two passes, with first pass at learning rate=1, second pass at learning rate=0.1.



bsconvnet_paper - Google Drive | bsautoconvnet_classify_cifar10.ip | Default weight initialization - Ger | CIFAR-10 Benchmark (Image Clas

paperswithcode.com/sota/image-classification-on-cifar-10

Bookmarks | matlab mex | hessian | belief_propagation | Tips for Writing Tec... | 科学网—[转载]Com... | Speedup tricks | Online Shopping is... | ATC shopping cent... | » | Other bookmarks

219	**Hybrid Vision Nystromformer** (ViN)	75.26	0.623706M		×	Vision Xformers: Efficient Attention for Image Classification	2021	Xf
220	**Hybrid PiN**	74	0.990298M		×	Vision Xformers: Efficient Attention for Image Classification	2021	Xf
221	**SmoothNetV1**	73.5			×	SmoothNets: Optimizing CNN architecture design for differentially private deep learning	2022	
222	**Vision Nystromformer** (ViN)	65.06	0.530970M		×	Vision Xformers: Efficient Attention for Image Classification	2021	Xf
223	**ANODE**	60.6			×	Augmented Neural ODEs	2019	
224	**ASF-former-B**			98.8%	×	Adaptive Split-Fusion Transformer	2022	

Figure 5: Table of accuracies of CIFAR10 models ever trained

From the table of accuracies of CIFAR10 models from Figure 5 (https://paperswithcode.com/sota/image-classification-on-cifar-10), our model is better. My BSnet has 17477 parameters. From the table of CIFAR10 accuracies, a SmoothNetV1 [1] model of accuracy 73.5% has 3.4 millions parameters, with is quite a few times larger than mine. This shows that my model is much powerful with lesser parameters.

You may think that making small changes (concatenate positive and negative vector of previous layer output, with weights constraint of greater or equal to zero) from the ordinary network to BSconvnet result in small improvement. But for neurons with high dimensions input, the improvement is very obvious.

Note that I use separable convolutional layers, which means instead of using a (5,5) convolution, a (5,1) and another (1,5) convolutions are used instead to prevent it from overfitting the CIFAR10 dataset, verified from experiments. This design also reduce the total parameter count by a lot.

My BSconvnet is to mimic a Discrete Markov Random Field Relaxation model [2], but it is not exactly the same. The relaxation model is a convex optimization model, so it is convex. My BSconvnet is optimization function is smooth (much more convex than ordinary deep learning network optimization function), with the help of batch normalization layers and special convolutional and fully connected layers.

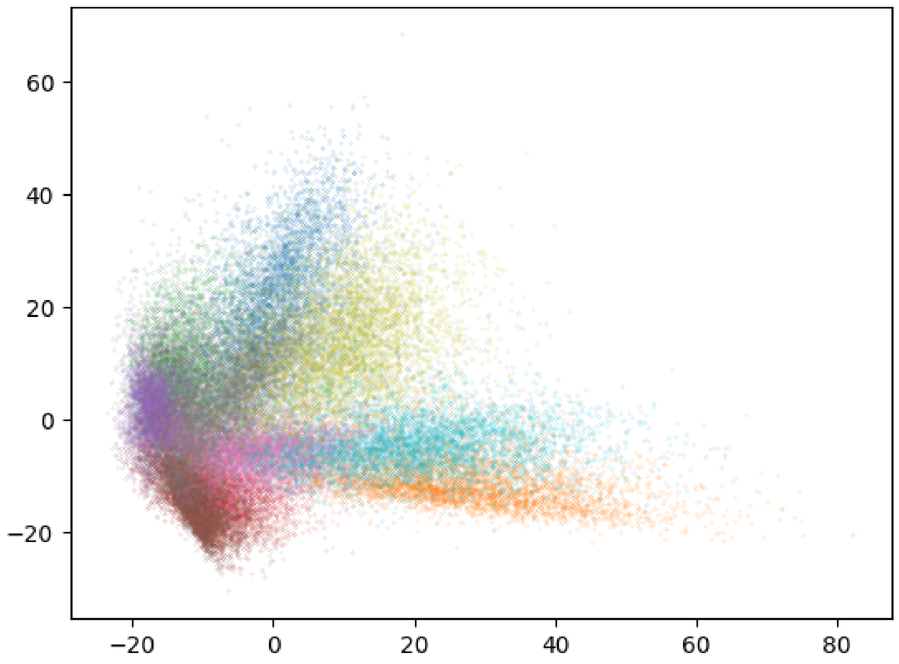


Figure 6: Training set embeddings (second last layer) of BSconvnet

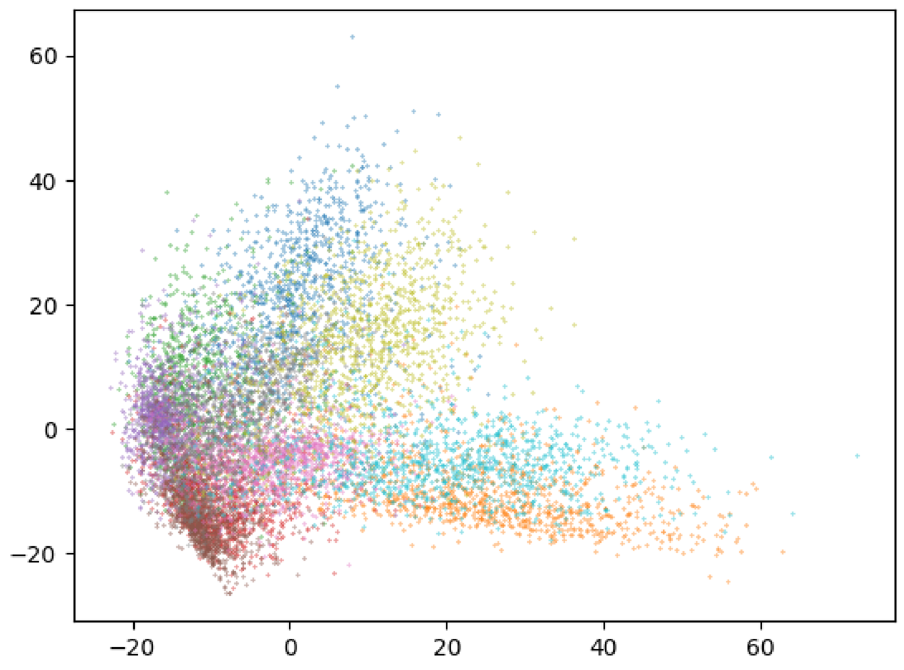


Figure 7: Test set embeddings (second last layer) of BSconvnet

For the scatter plot (see Figure 6 and Figure 7 of the second last layer (acts like embedding layer), we can see that the datapoints are coherent, very suitable to act as feature vectors for another classification task. For each scatter plot, there are 10 colors, where each color represents a class in CIFAR10 dataset.

Stochastic gradient descent is used instead of ADAM gradient descent (with adaptive step size) because ADAM does not work well with batch normalization layers.

It seems like for BSautonet autoencoder [4], batch normalization layers are not necessary as the output of the complex layers are not that skewed.

Access my GitHub codes thru this link:
https://github.com/singkuangtan/BSconvnet

5 Conclusion

I have developed a Boolean Structured Convolutional Deep Learning Network for general classification problem (CIFAR10 dataset) in machine learning. The design in this network can be easily transferred to other type of tasks (such as Imagenet dataset) for general classification problem.

The idea is to make training algorithm or gradient descent of the deep learning network convex, so that training is easier and faster without many hyper-parameters to tune. To get rid of dropout, weight decay, learning rate, batch normalization, other complicated and unnecessary designs so that the training process and the overall network structure become simple, easy to use and easy to design new networks for new problems.

The Boolean structure in the network is able to provide a theoretical model on how deep learning works, how it learns and how it can be used to model any general high dimensions function with the help of my Discrete Markov Random Field Relaxation model.

References

[1] Nicolas W. Remerscheid, Alexander Ziller, Daniel Rueckert, and Georgios Kaissis. Smoothnets: Optimizing CNN architecture design for differentially private deep learning. *CoRR*, abs/2205.04095, 2022.

[2] Sing Kuang Tan. Discrete markov random field relaxation. *https://vixra.org/abs/2112.0151*, 2021.

[3] Sing Kuang Tan. Boolean structured deep learning network (bsnet). *https://vixra.org/abs/2212.0193*, 2022.

[4] Sing Kuang Tan. Design autoencoder using bsnet (bsautonet). *https://vixra.org/abs/2212.0208*, 2022.