Boolean Structured Autoencoder Convolutional Deep Learning Network (BSautoconvnet)

Sing Kuang Tan
Email: singkuangtan@gmail.com

June 16, 2023

Abstract

In this paper, I am going to propose a new Boolean Structured Autoencoder Convolutional Deep Learning Network (BSautoconvnet) built on top of BSconvnet, 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 Autoencoder Convolutional Deep Learning Network with much lesser number of parameters on the CIFAR10 dataset. The model is evaluated by visual inspection of the quality of the reconstructed images against groundtruth with reconstructed images by models in the internet.

1 Introduction

My previous model BSautonet is trained on MNIST dataset is based on fully connected deep learning network. Because fully connected layer design is not suitable for CIFAR10 dataset (which our current model is trained on), we designed our current autoencoder model using only convolutional layers. I experimented with current model BSautoconvnet on CIFAR10 dataset, that is a small version of Imagenet dataset, which can prove that my BSautoconvnet can solve real world problem.

I added noise to the input by removing 70 percent of the input pixel color values and set them to white color. This will challenge the reconstruction ability of our BSautoconvnet and demostrate our BSautoconvnet ability to reconstruct input image with sparse number of input pixel values.

My BSautoconvnet can not only use for image reconstruction, it can also be used for image segmentation or image translation.

My BSautoconvnet is designed like a Boolean algebra, because Boolean algebra can model any mathematical function. Just like the Arithmetic and Logic Unit (ALU) in our microprocessor, the Boolean operations in the processor can compute any function. Boolean operations are the smallest units in a function, so by using Boolean algebra formulation for my deep learning network, it has simplified the function representation in the smallest form.

2 Hypotheses

Hypothesis 1: My BSautoconvnet with (height=3,width=3) convolutional layers is unable to reconstruct the input image. It needs at least (5,5) convolutional layers.

I have experimented using (height=3,width=3), the trained model is unable to reconstruct the input image. My conclusion is that the removing of 70 percent of input pixels leads to the spacing between a pair of informative pixels is about 5 pixels away along the width or height, and therefore (5,5) convolutional layers are the most suitable.

Hypothesis 2: My BSautoconvnet needs skip connections between layers. But BSnet [2] and BSconvnet [4] may not need skip connections. Skip connections will improve the training time, but will not affect the final trained model accuracy.

In my experiments, BSautoconvnet with skip connections will help to propagate input pixel values fast to the output. Otherwise, without skip connections, input information get lost and it makes the autoencoder difficult to reconstruct the output with high accuracy. It is the same for BSautonet, which needs skip connections.

As for BSnet and BSconvnet, they are classifiers. With my monotone Boolean algebra complex layers, they can learn without skip connections. Skip connection does not help to improve classifier accuracies, but helps to speed up the training time by a small factor, which is not really needed.

Hypothesis 3: My BSautoconvnet with 5 bit binarized input will generate better reconstructed output than real number input.

Our BSautoconvnet can work with 5 bits or real input. However, 5 bits input allows our model to detect certain input pixel color values better, such as black or white pixels, and therefore able to reconstruct the output with more precision, but having about the same mean squared error of using real input.

My BSautoconvnet network is a semi-binary newtwork. The inputs and output of each neuron are designed to accept semi-Boolean continuous values between 0 and 1. In the future, I will design netework that is fully binary, meaning that it only accept discrete 0 and 1 values.

I use 5 bits instead of 8 bits in BSautonet [3], so that the input image dimension size is more manageable.

Hypothesis 3: My BSautoconvnet does not need batch normalization layers.

With and without batch normalization layer, our BSautoconvnet can still learn to reconstruct input images. I think it is because the multiple output heads of our model makes the weights and gradients less skewed, where 1 output heads of BSconvnet makes the weights and gradients very skewed, thus eliminates the need for batch normalization for our BSautoconvnet.

Our BSautoconvnet can work without batch normalization. Putting batch normalization has no harm, and it still can do training as normal.

3 My Model

Modern deep learning networks are designed incrementally. New designs are added to old designs to solve old designs problems which makes the deep learning network unnecessary complex. It has unnecessary complex designs such as dropout, weight decay, learning rate and batch normalization, which many of them can be eliminated and consolidated into one unified design. Each unnecessary design is a stop gap measure which is put into the network in an adhoc manner. We should design network with designs that are put in together in a cohesive manner through a global design optimization function.

I developed a Boolean Structured Autoencoder Convolutional network (BSautoconvnet). Every complex convolutional layer of the input will first concatenate the positive and negative vector of previous layer output. 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 [1] https://vixra.org/abs/2112.0151.

It will go through a series of five layers of (height=5,width=5) complex convolutional layer. There will be skip connection from the first convolutional layer, skip the second convolutional layer and connect to the third convolutional layer. Another one will skip the second and third convolutional layer to the fourth convolutional layer. The skip connections make the network looks like an ensemble of few convolutional networks, will different number of layers. The skipping from the first to the last convolutional layer makes the model easier to reconstruct the output that is similar to the input because the input information is quickly sent to the output. Our design is simple, where the output of each convolutional layer are padded (with padding=same) and has the same width-height dimensions as the input so that information can be passes between certain pair of layers by just directly connect them through skip connection. This will make your life much easier. Note that each concatenation from skip connection to another layer, or negated branch with positive branch is done by concatenating along the channel dimension of the output from previous layer. Note that there is no need for batch normalization for our autoencoder model. There is no embedding layer like our BSautonet. You can create one by redesigning the skip connections if you want to.

There is no need for fully connected layers. Each convolutional layer has the same number of channels (channels=16) so that it is easy to understand. We do not have to waste time to fine tune the number of channels in each layer to get very small improvement, still get good result, show how our monontone neuron design helps in the final performance.

Input is $32 \times 32 \times 3 = 3072$ dimensions and output has the same dimension as the input, which is also $32 \times 32 \times 3 = 3072$. After converting each real number in the input into 5 bits, the input size is $32 \times 32 \times 3 \times 5 = 15360$.

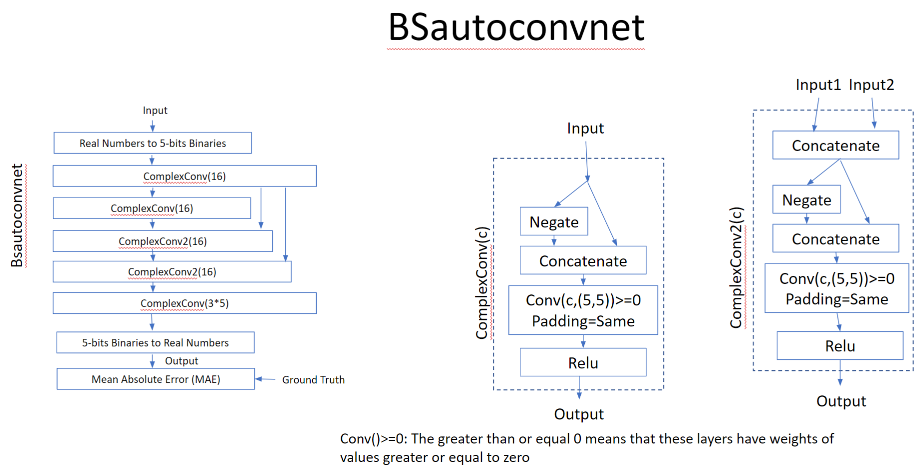


Figure 1: Network Diagram of BSautoconvnet

The model of the BSconvnet are shown in Figure 1.

Our BSautoconvnet is trained using ADAM gradient descent algorithm with learning rate=1.

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 2, 3 and 4 below shows the input, ground truth and output (prediction or reconstructed image) of the our BSautoconvnet network.

Figure 2: Input image

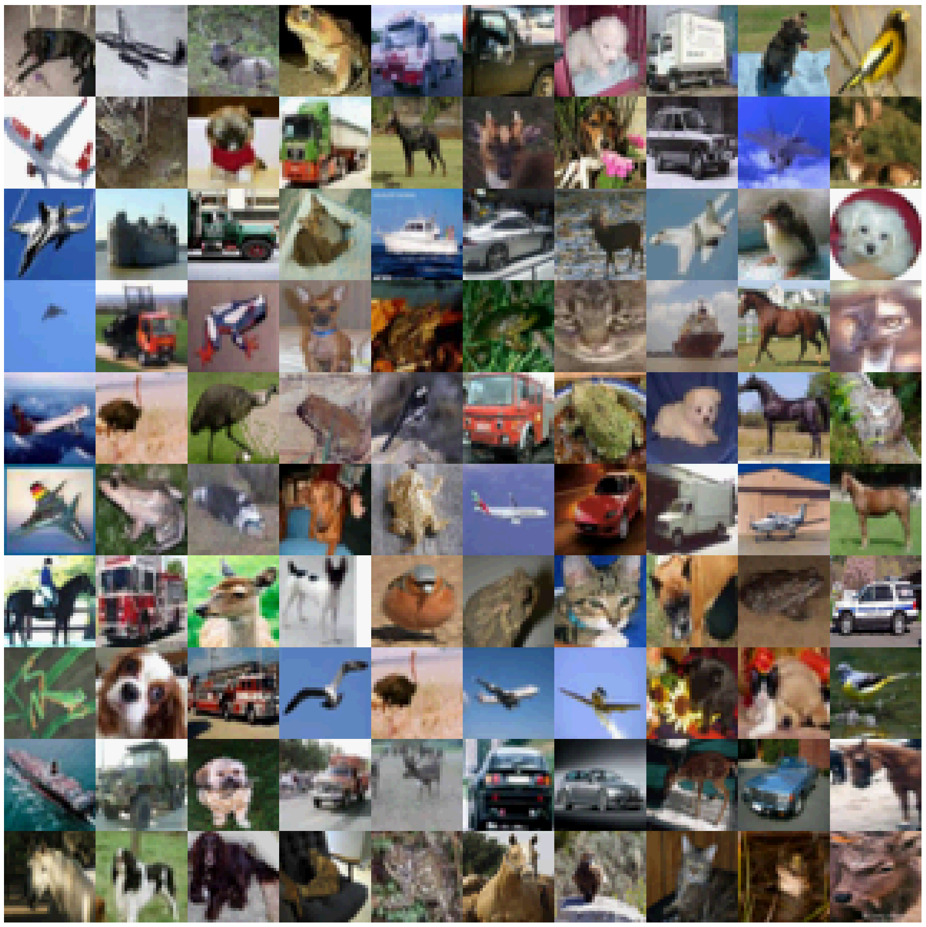


Figure 3: Groundtruth

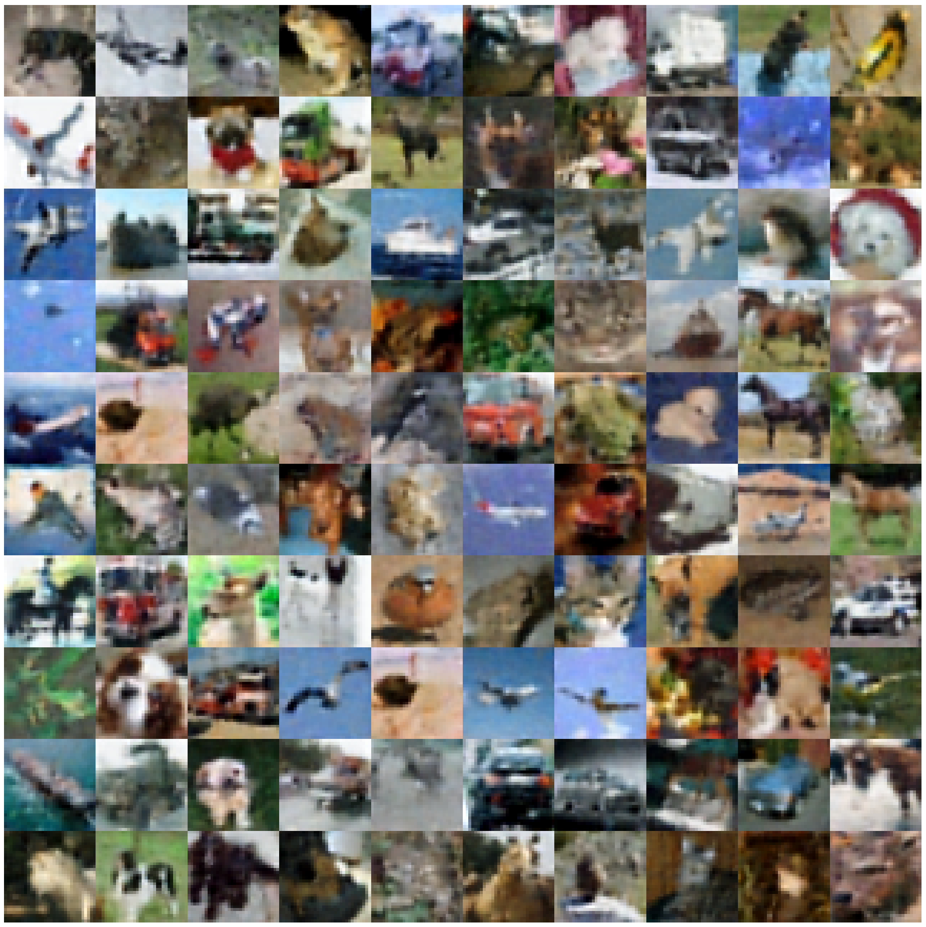
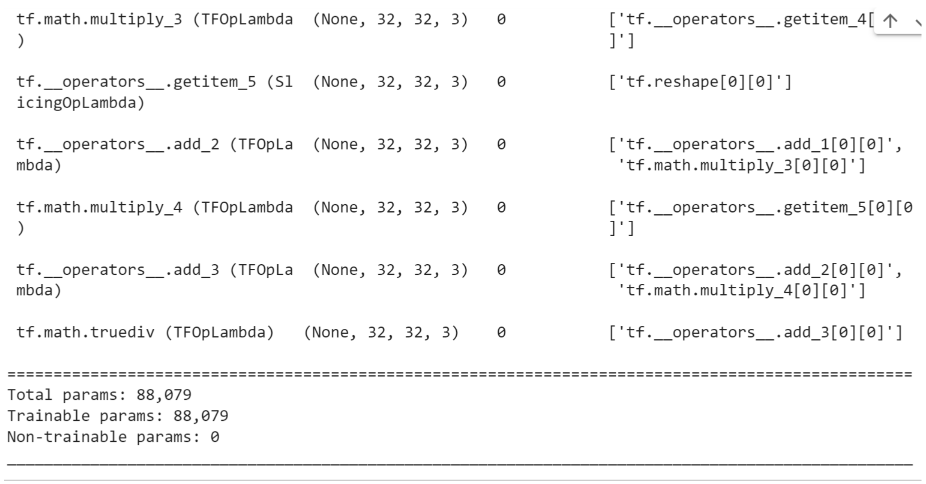


Figure 4: Output image

My BSconvnet will have mean absolute error (MAE) of 0.04107 (on CIFAR10 dataset), higher than an ordinary network. See the input, output (predicted, reconstructed image) and ground truth for the visual inspection of the reconstruction quality. The training is carried out in one pass, one learning rate, without any data augumentations and regularization such as weight decay and dropout. This simplicity makes training of the model and testing hypothesis on the model easy.



```
 tf.math.multiply_3 (TFOpLambda  (None, 32, 32, 3)   0          ['tf.__operators__.getitem_4[
 )                                                               ]']

 tf.__operators__.getitem_5 (Sl  (None, 32, 32, 3)   0          ['tf.reshape[0][0]']
 icingOpLambda)

 tf.__operators__.add_2 (TFOpLa  (None, 32, 32, 3)   0          ['tf.__operators__.add_1[0][0]',
 mbda)                                                            'tf.math.multiply_3[0][0]']

 tf.math.multiply_4 (TFOpLambda  (None, 32, 32, 3)   0          ['tf.__operators__.getitem_5[0][0
 )                                                               ]']

 tf.__operators__.add_3 (TFOpLa  (None, 32, 32, 3)   0          ['tf.__operators__.add_2[0][0]',
 mbda)                                                            'tf.math.multiply_4[0][0]']

 tf.math.truediv (TFOpLambda)   (None, 32, 32, 3)    0          ['tf.__operators__.add_3[0][0]']

==================================================================================================
Total params: 88,079
Trainable params: 88,079
Non-trainable params: 0
__________________________________________________________________________________________________
```

Figure 5: Number of parameters in our BSautoconvnet

Compared to the CIFAR10 reconstruction models from the internet, our model is better. A CIFAR10 reconstruction model in the internet has millions of parameters and they usually use U-Net as the deep learning model. My BSautoconvnet has 44040 parameters (88079/2). See Figure 5. The actual number of parameters is the total number of parameters divided by 2. Because of the negated branch in the model (which can be combined with the positive branch), actual parameters is half of the total paramters. Visually inspecting our reconstructed image output, it has near perfect result. Most of the models in the internet such as U-net requires millions of parameters to achieve this result. We use much lesser parameters to achieve the same result.

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 an ordinary network to BSconvnet result in small improvement. But for neurons with high dimensions input, the improvement is very obvious.

Note that I use only convolutional layers, which helps to prevent it from overfitting the CIFAR10 dataset, verified from experiments. This design also reduces the total parameter count by a lot.

My BSautoconvnet is to mimic a Discrete Markov Random Field Relaxation model [1], but it is not exactly the same. The relaxation model is a convex optimization model, so it is convex. My BSautoconvnet has an optimization function which is smooth (much more convex than ordinary deep learning network optimization function), with the use of special convolutional layers.

Our model can be trained in 50 epoches. It is quite fast.

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

5 Conclusion

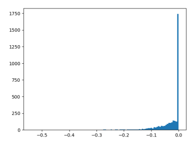
I have developed a Boolean Structured Autoencoder Convolutional Deep Learning Network for general noisy image reconstruction problem (CIFAR10 dataset) in machine learning. The design in this network can be easily transferred to other types of tasks (such as Imagenet dataset) for general reconstruction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

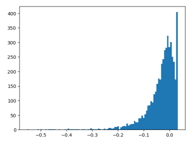
The Boolean structure in the network is able to provide a theoretical model on how deep learning works, how it learns and how it can be used to model any general high dimensions function with the help of my Discrete Markov Random Field Relaxation model.

A Appendix

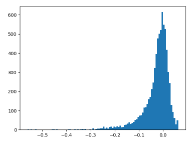
The output image contains 32 levels because it is binarized into 5 bits. For each level, the prediction error histograms (ground truth - prediction) are shown in the figures below (Figures 6 and 7). Note that the prediction errors are small (small variance in each histogram). And there is no noise that have non-zero mean. The noise are distributed like a zero mean Gaussian distribution. I hypothesize that it is impossible to do adversarial deep learning attack on my BSautoconvnet, because there is no non-zero mean noise in the error distributions.



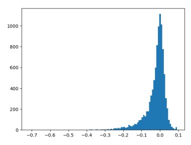
(a) Level 0



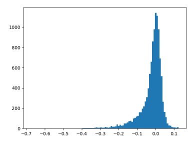
(b) Level 1



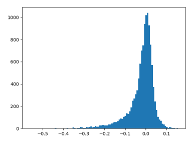
(c) Level 2



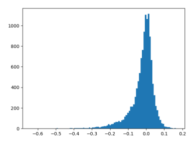
(d) Level 3



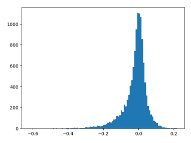
(e) Level 4



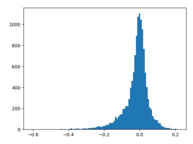
(f) Level 5



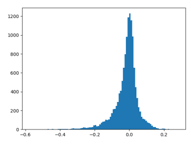
(g) Level 6



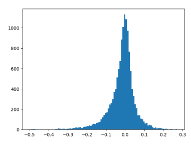
(h) Level 7



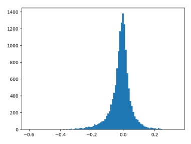
(i) Level 8



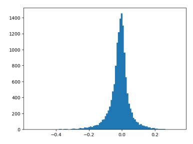
(j) Level 9



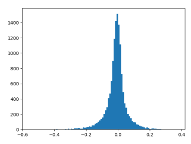
(k) Level 10



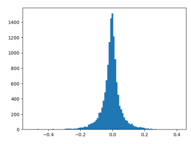
(l) Level 11



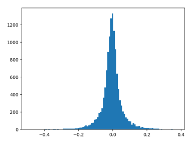
(m) Level 12



(n) Level 13

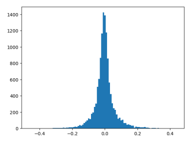


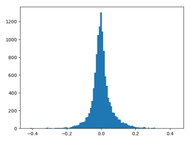
(o) Level 14

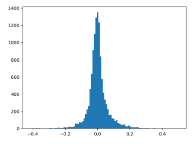


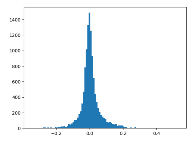
(p) Level 15

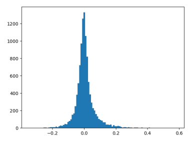
Figure 6: Prediction error for output at different levels from 0 to 15

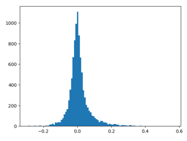

(a) Level 16

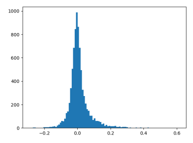

(b) Level 17

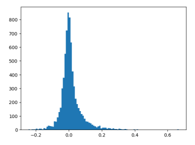

(c) Level 18

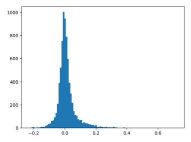

(d) Level 19

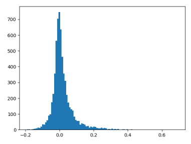

(e) Level 20

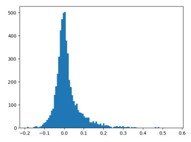

(f) Level 21

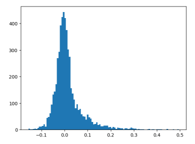

(g) Level 22

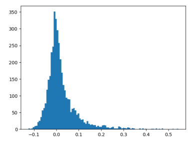

(h) Level 23

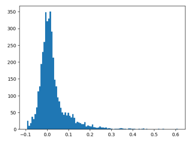

(i) Level 24

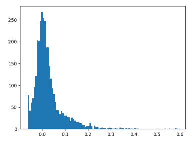

(j) Level 25


(k) Level 26


(l) Level 27


(m) Level 28


(n) Level 29


(o) Level 30

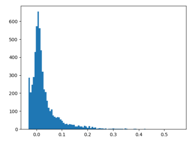

(p) Level 31

Figure 7: Prediction error for output at different levels from 16 to 31

References

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

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

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

[4] Sing Kuang Tan. Boolean structured convolutional deep learning network (bsconvnet). *https://vixra.org/abs/2305.0166*, 2023.