# Sensitivity Analysis of Batch Normalization in GAN


1st Zheng Li
*Civil and Environmental Engineering*
*Southern Methodist University*
Dallas,TX, USA
zli1@mail.smu.edu

2nd Yue Li
*Electrical Engineering*
*Southern Methodist University*
Dallas,TX, USA
yli3@mail.smu.edu

3rd Min Gautam
*Electrical Engineering*
*Southern Methodist University*
Dallas,TX, USA
mpgautam@smu.edu



***Abstract*—In this study, we trained a Deep Convolutional Generative Adversarial Net(DCGAN) and investigated several model settings to examine the impact of each variable on the model performance. Specifically, with the focus of batch normalization(BN) included architecture. Our initial attempt to train DCGAN did not success because of adding BN layers leads to model collapse. By altering variables such as batch size, kernel size, training method as well as other train techniques(adding noise), we found that when training discriminator using only real and fake images separately, our model works very with BN layers and received a quality enhancement. On the other hand, if the discriminator is trained using combined batch of both real and fakes images, the model failed to produce any good results with BN layers. We also discovered other techniques have minor/subtle impact on the training: 1) Large batch size tends to reduce the image quality. 2) Adding noise to labels only has minor impact of model performance. 3) BN should be applied to both generator and discriminator to produce good results. Lastly, model loss and layer gradient were discussed to confirm our findings.**



***Index Terms*—Generative Adversarial Networks, Batch Normalization, Sensitivity Analysis.**


## I. Introduction

In previous GAN lab exercise, we modified and trained the original GAN structure to Deep Convolution GAN(DCGAN) as provided in Keras-GAN [5]. The trained model is able to produce recognizable images with some artifacts. Then we tried to improve the model by adding Batch Normalization(BN) to the network(see two architectures in the Table I) following the suggestion mentioned by [6]. BN is supposed to give us better result since it can mitigate poor initialization and help gradients to flow into deeper models [4]. However, the result is even worse than the original one(Figure 1). It seems applying BN somehow make the GAN collapse into random noise which completely disobey what [4] concluded. This paper serves an investigation in what effects the training result of a DCGAN with BN and in which way could we improve the training result. We presented our experiments on four possible influence factors: Noise Labels, Batch Size, Kernel Size and Training Routine. In the following section, we will review and discuss the intuition of behind those factors.

## II. Experiments

This section provides several experiments that are designed to explore the cause of collapse issue after adding Batch normalization layer. The intuition of designing the following experiments was largely from trial and error since neural network is really difficult to diagnose with large set of parameters. With lots of existing heuristic "tricks" to improve GAN training, this project can also be considered as a toy GAN sensitivity analysis specifically on Batch normalization. Throughout all the experiments, we will be tracking(both generator and discriminator) loss, top layer gradient and bottom layer gradient and use those as the visualization tool to monitor the model convergence.

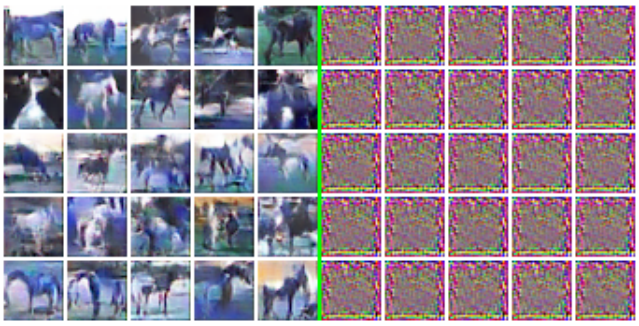

Fig. 1. (*Left*) Generated Images using DCGAN without BN layers; (*right*) Generated Images using DCGAN with BN layers

TABLE I
Architectures of different GANs

| | |
|---|---|
| **Example GAN** | G: C256-C256-C3<br>D: C128-C128-C128-C128 |
| **DC-GAN w/o BN** | G: C128-C64-C3<br>D: C32-C64-C128-C128 |
| **DC-GAN w/ BN** | G: C128(b)-C64(b)-C3<br>D: C32-C64(b)-C128(b)-C128(b) |

$^*C_k(b)$: CONV2D layer with k filters followed by BN layer

### A. Noise Labels

It has been documented in several papers [1]–[3], [7] that training a good GAN model requires some special implementations. One of those techniques is to add noise to labels while training the discriminator. Such noise will help model learn and accelerate convergence. In our experiments, we implemented both label smoothing and label flip in additional to BN structure. we randomly flipped 5 percent of the labels in our experiment.

### B. "One-sided" Batch normalization

One of the assumptions that we proposed during the diagnosis is that adding BN may or may not break one of the networks(either generator or discriminator), hence leads to collapse. Although it somehow contradicts the theory since [4] showed that the effect of BN is mostly positive (accelerate and stabilize training) regarding training the model. Out of curiosity, we ran one-sided Batch normalization (meaning only adding BN to one of the networks) to see if alters the result.

### C. Kernel Size & Number of filters

A large kernel size was experimented since it is commonly recommended in practice to allow generator/discriminator to take more information and enlarge models' capability to learn complex features. In our original implementation, kernel size was set to 3 for generator and 4 for discriminator with varying filters from layer to layer. In our experiments, we increased the kernel size for generator to 5(discriminator remains as 4) and use 128 filters in all layers.

### D. Batch Size

The batch size will affect how the mini-batch mean/variance are estimated in BN layer, so we experimented different batch size varying from 16 to 128.

### E. Training routine

The training process for GAN consists of two main steps:

1) Unfreeze weights in discriminator and train discriminator using images combined from real and fake images.
2) Freeze weights in discriminator and train the generator only using fake images.

While lots of online resources have shown success of using such training routine [1]–[3], an alternative training procedure is also proposed as follows:

1) Unfreeze weights in discriminator and train discriminator using real noised images only.
2) Train discriminator again using fake noised images only.
3) Freeze weights in discriminator and train the generator only using fake images.

This new train routine was also investigated in this project and has been proved to have an impact on model performance.

The number of variables we investigated is more than a few, due to the time limit, we did not run full scenarios analysis(meaning every possible combinations of parameters) and only ran 2 to 3 experiments for one proposed factor. This analysis can be easily extended if it is of interest in the future. A full summary of conducted experiments is presented in Table 2.

TABLE II
SUMMARY OF ALL EXPERIMENTS

| ID | Noise Labels | One-sided BN | Kernel | Batch | Training |
|---|---|---|---|---|---|
| 1 | NO | BOTH | 3 | 16 | SEPARATE |
| 2 | YES | BOTH | 3 | 16 | SEPARATE |
| 3 | NO | BOTH | 3 | 128 | COMBINED |
| 4 | YES | BOTH | 3 | 128 | COMBINED |
| 5 | YES | BOTH | 5 | 8 | SEPARATE |
| 6 | YES | BOTH | 5 | 16 | COMBINED |
| 7 | YES | BOTH | 5 | 16 | SEPARATE |
| 8 | YES | BOTH | 5 | 32 | SEPARATE |
| 9 | YES | BOTH | 5 | 64 | SEPARATE |
| 10 | YES | BOTH | 5 | 128 | COMBINED |
| 11 | YES | BOTH | 5 | 128 | SEPARATE |
| 12 | YES | G(0.5) | 5 | 128 | COMBINED |
| 13 | YES | G(0.7) | 5 | 128 | COMBINED |
| 14 | YES | G(0.9) | 5 | 128 | COMBINED |

[a]Default BN use 0.9 as momentum.
G(0.5): BN(momentum=0.5) in Generator.

## III. RESULTS&DISCUSSION

14 different models were run according to Table II, we used generated images, along with model loss, top layer gradient and bottom layer gradient to judge the model's overall performance. We will discuss the impact of each proposed variable accordingly as follows:

- Noise label
  Compare the upper and lower plot in Figure 2, we cannot find any major differences. So, we could presume that noise label has little effect on the model. Figure 3 proves our assumption. Although images generated from model with noise label(Figure 3 right) seems to have less blank spaces, the overall improvement is trivial.
- One-sided batch normalization
  The result shown in left-side Figure 5 is much better than the right-side which is basically random noise. It indicate that one-sided BN provides better result than both-side BN while other parameters are set as in Experiment 10 and 12. The comparison between right-side Figure 5 and left-side Figure 1 ensures us that adding BN layer would not collapse the model.
- Kernel size
  The results are shown in Figure 12 and Figure 13. Although it is said that larger kernel size could positively effect the model's learning ability, the improvement is unnoticeable in our experiment. The reason could be our limited image size(32x32). It could possibly show better results in images dataset with greater scales.
- Batch size
  Larger batch size seems to negatively influence the image quality as we observed from Figure 9. According to Figure 8, the overlapping area between the discriminator and generator is reducing as the batch size increases. It raises our interest in the relation between generated image quality and overlapping area of loss values.
- Training routine
  We discovered that both routines can lead to convergence(Previous lab used combined training method). However, in our experiments, we found that with the batch size =128, using combined training will not generate good result when BN layers introduced(Figure 5, 7 left). Separated training works very well with BN and greatly enhance the result compared to original model in

Figure 1. We can also tell from the loss/gradient plots in Figure 6 that under combined training, gradients received for top and bottom layer for discriminator dropped quickly and are close to zero after some iterations, which indicates discriminator stopped learning and as a consequence it also fails the generator training in the end.

## IV. Conclusion

By experimenting different variables, training a good GAN model is not an easy task. The competitive nature between generator and discriminator lead the GAN to be unstable and highly sensible to hyper-parameters. Therefore, hyper-parameters need to be carefully chosen to avoid any model collapse. In practice, it is hard to exploit all possible combinations of hyper-parameters. A bag of "training GAN tricks" thereby are developed to help improve the GAN training. This project was also motivated by the suffer from training a GAN where adding a Batch normalization layer would not bring any improvement to the model training and even worsen the result. This is against the argument mentioned in [4], [6]. Due to this conflicted finding, we examined a few parameters including noise label, batch size, kernel size as well training method to show the model performance with Batch normalization implementation. We conclude that:

- Under each iteration, training discriminator separately with real images only and fake images only will significant improve the model performance.
- A large batch size(128) tends to have minor negative impact on the quality of generated images from generator in compare to smaller batch size(16).
- Adding noise to labels has little impact on model performance, at least, in this experiment, we did not see any noticeable improvement when applying noise/random flipping to labels.
- Kernel size also appear to have less impact than our expectation, barely noticeable by human.
- Our experiment of one-sided batch normalization indicated that just adding BN to generator will not collapse the model. The model will still learn according to our analysis. But we can conclude that just adding one BN to generator do not as much quality boost as BN to both models.

Overall, training a good GAN is just like playing a frozen lake game, every decision you make has 40% chance to do nothing on the model, 40% chance to fail the model and 15% chance to crash your laptop and only **5%** chance for you to step on the right land.

## V. Appendix

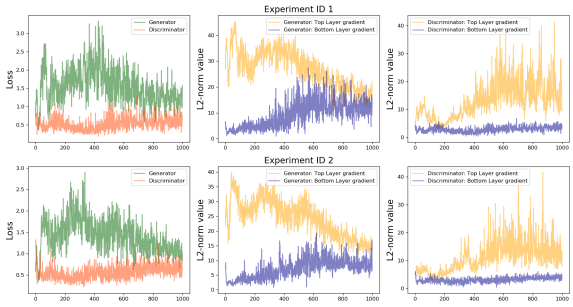


Fig. 2. Noise Label comparison

Fig. 3. Left: Images generated without noise labels(*Experiment 1*). Right: Images generated with noise label(*Experiment 2*).

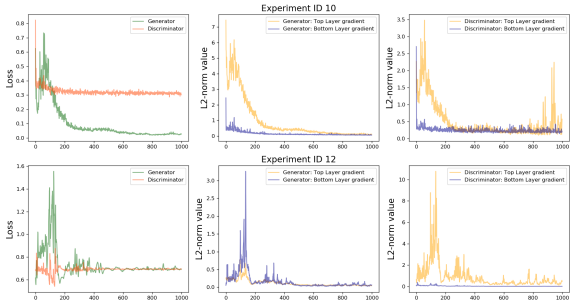


Fig. 4. Oneside BN comparison

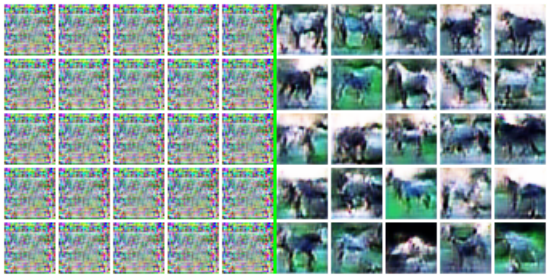

Fig. 5. Left: Images generated without both-side BN (Experiment 10). Right: Images generated with one-side BN(*Experiment 12*).

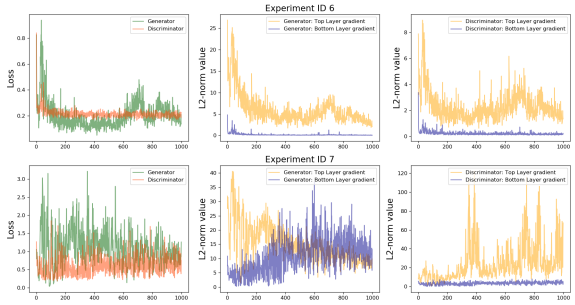


Fig. 6. Training routine comparison with batch size =16

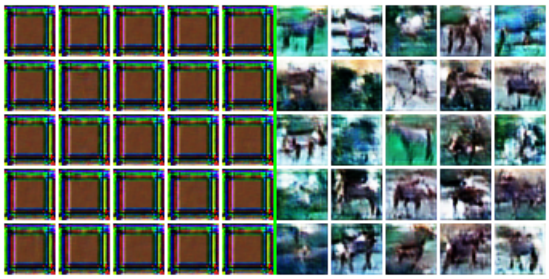

Fig. 7. Left: Images generated with combined training *(*Experiment 6). Right: Images generated with separated training(*Experiment 7*).

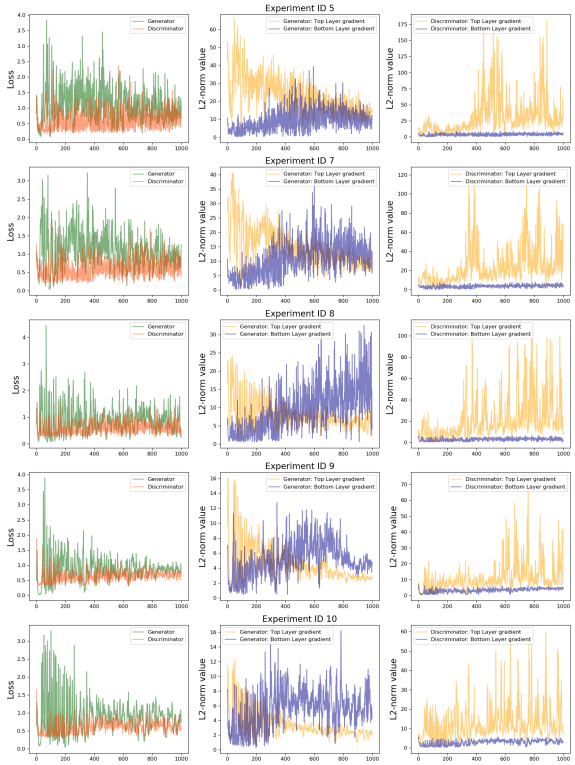


Fig. 8. Batch size comparison

Fig. 9. Images generated with batch size:8(*Experiment 5*), 16(*Experiment 7*), 32(*Experiment 8*), 64(*Experiment 9*), 128(*Experiment 10*)(from left to right).

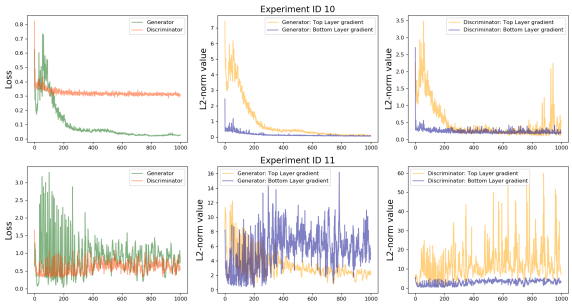


Fig. 10. Training routine comparison with batch size =128

Fig. 11. Left: Images generated with combined training (*Experiment 10*). Right: Images generated with separate training(*Experiment 11*).

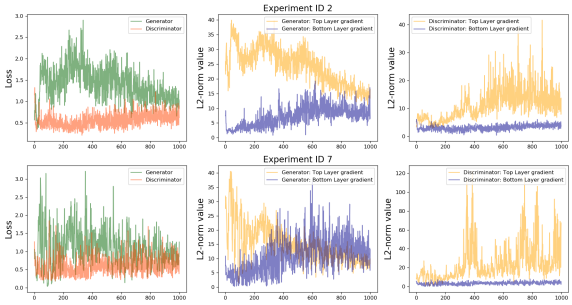


Fig. 12. Kernel size comparison

Fig. 13. Left: Images generated with kernel size=3 (Experiment 2). Right: Images generated with kernel size=5(*Experiment 7*).

## REFERENCES


[1] Decrypt generative artificial intelligence and gans. https://sergioskar.github.io/Generative$_{A}rtificial_{I}ntelligence/$.

[2] Utkarsh Desai. Keep calm and train a gan. pitfalls and tips on training generative adversarial networks. https://medium.com/@utk.is.here/keep-calm-and-train-a-gan-pitfalls-and-tips-on-training-generative-adversarial-networks-edd529764aa9.

[3] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[4] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[5] Erik Linder-Norn. Keras-gan. https://github.com/eriklindernoren/Keras-GAN/blob/master/dcgan/dcgan.py.

[6] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[7] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. In *Advances in neural information processing systems*, pages 2234–2242, 2016.