Semi-supervised learning based on generative adversarial network: a comparison between good GAN and bad GAN approach

Wenyuan Li
Electrical and Computer Engineering
University of California, Los Angeles
liwenyuan.zju@gmail.com*

Zichen Wang
Bioengineering
University of California, Los Angeles
zcwang0702@g.ucla.edu

Jiayun Li
Bioengineering
University of California, Los Angeles
jiayunli@ucla.edu

Jennifer Polson
Bioengineering
University of California, Los Angeles
jpolson@g.ucla.edu

William Speier
Radiology
University of California, Los Angeles
speier@ucla.edu

Corey Arnold
Radiology, Pathology, Bioengineering
University of California, Los Angeles
cwarnold@ucla.edu

Abstract

Recently, semi-supervised learning methods based on generative adversarial networks (GANs) have received much attention. Among them, two distinct approaches have achieved competitive results on a variety of benchmark datasets. Bad GAN learns a classifier with unrealistic samples distributed on the complement of the support of the input data. Conversely, Triple GAN consists of a three-player game that tries to leverage good generated samples to boost classification results. In this paper, we perform a comprehensive comparison of these two approaches on different benchmark datasets. We demonstrate their different properties on image generation, and sensitivity to the amount of labeled data provided. By comprehensively comparing these two methods, we hope to shed light on the future of GAN-based semi-supervised learning.

1. Introduction

Semi-supervised learning (SSL) aims to make use of large amounts of unlabeled data to boost model performance, typically when obtaining labeled data is expensive and time-consuming. Various semi-supervised learning methods have been proposed using deep learning and proven to be successful on several standard benchmarks. Weston *et al*. [25] employed a manifold embedding technique using the pre-constructed graph of unlabeled data; Rasmus *et al*. [21] used a specially designed auto-encoder to extract essential features for classification; Kingma and Welling [8] developed a variational auto encoder in the context of semi-supervised learning by maximizing the variational lower bound of both labeled and unlabeled data; Miyato *et al*. [16] proposed virtual adversarial training (VAT) that tied to find a deep classifier, which had a good prediction accuracy on training data and meanwhile was less sensitive to data perturbation towards the adversarial direction.

Recently, generative adversarial networks (GANs) [6], have demonstrated their capability in SSL frameworks [23, 3, 4, 2, 10, 12, 14]. GANs are a powerful class of deep generative models that are able to model data distributions over natural images [20, 15]. Salimans *et al*. first proposed to use GANs to solve a $(K+1)$-class classification problem, where the dataset contained K class originally and the additional $(K+1)$th class consisted of the synthetic images generated by the GAN's generator. Later on, Li *et al*. [2] realized that the generator and discriminator in [23] may not be optimal at the same time (*i.e.*, the discriminator was able to achieve good performance in SSL, while the generator may generate visually unrealistic images). They proposed a three-player game (Triple-GAN) to simultaneously achieve good classification results and obtained a good image generator. Dai *et al*. [3] realized the same problem, but

[*] W. Li and C. Arnold are the corresponding authors.

instead gave theoretical justifications of why using bad samples from the generator was able to boost SSL performance. Their model is called Bad GAN, which achieves state-of-the-art performance on multiple benchmark datasets. Another line of work focused on manifold regularization [1]. Kumar *et al*. [10] estimated the manifold gradients at input data points and added an additional regularization term to a GAN, which promoted invariance of the discriminator to all directions in the data space. Lecouat *et al*. [12] performed manifold regularization by approximating the Laplacian norm that was easily computed within a GAN and achieved competitive results.

In this paper, we focus on two GAN-based SSL models, Triple GAN and Bad GAN, and perform a comprehensive comparison between them. As both of models attempt to solve a similar issue in the original setting [23] but are motivated by dissimilar perspectives, we believe that our comparison will provide insight for future SSL research. For simplicity, we refer to Triple GAN as Good GAN in contrast to Bad GAN. In Section 2, we briefly review the two models and their different approaches for solving loss function incompatibility; in Section 3, we show the network architecture we employed, benchmark datasets we used, and hyperparameters we selected in order to perform a fair comparison between these two models; in Section 4, we demonstrate our comparison results and discuss several important aspects we found for these two models; we conclude our paper in Section 5.

2. Related Work

2.1. Bad GAN

Suppose we have a classification problem that requires classifying a data point $\boldsymbol{x}$ into one of K possible classes. A standard classifier takes in $\boldsymbol{x}$ as input and outputs a K-dimensional vector of logits $\{l_1, ..., l_K\}$. Salimens *et al*. [23] extend the standard classifier *C* by simply adding samples from the GAN generator *G* to the dataset, labeling them as a new "generated" class $y = K+1$, and correspondingly increasing the dimension of *C* output from K to $K+1$. The loss function $L_{C/D}$ for training *C* (*i.e.*, the extended discriminator *D* from the GAN's perspective) then becomes

$$\begin{aligned} L_{C/D} &= L_{\text{supervised}} + L_{\text{unsupervised}} \\ L_{\text{supervised}} &= \mathop{\mathbb{E}}_{\boldsymbol{x},y\sim p_l(\boldsymbol{x},y)} [-\log(p_{C/D}(y|\boldsymbol{x}, y < K+1))] \\ L_{\text{unsupervised}} &= \mathop{\mathbb{E}}_{\boldsymbol{x}\sim p_u(\boldsymbol{x})} [-\log(1 - p_{C/D}(y = K+1|\boldsymbol{x}))] \\ &+ \mathop{\mathbb{E}}_{\boldsymbol{x}\sim p_g(\boldsymbol{x})} [-\log(p_{C/D}(y = K+1|\boldsymbol{x}))] \end{aligned} \tag{1}$$

The supervised loss term $L_{\text{supervised}}$ is a traditional cross-entropy loss that is applied to labeled data $(\boldsymbol{x}, y) \sim p_l(\boldsymbol{x}, y)$. The unsupervised loss requires *C/D* to put the synthetic data from generator $\boldsymbol{x} \sim p_g(\boldsymbol{x})$ into the $(K+1)$th class, while putting the unlabeled data $\boldsymbol{x} \sim p_u(\boldsymbol{x})$ into the real K classes. For the generator, [23] found feature matching loss in Eq. 2 is the best in practice, though they generated visually unrealistic images. The feature matching loss is,

$$L_G = \left\| \mathop{\mathbb{E}}_{\boldsymbol{x}\sim p_u} (\boldsymbol{f}(\boldsymbol{x})) - \mathop{\mathbb{E}}_{\boldsymbol{z}_g\sim p_z(z)} (\boldsymbol{f}(G(\boldsymbol{z}_g))) \right\|_2^2 \tag{2}$$

where $\boldsymbol{z}_g \sim p_z(\boldsymbol{z})$ is drawn from a simple distribution such as uniform.

On the basis of this formulation, Dai *et al*. [3] give a theoretical justification on why the visually unrealistic images (*i.e.*, "bad" samples) from the generator could help with SSL. Loosely speaking, the carefully generated "bad" samples along with the loss function design in Eq. 1 could force *C*'s decision boundary to lie between the data manifolds of different classes, which in turn improves generalization of the classifier. Based on this analysis, they propose a Bad GAN model that learns a bad generator by explicitly adding a penalty term to generate "bad" samples. Their objective function of the generator becomes:

$$\begin{aligned} L_G = -\mathcal{H}[p_g(\boldsymbol{x})] + \mathop{\mathbb{E}}_{\boldsymbol{x}\sim p_g(\boldsymbol{x})} (\log p^{pt}(x)\, \mathbb{I}[p^{pt}(x) > \epsilon] \\ + \left\| \mathop{\mathbb{E}}_{\boldsymbol{x}\sim p_u(\boldsymbol{x})} (\boldsymbol{f}(\boldsymbol{x})) - \mathop{\mathbb{E}}_{\boldsymbol{z}_g\sim p_z(z)} (\boldsymbol{f}(G(\boldsymbol{z}_g))) \right\|_2^2 \end{aligned} \tag{3}$$

where the first term measures the negative entropy of the generated samples and tries to avoid collapsing while increasing the coverage of the generator. The second term explicitly penalizes generated samples that are in high density areas by using a pre-trained model, and the third term is the same feature matching term as in Eq. 2.

2.2. Good GAN

Li *et al*. [2] also noticed the same problem in [23] as the generator and the discriminator have incompatible loss functions, but took a different approach to tackling this issue. Intuitively, assume the generator can generate good samples in the original settings of [23], the discriminator should identify these samples as fake samples as well as predict the correct class for the generated samples. To address the problem, [2] present a three-player game called Triple-GAN that consists of a generator *G*, a discriminator *D*, and a separate classifier *C*. *C* and *D* are two conditional networks that generate pseudo labels given real data and pseudo data given real labels respectively. To jointly evaluate the quality of the samples from the two conditional networks, a single

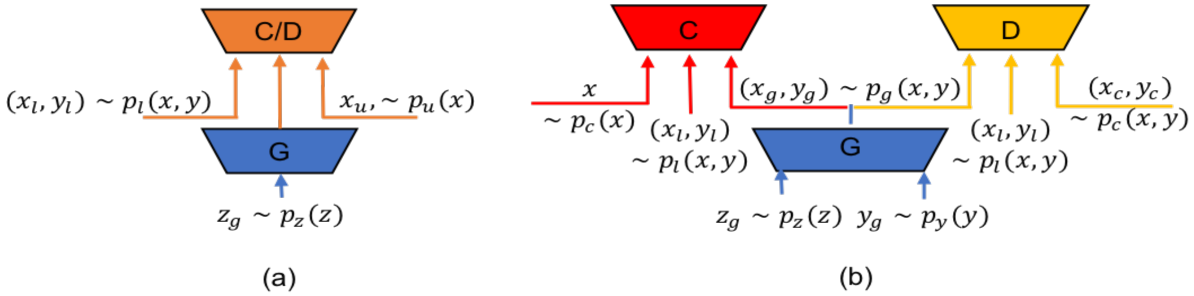


Figure 1. Network architecture of Bad GAN (a) and Good GAN (b). Bad GAN (a) consists of two parts: a generator *G* aims to generates "bad" samples, and a discriminator/classifier *D/C* that distinguishes real and fake samples and put the labeled samples into the right classes; Good GAN (b) consists of three parts: two conditional networks *G* and *C* that generate pseudo labels given real data and pseudo data given real labels respectively, and a separate discriminator *D* that distinguish the generated data-label pair from the real data-label pair.

discriminator *D* is used to distinguish whether a data–label pair is from the real labeled dataset or not. We refer this model as Good GAN because one of the aims for this formulation is to obtain a good generator.

The authors prove that instead of competing equilibrium states as in [23], Good GAN has the unique global optimum for both *C* and *G*, *i.e.*, $p(\boldsymbol{x}, y) = p_g(\boldsymbol{x}, y) = p_c(\boldsymbol{x}, y)$, the three joint distributions match one another. In other words, a good classifier will result in a good generator and vice versa. Furthermore, Good GAN is trained using the REINFORCE algorithm, in which it generates pseudo labels through *C* for some unlabeled data and uses these pairs as positive samples to feed into *D*. This is a key to the success of the model, as one of the crucial problems of SSL is the limited size of the labeled data. Figure 1 shows the network architecture of Good GAN and Bad GAN.

3. Comparison Method

3.1. Network Architecture

In Bad GAN, the discriminator has two roles: to classify the real data into the right class and to distinguish the real samples from the fake samples. For clarity, we refer to Bad GAN's discriminator as the classifier, since its input and output are exactly the same as the classifier in Good GAN due to the over-parameterization of the softmax layer [23].

To perform a fair comparison between Good GAN and Bad GAN, we use the same network architecture for the generator *G* and the classifier *C* in both models. We follow the architecture closely in [2] to set up the additional discriminator *D* in Good GAN. Both of them use Leaky-Relu activation and weight normalization to ease the difficulty of GAN's training. Implementing them using same architecture ideally avoids the possibility of using an architecture that is custom-tailored to work well with one or the other. Detailed model architectures can be found in the Appendix A.

3.2. Datasets

Using the above-defined network architectures, we compare the two models on the widely adopted MNIST [13], SVHN [17], and CIFAR10 [9] datasets. MNIST consists of 50,000 training samples, 10,000 validation samples, and 10,000 testing samples of handwritten digits of size 28×28. SVHN consists of 73,257 training samples and 26,032 testing samples. Each sample is a colored image of size 32×32, containing a sequence of digits with various backgrounds. CIFAR10 consists of colored images distributed across 10 general classes – *airplane*, *automobile*, *bird*, *cat*, *deer*, *dog*, *frog*, *horse*, *ship* and *truck*. It contains 50,000 training samples and 10,000 testing samples of size 32×32. Following [2], we reserve 5,000 training samples from SVHN and CIFAR10 for validation if needed. For our CIFAR10 experiment, we perform zero-based component analysis (ZCA) [11] as suggested in [2] for the input of *C*, but still generate and estimate the raw images using *G* and *D*.

We perform an extensive investigation by varying the amount of labeled data. Following common practice, this is done by throwing away different amounts of the underlying labeled dataset [23, 19, 22, 24]. The labeled data used for training are randomly selected stratified samples unless otherwise specified. We perform our experiments on setups with 20, 50, 100, and 200 labeled examples in MNIST, 500, 1000, and 2000 labeled examples in SVHN, and 1000, 2000, 400, 8000 examples in CIFAR10.

3.3. Hyperparameter Selection

For the hyperparameter selection such as learning rate and beta for Adam optimization, and the coefficient for each cost function term, we closely follow [2, 3]. In addition, we perform extensive study of the effects of batch size on performance for Bad GAN. As reported by [12], Bad GAN training is sensitive to training batch size, and thus we vary batch size in the training phase and compare their final performances on MNIST and SVHN.

4. Experimental Results and Discussion

We implement Good GAN based on Tensorflow 1.10 [5] and Bad GAN based on Pytorch 1.0 [18]. The generated images from gG is not applied until the number of epochs reach a threshold that gG could generate reliable imagelable pairs. We choose 200 in all three cases. All of the other hyperparameters including initial learning rate, maximum epoch number, relative weights and parameters in Adam [7] are fixed according to [23, 2, 3] across all of the experiments.

4.1. Classification

We report our classification accuracy on the test set in Table 1, Table 2 and Table 3 for MNIST, SVNH and CIFAR10, respectively, along with the results reported in the original papers. The similarity of our results to those reported in the original papers suggests that our reproduced models are accurate instantiations of Good GAN and Bad GAN. Furthermore, we perform extensive study by varying the amount of labeled data and observe that Good GAN and Bad GAN behave quite differently under various circumstances.

First, with a medium amount of labeled data (*e.g.*, MNIST with 100 or 200 labeled data, SVHN with more than 2000 labeled data, or CIFAR10 with more than 2000 labeled data), Bad GAN performs better than Good GAN. In fact, to the best of our knowledge, Bad GAN achieves the current state-of-the-art performance on those benchmark datasets. However, with low amounts of labeled data, Good GAN performs better, which demonstrates that Good GAN is less sensitive to the amount of labeled data than Bad GAN. One possible explanation is due to the use of the REINFORCE algorithm in Good GAN, because it generates pseudo labels through *C* for some unlabeled data and use these pairs as positive samples of *D*. Since *C* converges quickly, this trick provides a clever way to enable the generator to explore a much larger data manifold that includes both the labeled and unlabeled data information. In other words, the classifier is able to provide pseudo labels for the unlabeled data, while the discriminator will judge if the pseudo labels are reliable or not throughout the training. This in return will affect the evolution of the generator, which will take advantage of the unlabeled data to generate good images. Generated good image-label pairs that implicitly contain unlabeled data information will eventually benefit the classifier. This works extremely well for relatively simple datasets like MNIST, as Good GAN is able to model the class-awarded data distribution through weak supervision. On the other hand, Bad GAN yields decreased performance when the amount of labeled data is low, as it does not have any mechanism to augment the information that could be used to train the classifier in this case.

4.2. Generated Images

In Figure 2, we compare the quality of images generated by Good GAN and Bad GAN. As can be seen, Good GAN is able to generate clear images and meaningful samples conditioned on class labels, while Bad GAN generates "bad" images that look like a fusion of samples from different classes. In addition, Good GAN is able to disentangle classes and styles. In Figure 2 bottom, we vary the class label y in the vertical axis and the latent vectors z in the horizontal axis to generate the images. As shown in the figure, the latent vector z encodes meaningful physical appearances, such as scale, intensity, orientation, color and so on, while the label y controls the semantics of the generated images. Furthermore, Good-GAN can transition smoothly from one style to another with different visual factors without losing the label information as shown in Figure 3. This proves that Good GAN can learn meaningful latent space representations instead of simply memorizing the training data.

4.3. Importance of Selection of Labeled Data

Another interesting observation is that the selection of labeled data plays a crucial role for training Good GAN model in the low labeled data scenario. As mentioned above, the labeled data used for the training are randomly selected stratified samples, except for the MNIST-20 case. In this case, we found selecting representative labeled data to train is the key to achieving good performance. The reported accuracy in Table 1 is averaged over 10 runs where we manually selected different representative labeled data in a stratified way. Figure 4 (a) shows a single run that uses randomly selected labeled data and does not achieve good results, while Figure 4 (b) shows another run that is able to achieve higher accuracy. The failure of the first run is due to the initial selections for digit 4 being similar to 9, causing the generator to generate many 9s when conditioned on label 4. The generator also generates low-quality images. We also report that with a random selection of 20 labeled data, the Good GAN was able to achieve $76.78 \pm 6.47\%$ accuracy over 3 runs.

4.4. Importance of Batch Size

We found that batch size largely affect the final training results, in both Good GAN and Bad GAN. To investigate the effect of batch size on Bad GAN performance, we performed experiments with different batch size on MNIST (with 100 labeled samples) and SVHN (with 1000 labeled samples) using Bad GAN. As shown in Table 4, we empirically show that the performance of Bad GAN is sensitive to training batch size, and the optimal performance for each dataset is achieved with a batch size of 100.

To further understand the effect of the batch size on Bad GAN training, we present the generator loss with different

Table 1. Test accuracy on semi-supervised MNIST. Results are averaged over 10 runs. * denotes the special selection of labeled data. See details in Section 4.3.

Model	Test accuracy for a given number of labeled samples			
	20	50	100	200
Bad GAN [3]	-	-	$\mathbf{99.21} \pm \mathbf{0.01}\%$	-
Triple GAN [2]	$95.19 \pm 4.95\%$	$98.44 \pm 0.72\%$	$99.09 \pm 0.58\%$	$\mathbf{99.33} \pm \mathbf{0.16}\%$
Bad GAN (ours)	$68.12 \pm 0.60\%$	$96.24 \pm 0.16\%$	$99.17 \pm 0.03\%$	$99.20 \pm 0.03\%$
Good GAN (ours)	$\mathbf{95.93} \pm \mathbf{4.45}\%$*	$\mathbf{98.68} \pm \mathbf{1.12}\%$	$99.07 \pm 0.46\%$	$99.17 \pm 0.08\%$

Table 2. Test accuracy on semi-supervised SVHN. Results are averaged over 10 runs.

Model	Test accuracy for a given number of labeled samples		
	500	1000	2000
Bad GAN[3]	-	$\mathbf{95.75} \pm \mathbf{0.03}\%$	-
Triple GAN[2]	-	$94.23 \pm 0.17\%$	-
Bad GAN (ours)	$94.21 \pm 0.45\%$	$95.32 \pm 0.07\%$	$\mathbf{95.47} \pm \mathbf{0.39}\%$
Good GAN (ours)	$\mathbf{94.67} \pm \mathbf{0.12}\%$	$95.30 \pm 0.38\%$	$95.37 \pm 0.09\%$

Table 3. Test accuracy on semi-supervised CIFAR10. Results are averaged over 10 runs.

Model	Test accuracy for a given number of labeled samples			
	1000	2000	4000	8000
Bad GAN [3]	-	-	$\mathbf{85.59} \pm \mathbf{0.03}\%$	-
Triple GAN [2]	-	-	$83.01 \pm 0.36\%$	-
Bad GAN (ours)	$77.58 \pm 0.17\%$	$81.36 \pm 0.08\%$	$82.89 \pm 0.13\%$	$\mathbf{85.47} \pm \mathbf{0.10}\%$
Good GAN (ours)	$\mathbf{81.08} \pm \mathbf{0.57}\%$	$\mathbf{81.79} \pm \mathbf{0.37}\%$	$82.82 \pm 0.41\%$	$85.37 \pm 0.18\%$

batch sizes for MNIST and SVHN in Figure 5. The results indicate that smaller batch sizes lead to larger generator loss in the final stage of training. As that generator loss mainly depends on the first-order feature matching loss in Bad GAN, an intuitive explanation could be that larger batch sizes reduce the variance of the sample mean, allowing the generator to quickly approximate the entire training set. This leads to smaller generator loss, especially when model training becomes more stable in the final stage.

As noted by [3], feature matching is performing distribution matching in a weak manner, which could be largely affected by batch size. On one extreme, when the batch size is too small, the power of the generator in distribution matching is weak due to the excessive generator loss. Generated samples are therefore more likely to diverge from the manifold. Especially when data complexity increases, it is more difficult to minimize the KL divergence between the generator distribution and a desired complement distribution in Bad GAN, which could be one possible reason why model degradation is more significant on SVHN when using 20 batch size. On the other extreme, larger batch size leads to smaller generator loss, which comes with reduced diversity of generated samples. When the batch size is too large, the small generator loss will lead to a collapsed generator which fails to generate diverse samples that cover complement manifolds. As a result, the decision boundary between such missing manifolds becomes under-determined, which will also degrades model performance. We plot Bad GAN performance under different batch sizes for MNIST and SVHN in Appendix B.

Based on our experience, Good GAN is best when we use a large batch size. Intuitively, a small batch size is not good for the REINFORCE algorithm adopted in Good GAN because a single wrong prediction of the unlabeled data will have a big impact on the weight update in each iteration. We perform Good GAN experiments on SVHN using different batch size. The results are shown in Table 5. Empirically, we find that with small batch size, Good GAN is not able to generate good image-label pairs, hence the generated image-label pairs even hurt the classifier's performance when we use them to train. (See more details in Appendix B).

5. Conclusion

In this paper, we systematically and extensively compared two GAN-based SSL methods, Good GAN and Bad GAN, by applying these two models with commonly-used benchmark datasets. We illustrate the distinct characteris-

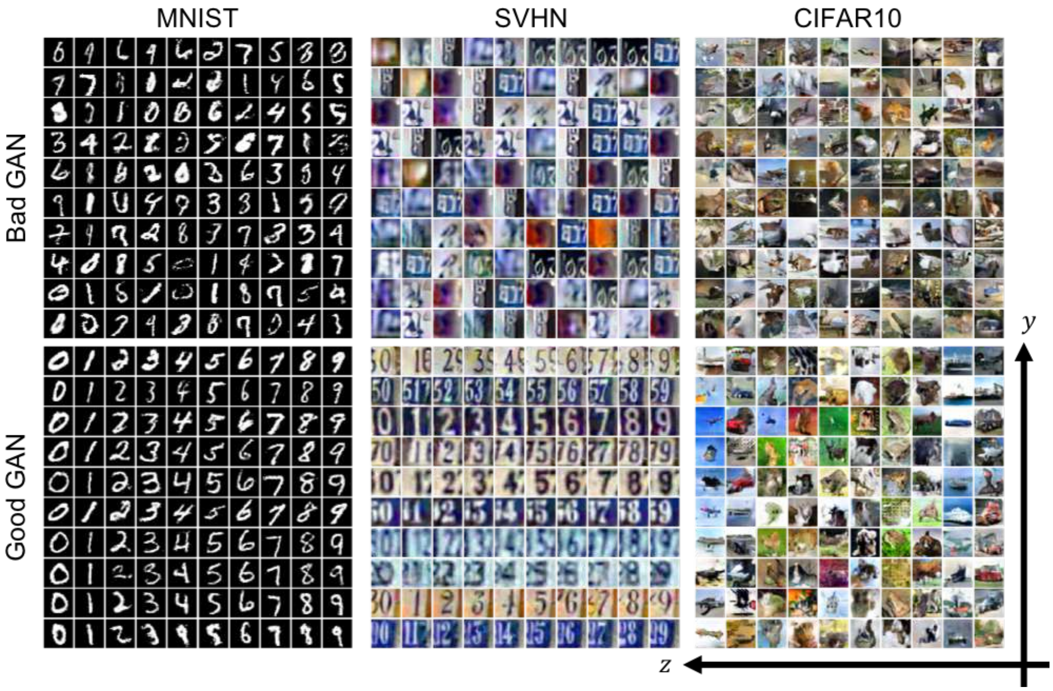


Figure 2. Generated images from both Bad GAN (top) and Good GAN (bottom). The images generated from Good GAN are produced by varying the class label y in the vertical axis and the latent vector z in the horizontal axis.

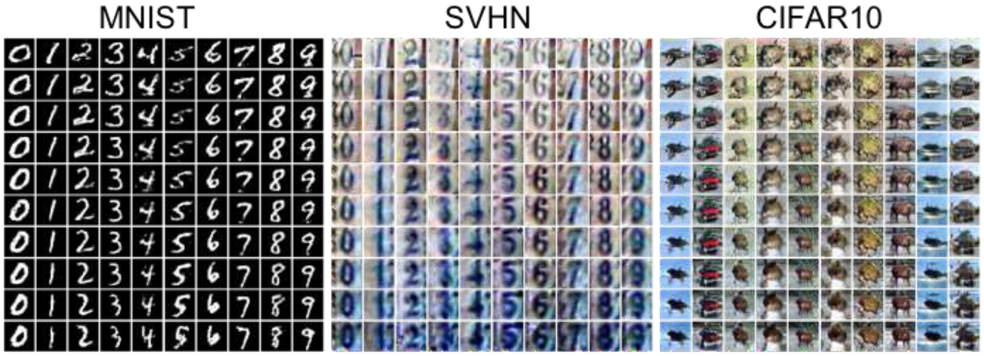


Figure 3. Class-conditional latent space interpolation. We first sample two random latent vectors z and linearly interpolate them. Then we map these vectors to the image space conditioned on each class y. The vertical axis is the direction for latent vector interpolation while the horizontal axis is the direction for varying the class labels.

Table 4. Bad GAN performance versus batch size on MNIST and SVHN. The results are achieved using 100 labeled samples in MNIST and 1000 labeled samples in SVHN.

Batch size	20	50	100	200	400
MNIST-100	$98.90 \pm 0.04\%$	$99.10 \pm 0.03\%$	$\mathbf{99.17 \pm 0.03}\%$	$99.16 \pm 0.03\%$	$98.89 \pm 0.02\%$
SVHN-1000	$93.35 \pm 0.05\%$	$95.29 \pm 0.03\%$	$\mathbf{95.56 \pm 0.02}\%$	$95.19 \pm 0.02\%$	$94.20 \pm 0.04\%$

tics of the images they generated, as well as each models sensitivity to varying the amount of labeled data used for training. In the case of low amounts of labeled data, model performance is contingent on the selection of labeled samples; that is, selecting non-representative samples results in generating incorrect image-label pairs and deteriorating classification performance. Furthermore, selecting the optimal batch size is crucial to achieve good results in both models. Notably, Good GAN and Bad GAN models can be used for complementary purposes; Good GAN gen-

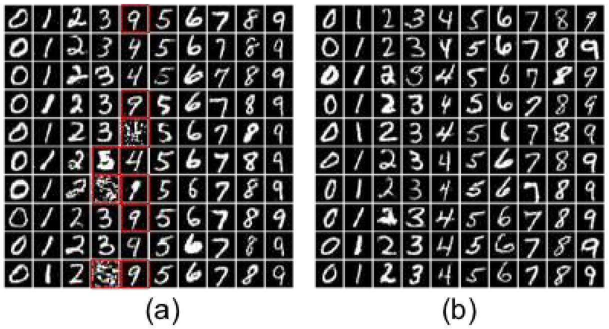


Figure 4. Two-runs of Good GAN model on MNIST dataset. (a) A single run where we randomly select 20 labeled data. The generator generates a lot of wrong images conditioned on the label and the classifier has lower performance. (b) Another run where we manually select 20 representative labeled examples. This time the generator is able to generate correct images, and the classifier achieves good classification performance.

Table 5. Good GAN performance versus batch size on SVHN. The results are achieved using 1000 labeled samples in SVHN.

Batch size	20	50	100
SVHN-1000	92.47%	92.59%	**95.30%**

erates good image-label pairs to train the classifier, while Bad GAN generates samples that force the decision boundary between data manifold of different classes. We envision that combining these two methods should yield further performance improvement in SSL.

Acknowledgements

The authors would like to acknowledge support from the UCLA Radiology Department Exploratory Research Grant Program (16-0003) and NIH/NCI R21CA220352. This research was also enabled in part by GPUs donated by NVIDIA Corporation.

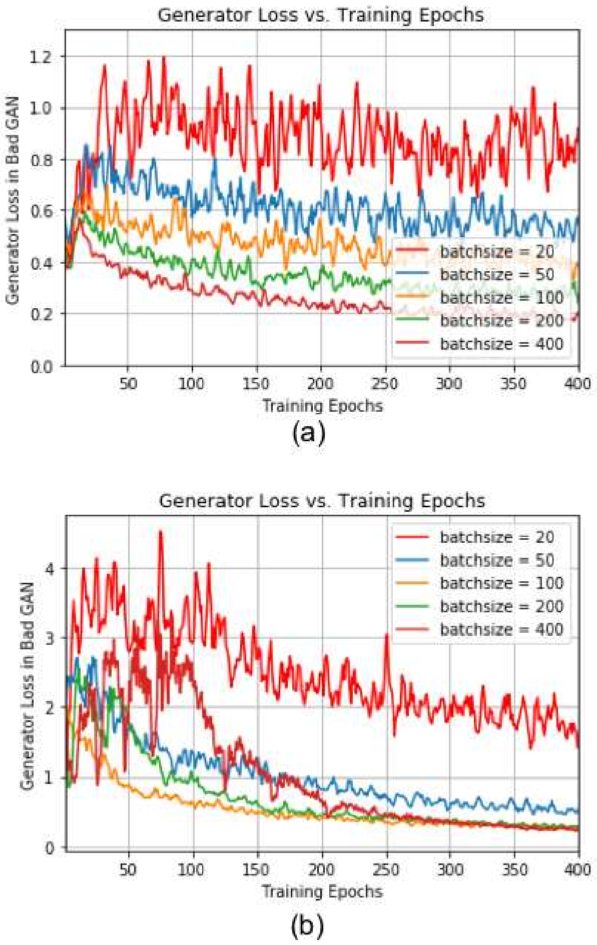


Figure 5. Batch size effect on generator loss in Bad GAN. The experiments are performed on (a) MNIST using 100 labeled samples and (b) SVHN using 1000 labeled samples.

References

[1] Mikhail Belkin, Partha Niyogi, and Vikas Sindhwani. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *Journal of machine learning research*, 7(Nov):2399–2434, 2006. 2

[2] LI Chongxuan, Taufik Xu, Jun Zhu, and Bo Zhang. Triple generative adversarial nets. In *Advances in neural information processing systems*, pages 4088–4098, 2017. 1, 2, 3, 4, 5

[3] Zihang Dai, Zhilin Yang, Fan Yang, William W Cohen, and Ruslan R Salakhutdinov. Good semi-supervised learning that requires a bad gan. In *Advances in neural information processing systems*, pages 6510–6520, 2017. 1, 2, 3, 4, 5

[4] Zhe Gan, Liqun Chen, Weiyao Wang, Yuchen Pu, Yizhe Zhang, Hao Liu, Chunyuan Li, and Lawrence Carin. Triangle generative adversarial networks. In *Advances in Neural Information Processing Systems*, pages 5247–5256, 2017. 1

[5] Sanjay Surendranath Girija. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *Software available from tensorflow. org*, 2016. 4

[6] 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. 1

[7] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 4

[8] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013. 1

[9] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009. 3

[10] Abhishek Kumar, Prasanna Sattigeri, and Tom Fletcher. Semi-supervised learning with gans: Manifold invariance with improved inference. In *Advances in Neural Information Processing Systems*, pages 5534–5544, 2017. 1, 2

[11] Samuli Laine and Timo Aila. Temporal ensembling for semi-supervised learning. *arXiv preprint arXiv:1610.02242*, 2016. 3

[12] Bruno Lecouat, Chuan-Sheng Foo, Houssam Zenati, and Vijay R Chandrasekhar. Semi-supervised learning with gans: Revisiting manifold regularization. *arXiv preprint arXiv:1805.08957*, 2018. 1, 2, 3

[13] Yann LeCun, Léon Bottou, Yoshua Bengio, Patrick Haffner, et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. 3

[14] Wenyuan Li, Yunlong Wang, Yong Cai, Corey Arnold, Emily Zhao, and Yilian Yuan. Semi-supervised rare disease detection using generative adversarial network. *arXiv preprint arXiv:1812.00547*, 2018. 1

[15] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014. 1

[16] Takeru Miyato, Shin-ichi Maeda, Shin Ishii, and Masanori Koyama. Virtual adversarial training: a regularization method for supervised and semi-supervised learning. *IEEE transactions on pattern analysis and machine intelligence*, 2018. 1

[17] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. In *Advances in neural information processing systems*, 2011. 3

[18] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. In *Advances in neural information processing systems Workshop*, 2017. 4

[19] Yunchen Pu, Zhe Gan, Ricardo Henao, Xin Yuan, Chunyuan Li, Andrew Stevens, and Lawrence Carin. Variational autoencoder for deep learning of images, labels and captions. In *Advances in neural information processing systems*, pages 2352–2360, 2016. 3

[20] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015. 1

[21] Antti Rasmus, Mathias Berglund, Mikko Honkala, Harri Valpola, and Tapani Raiko. Semi-supervised learning with ladder networks. In *Advances in neural information processing systems*, pages 3546–3554, 2015. 1

[22] Mehdi Sajjadi, Mehran Javanmardi, and Tolga Tasdizen. Mutual exclusivity loss for semi-supervised deep learning. In *2016 IEEE International Conference on Image Processing (ICIP)*, pages 1908–1912. IEEE, 2016. 3

[23] 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. 1, 2, 3, 4

[24] Antti Tarvainen and Harri Valpola. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. In *Advances in neural information processing systems*, pages 1195–1204, 2017. 3

[25] Jason Weston, Frédéric Ratle, Hossein Mobahi, and Ronan Collobert. Deep learning via semi-supervised embedding. In *Neural Networks: Tricks of the Trade*, pages 639–655. Springer, 2012. 1

Appendices

A. Network Architecture

We list the detailed architecture we used to compare Good GAN and Bad GAN on MNIST, SVHN, and CIFAR10 datasets in Table 6, Table 7 and Table 8 respectively.

Table 6. MNIST

Generator G	Classifier C	Discriminator D (Good GAN only)
Input Label y, Noise z	Input 28×28 Gray Image	Input 28×28 Gray Image, Label y
MLP 500 units, softplus, batch norm MLP 500 units, softplus, batch norm MLP 500 units, softplus, batch norm	MLP 1000 units, lRelu, Gaussian noise, weight norm MLP 500 units, lRelu, Gaussian noise, weight norm MLP 250 units, lRelu, Gaussian noise, weight norm MLP 250 units, lRelu, Gaussian noise, weight norm MLP 250 units, lRelu, Gaussian noise, weight norm MLP 10 units, softmax, Gaussian noise, weight norm	MLP 1000 units, lRelu, Gaussian noise, weight norm MLP 500 units, lRelu, Gaussian noise, weight norm MLP 250 units, lRelu, Gaussian noise, weight norm MLP 250 units, lRelu, Gaussian noise, weight norm MLP 250 units, lRelu, Gaussian noise, weight norm MLP 12 units, sigmoid, Gaussian noise, weight norm

Table 7. SVHN

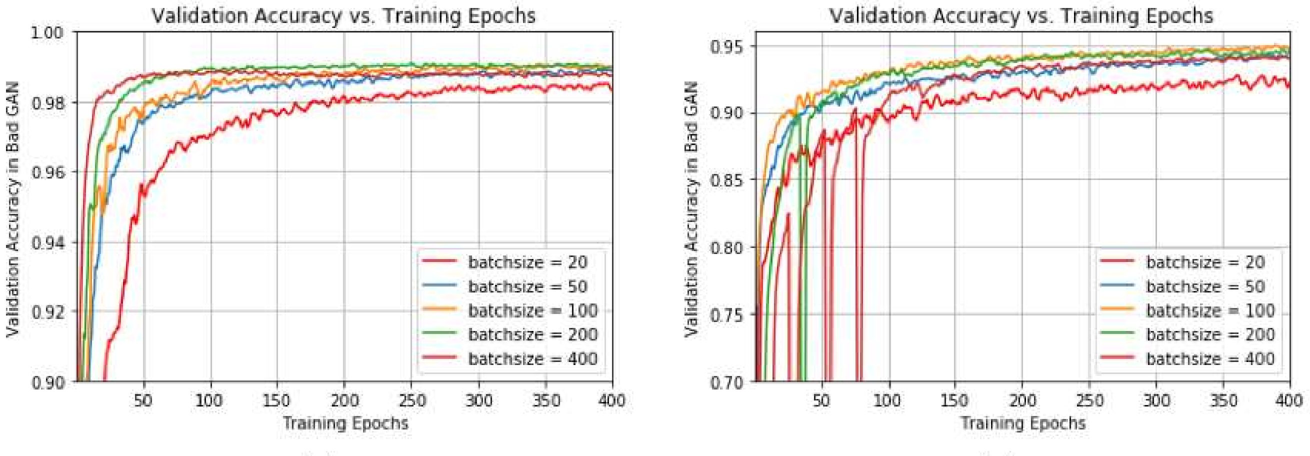
Generator G	Classifier C	Discriminator D (Good GAN only)
Input Label y, Noise z	Input 32×32 Colored Image	Input 32×32 Colored Image, Label y
MLP 8192 units, Relu, batch norm Reshape $512 \times 4 \times 4$ 5×5 deconv. 256. stride 2, Relu, batch norm	Gaussian noise, 0.2 dropout 3×3 conv. 64. lRelu, weight norm 3×3 conv. 64. lRelu, weight norm 3×3 conv. 64. lRelu, stride 2, weight norm 0.5 dropout	0.2 dropout 3×3 conv. 32. lRelu, weight norm 3×3 conv. 32. lRelu, stride 2, weight norm 0.2 dropout
5×5 deconv. 128. stride 2, Relu, batch norm	3×3 conv. 128. lRelu, weight norm 3×3 conv. 128. lRelu, weight norm 3×3 conv. 128. lRelu, stride 2, weight norm 0.5 dropout	3×3 conv. 64. lRelu, weight norm 3×3 conv. 64. lRelu, stride 2, weight norm 0.2 dropout
5×5 deconv. 3. stride 2, sigmoid, weight norm	3×3 conv. 128. lRelu, weight norm 3×3 conv. 128. lRelu, weight norm 3×3 conv. 128. lRelu, weight norm Global pool MLP 10 units, softmax, weight norm	3×3 conv. 128. lRelu, weight norm 3×3 conv. 128. lRelu, weight norm Global pool MLP 1 unit, sigmoid, weight norm

B. Batch Size Effect in Bad GAN

Figure 7 shows the classification accuracy under different batch size of Bad GAN during the first 400 epochs of training. As can be seen, the model performance is very sensitive to batch size. Figure 6 shows the generated images of Good GAN under different batch size. With small batch size, Good GAN is not able to generate good image-label pairs.

Table 8. CIFAR10

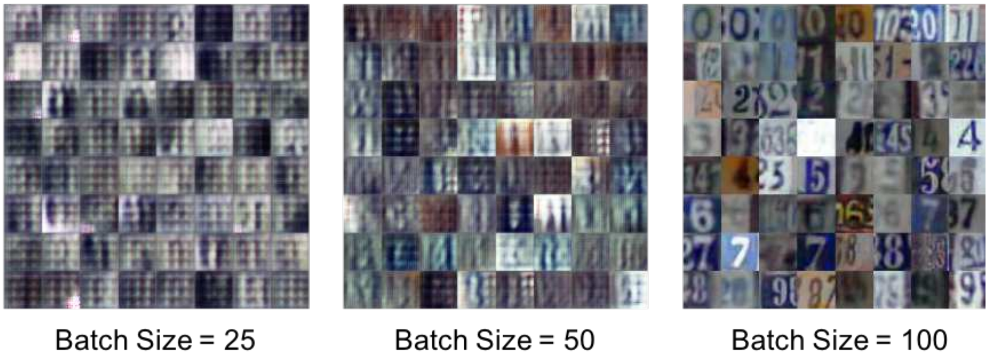
Generator G	**Classifier C**	**Discriminator D (Good GAN only)**
Input Label y, Noise z	Input 32×32 Colored Image	Input 32×32 Colored Image, Label y
MLP 8192 units, Relu, batch norm Reshape $512 \times 4 \times 4$ 5×5 deconv. 256. stride 2, Relu, batch norm	Gaussian noise, 0.2 dropout 3×3 conv. 96. lRelu, weight norm 3×3 conv. 96. lRelu, weight norm 3×3 conv. 96. lRelu, stride 2, weight norm 0.5 dropout	0.2 dropout 3×3 conv. 32. lRelu, weight norm 3×3 conv. 32. lRelu, stride 2, weight norm 0.2 dropout
5×5 deconv. 128. stride 2, Relu, batch norm	3×3 conv. 192. lRelu, weight norm 3×3 conv. 192. lRelu, weight norm 3×3 conv. 128. lRelu, stride 2, weight norm 0.5 dropout	3×3 conv. 64. lRelu, weight norm 3×3 conv. 64. lRelu, stride 2, weight norm 0.2 dropout
5×5 deconv. 3. stride 2, sigmoid, weight norm	3×3 conv. 192. lRelu, weight norm 3×3 conv. 192. lRelu, weight ntheirorm 3×3 conv. 192. lRelu, weight norm Global pool MLP 10 units, softmax, weight norm	3×3 conv. 128. lRelu, weight norm 3×3 conv. 128. lRelu, weight norm Global pool MLP 1 unit, sigmoid, weight norm



(a)

(b)

Figure 6. Batch size effect in Bad GAN. The classification accuracy over the initial 400 training epochs under different batch size. (a) The experiments are performed on MNIST dataset, using 100 labeled data. (b) The experiments are performed on SVHN dataset, using 1000 labeled data.



Batch Size = 25 Batch Size = 50 Batch Size = 100

Figure 7. Batch size effect in Good GAN. With small batch size, Good GAN is not able to generate good image-label pairs. Experiments are performed on SVHN with $n = 1000$. All the images are generated at epoch $= 200$ when we start to use the generated image to train.