Journal of Machine Learning Research 1 (2018) 1-48 Submitted 4/00; Published 10/00

Table 1: DC-GAN generator

Dimension	Module
100-512	2D transposed Convolutional operator with kernel size as (4,4), stride as (1,1)
512	2D Batch Normalization
512	ReLU unit
512-256	2D transposed Convolutional operator with kernel size as (4,4), stride as (2,2)
256	2D Batch Normalization
256	ReLU unit
256-128	2D transposed Convolutional operator with kernel size as (4,4), stride as (2,2)
128	2D Batch Normalization
128	ReLU unit
128-64	2D transposed Convolutional operator with kernel size as (4,4), stride as (2,2)
64	2D Batch Normalization
64	ReLU unit
64-3	2D transposed Convolutional operator with kernel size as (4,4), stride as (2,2)
3	Tanh unit

On the training of W-GAN network

Xin Man XIN-MAN@UIOWA.EDU
Department of Computer Science
University of Iowa
Iowa city, IA 52240, USA

Editor: Xin Man

Abstract

This reposrt presents the results and efforts in W-GAN,i.e. the Wassertein Generative Adversarial Network, training on various datasets. We explore the optimization method in versatility since generative adversarial network are known to be difficult to train. We used SGD, SGD with momentum, Adagrad, RMSprop and Adam on datasets such as MNIST, Cifar10, Cifar100. We also reproduced the result from the original paper, proving the validity of their work.

1. Experiment setup

1.1 Network Structure

We employed the same network structure admitted in the original work (Arjovsky et al., 2017), which is DC-GAN structure for both generator and discriminator. See Table.1 and Table.2

Table 2: DC-GAN discriminator

Dimension	Module
3,64	2D Convolutional operator with kernel size as (4,4), stride as (2,2)
64	Leaky ReLU with negative slope 0.2
64-128	2D Convolutional operator with kernel size as (4,4), stride as (2,2)
128	2D Batch Normalization
128	Leaky ReLU with negative slope 0.2
128-256	2D Convolutional operator with kernel size as (4,4), stride as (2,2)
256	2D Batch Normalization
256	Leaky ReLU with negative slope 0.2
256-512	2D transposed Convolutional operator with kernel size as (4,4), stride as (2,2)
512	2D Batch Normalization
512	Leaky ReLU with negative slope 0.2
512-1	2D transposed Convolutional operator with kernel size as (4,4), stride as (1,1)

1.2 Parameter setting

The purpose of the GAN network is to learn a parametric family of distribution closest to our data. By measuring the "closedness", we introduce the *Earth Mover distance* or *Wasserstein-1*:

$$W(P_r, P_g) = \inf E_{(x,y)\sim\gamma}[\|x - y\|]$$

In the training phase, we adopted tractable variants of this formulation such as differentiation approximation and minimize with regards to generator and discriminator both. More precisely, for a certain number of iterations, here set as n_critic, we train the discriminator to some extent. And then we train the generator to generate more fake examples. Then we finished an epoch. Here in the experiment, during the early stage of discriminator training, the $n_{critic} = 100$, which is high to encourage exploration compared to later $n_{critic} = 5$. The total number of epoch is 25, which is moderate considering the training time. The batch size is set at 64.
Now we examine our optimization technique. By the words of the author, traditional methods such as SGD and SGD-momentum performed poorly on this subject, by the nonstationarity of the loss for the discriminator (Arjovsky et al., 2017). Hence they choose RMSprop, which is known to perform robustly even on very non-stationary problems (Mnih et al., 2016).
For optimization algorithms, the main parameter at hand is the learning rate. So we choose to tune the learning rate to see the influence of such and also check if the claim of low learning rate being good is true. The range for such learning rate is $[0.00001, 0.0001]$ with 0.00001 as stepsize and the momentum coefficient for SGD with momentum is 0.9.

1.3 Datasets

We adopted SGD(Nemirovski et al., 2009), SGD with Momentum, Adam (Kingma and Ba, 2014), Adagrad (Duchi et al., 2011), RMSprop (Tieleman and Hinton, 2012) on several datasets, such as

- MNIST (LeCun et al., 2010): the human hand-written digits recognition dataset
- Cifar10 (Krizhevsky and Hinton, 2009): the well-known 10-class image classification dataset by Alex Krizhevsky, Geoffrey Hinton.
- Cifar100: Same as above with 100 class

2. Experimental results

2.1 Similarity between Real image sample and Fake image sample

The following observation is based on observing the fake sample generated at $500, 1000, 1500, ..., 3000$ iterations compared with a real image sampled from the data. Usually different learning rate would not show too much difference in this phase.

SGD Observation: On Dataset *Cifar10*, All fake examples fail to capture the distribution. At any iterations, the comparison shows like the 1. SGD does not seem to help with learning the desired distribution.
Cifar100 is practically the same. And on MNIST, the result is also consistent with previous observation, which is not learning distribution. See Figure.3

SGD with Momentum Observation: Very similar to the case of SGD. All fake examples fail to present any distribution, on *Cifar10* and *Cifar100* alike. See Figure.2. Same with MNIST. To avoid too many similar pictures, the failed results will only have one as an example for all.

Adam Observation: The result of Adam shows some promise. Fake example captured some of the feature of real data, as shown in Figure.4. And overall the training shows increased image quality. This trend is same on *Cifar10*,*Cifar100*, and *MNIST*, which can be proven by Figure.5. And by my opinion, the quality gets better when learning rate is larger as in $[0.00001, 0.0001]$. The effect is more relevant on Cifar dataset than MNIST, probably because Cifar is far more complex and comprehensive.

AdaGrad Observation: The case of AdaGrad is similar to that of SGD and SGD momentum, being that it fails to capture the distribution.
MNIST is the same with before, as in failing to capture distribution.

RMSprop Observation: The result of RMSprop shows promising effect. Figure.6 of *Cifar10*($\mathrm{lr} = 0.4$) shows that the last fake example has shown considerable features of the data distribution. And the whole training process demonstrates increase of learned image quality.
The observation on *Cifar100* is similar. And on *MNIST* as well

Figure 1: SGD on Cifar10, Left:last fake, Right:Real

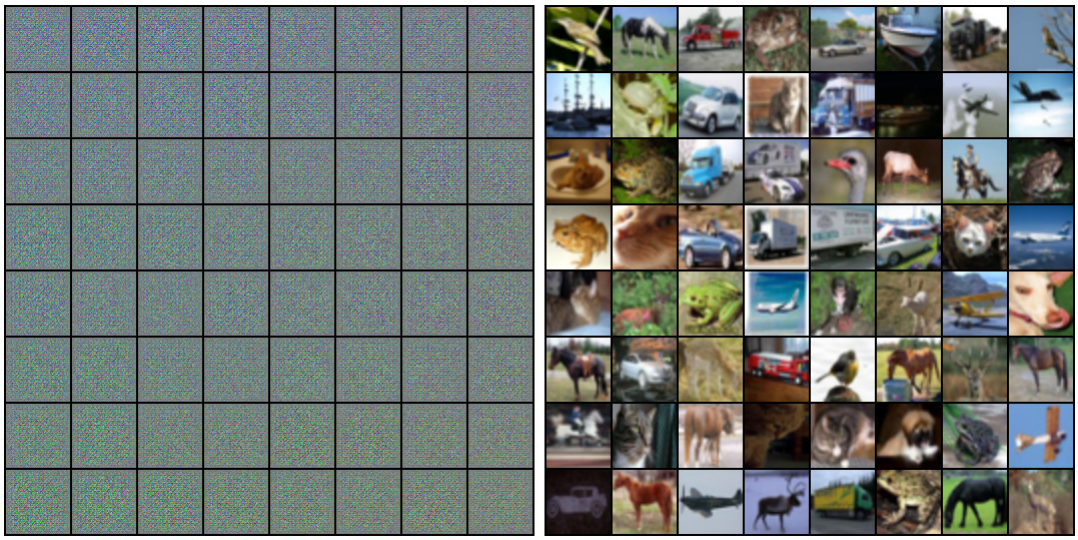

Figure 2: SGD with momentum on Cifar10, Left:last fake, Right:Real

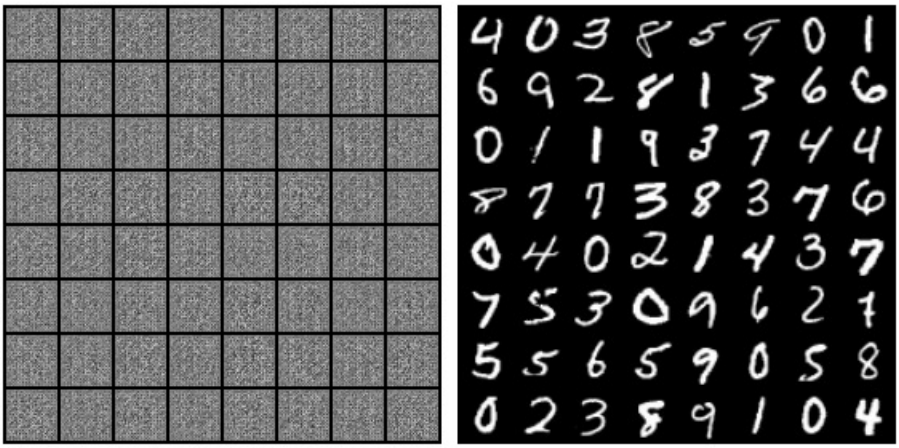

Figure 3: SGD on MNIST

Figure 4: Adam on MNIST

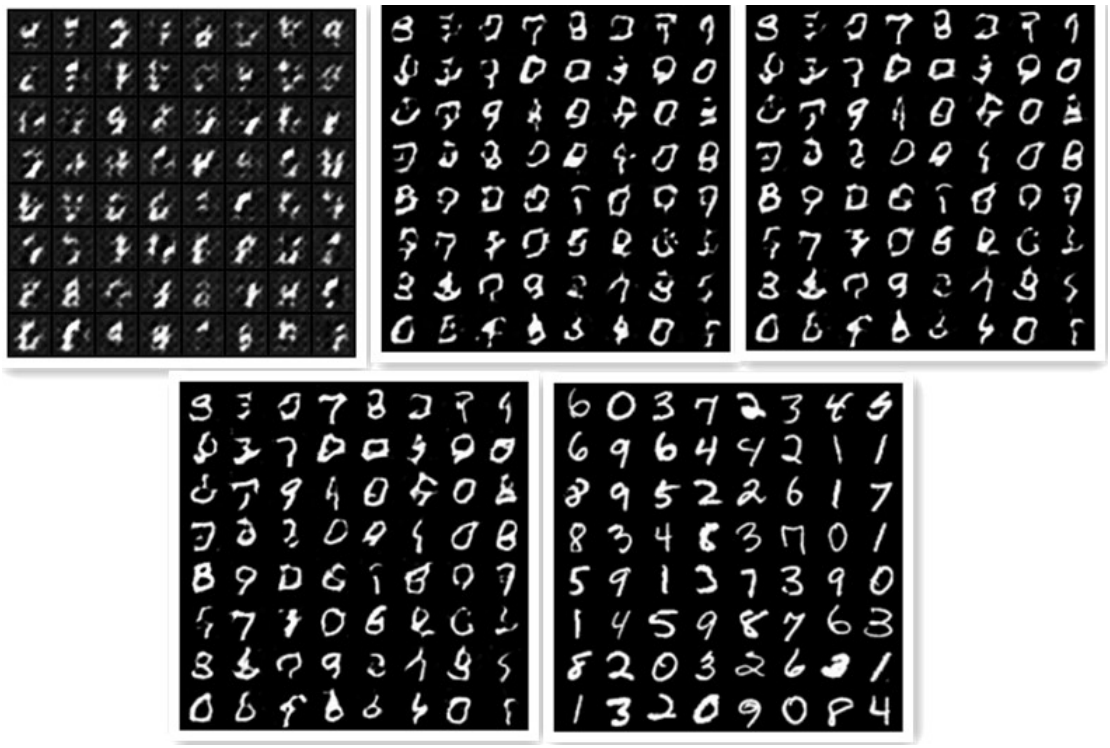

Figure 5: Adam on MNIST from left to right epoch increases, bottom right:REAL

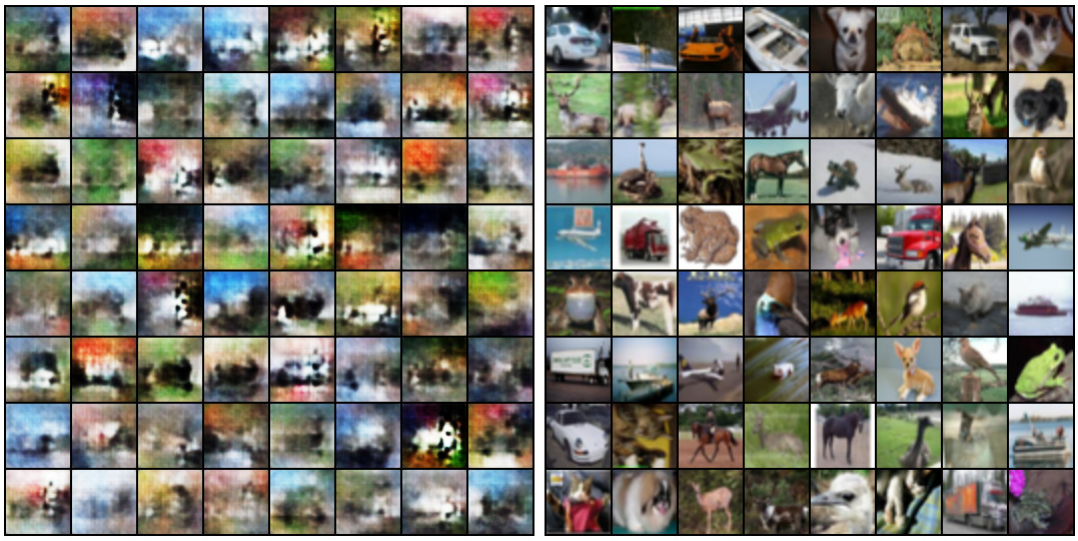

Figure 6: RMSprop on Cifar10 Left: Last Fake, Right: Real

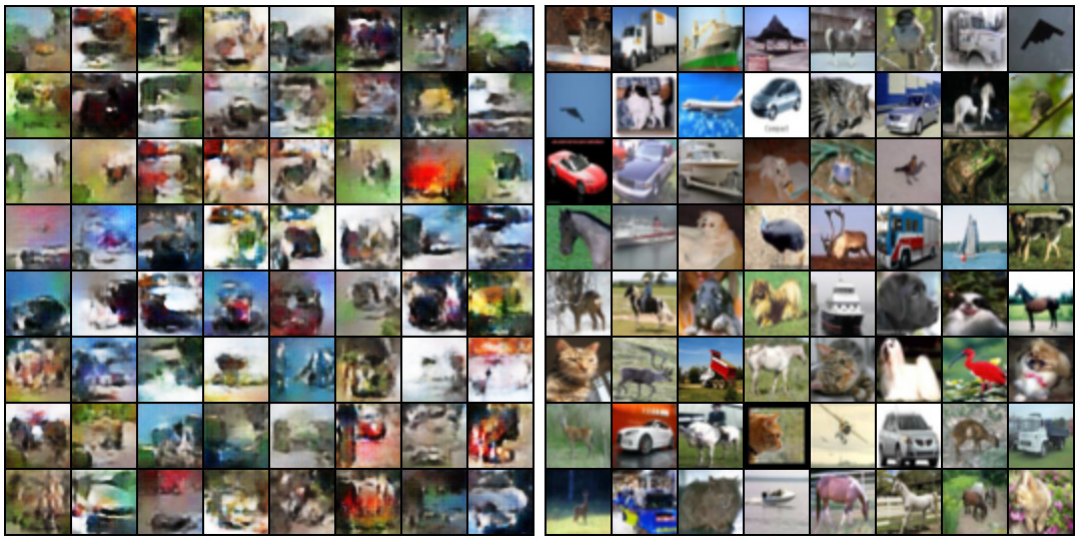

Figure 7: Adam on Cifar10 Left: Last Fake, Right: Real

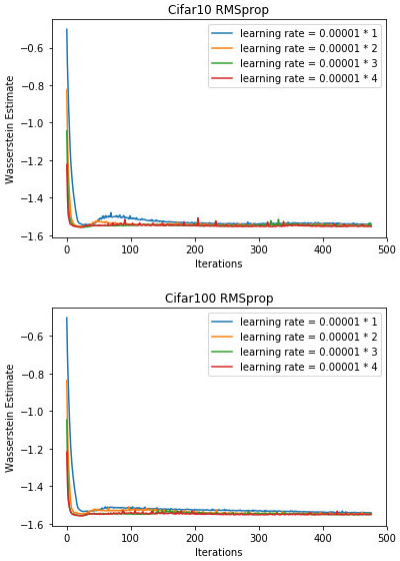

Figure 8: W-estimate on Cifar10 and Cifar100 with RMSprop

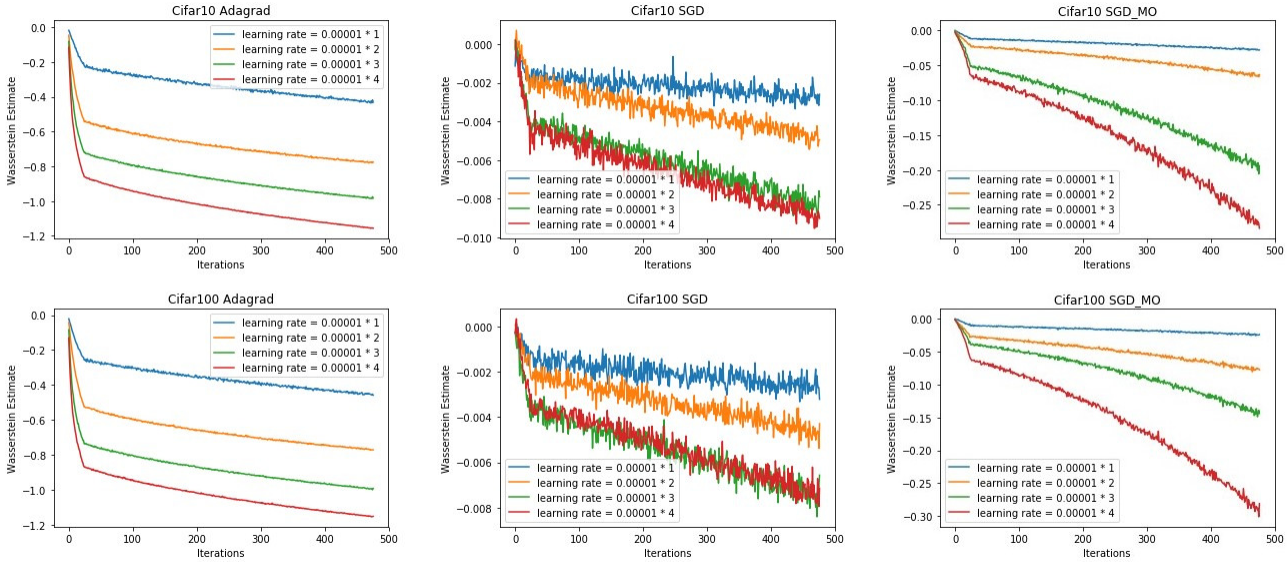

Figure 9: W-estimate on Cifar10 and Cifar100 with failed method(SGD, SGD-Momentum, Adagrad) in different learning rate(top:1 to 5,bottom:1 to 10 for comparison)

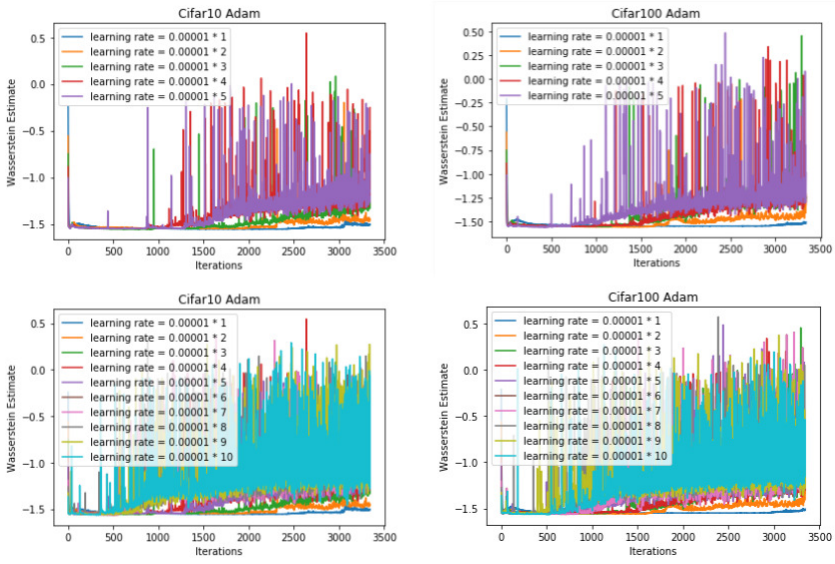

Figure 10: W-estimate on Cifar10 and Cifar100 with Adam in different learning rate(top:1 to 5,bottom:1 to 10 for comparison)

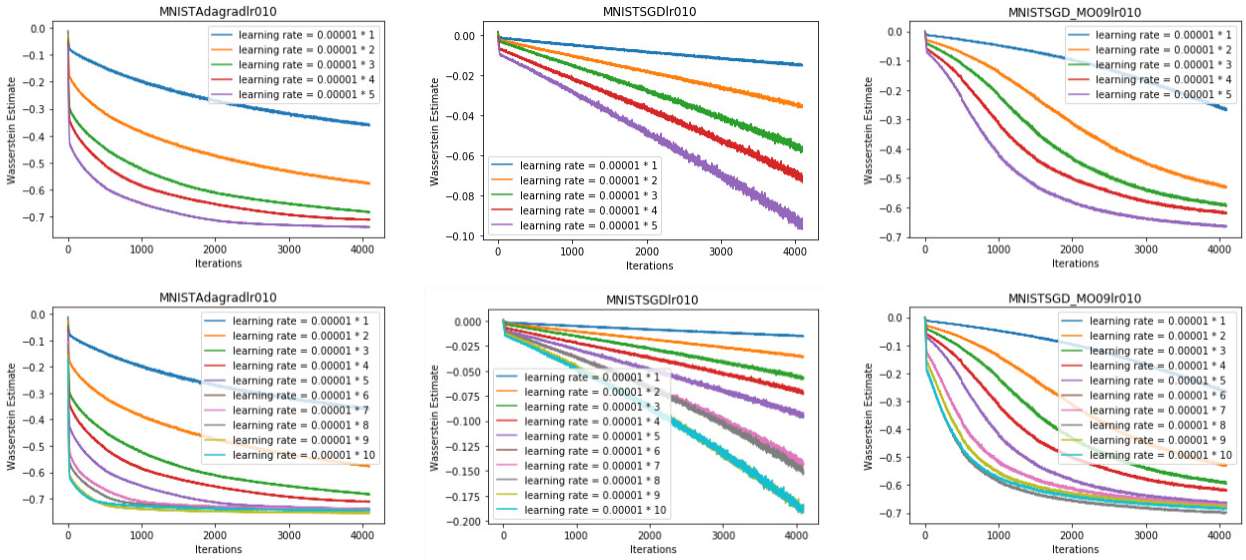

Figure 11: W-estimate on MNIST with failed method(SGD, SGD-Momentum, Adagrad) in different learning rate(top:1 to 5,bottom:1 to 10 for comparison)

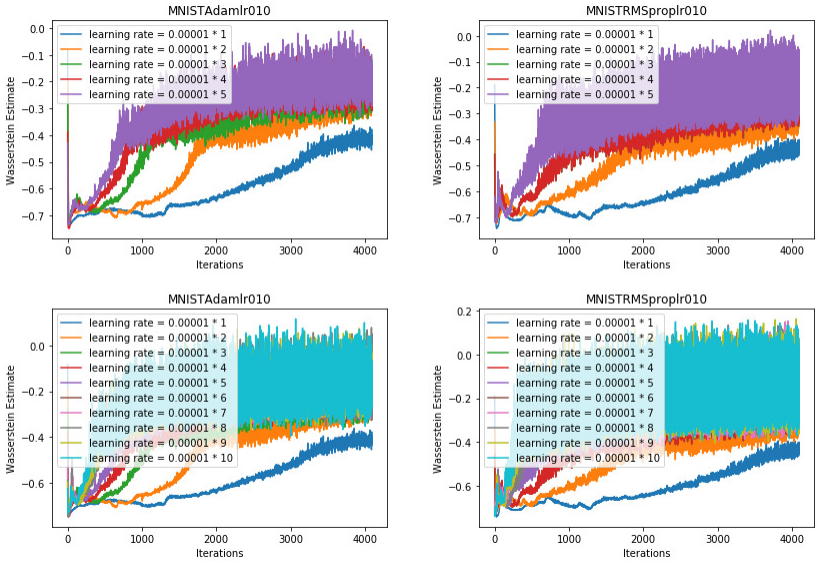

Figure 12: W-estimate on MNIST with Adam and RMSprop in different learning rate(top:1 to 5,bottom:1 to 10 for comparison)

Figure 13: W-estimate on MNIST with Adam and RMSprop in different learning rate(top:Adam, bottom:RMSprop,Left: Start Fake sample, Middle image: fake sample after about half of the epochs, Right: Real)

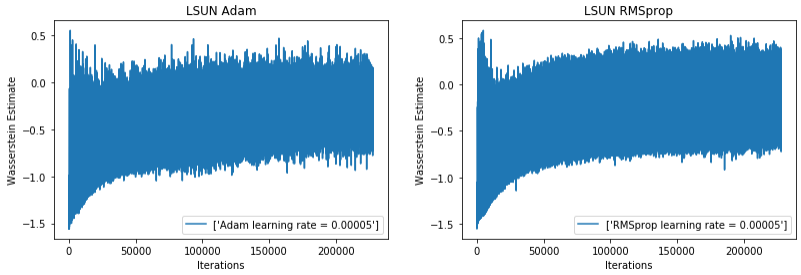

Figure 14: W-estimate on LSUN-bedroom with Adam and RMSprop

2.2 Meaningful Convergence

The author claim that one of the major benefit of W-GAN is that it has a meaningful, convergent-like loss metric. He claim that the W-estimate, if calculated(Loss of Discriminator minus Loss of Generator) can be a flag for better sample quality. As we can see from Figure.9 and Figure.10, the failed methods show a decrease of corresponding value, whereas the relatively successful method shows increasing although highly volatile. This pattern appears again in Figure.11 and Figure.12 in MNIST dataset. I have also shown the influence of learning rate by showing the first 5 and then the full 10 tuning parameters. We can observe that generally larger rate cause larger oscillation in later stages, but that is not necessarily a bad thing. As we can see from Figure.13 and Figure.14, which is a reproduction of author's result. The result on this large dataset LSUN-bedroom(43 GB) is quite good on appearance, but the underlying W-estimate is quite volatile. The increasing nature of the successful method's W-estimate convince us that it has inner relationship with the quality of the learning. And it also can help greatly in tuning and debugging the generative adversial network, which is known to be difficult to train.

3. Future work

For now, the code of WGAN training is written in pytorch 0.3.0 and does not fit in the framework of newly released pytorch 0.4.0. Translating it into the newer version and maintain its compatibility would be non-trivial to say the least. But the benefit of new version would come along as we can test out more optimization method such as AMSgrad and so on. Also, to further study the property that it has shown, in terms of the guidance for tuning and better sample quality, more efforts must be put into the qualitative analysis instead of empirical observation.

References

Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein gan. *arXiv preprint arXiv:1701.07875*, 2017.

John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul): 2121–2159, 2011.

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

Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. 2009.

Yann LeCun, Corinna Cortes, and CJ Burges. Mnist handwritten digit database. *AT&T Labs [Online]. Available: http://yann. lecun. com/exdb/mnist*, 2, 2010.

Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for

deep reinforcement learning. In *International Conference on Machine Learning*, pages 1928–1937, 2016.

Arkadi Nemirovski, Anatoli Juditsky, Guanghui Lan, and Alexander Shapiro. Robust stochastic approximation approach to stochastic programming. *SIAM Journal on optimization*, 19(4):1574–1609, 2009.

Tijmen Tieleman and Geoffrey Hinton. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2):26–31, 2012.