Conditional Wasserstein Generative Adversarial Networks With Conditional Batch Normilization

Lin, Han-Hsin
National Chiao Tung University
EECS Undergraduate Honors Program
achilles92434.eecs04@g2.nctu.edu.tw

***Abstract*—Generative Adversarial Networks (GANs) have shown to be powerful generative models, but have training instability issues. A proposed improvement, Wasserstein GAN (WGAN), has significantly stabilized the training process for unsupervised tasks and supervised tasks under certain network architecture settings. This work searches the probability of WGAN working on popular settings, which is the architecture of Deep Convolutional Generative Adversarial Network (DCGAN) in conditional sense. The condition method used is alike Conditional Generative Adversarial Network (CGAN), and additionally uses Conditional Batch Normalization (CBN) to let the network perform better.**

Keywords—WGAN, conditional, BN

I. INTRODUCTION

Generative Adversarial Networks (GANs) [1] is an framework for training generative models that overcome the problem of capturing complex data distributions by approximating the probability distribution function. GANs are trained by a game between two networks: the generator that tries to cheat the discriminator by generating data from a noise source and the discriminator that distinguish between true data and generated data.

GANs can generate realistic results, but are usually hard to train. Therefore a lot of effort was made to make the training process stable and reliable, for example, Deep Convolutional Generative Adversarial Network (DCGAN) [10] is an optimal network structure chosen from various parameter setting. In all the different attempts, Wasserstein GAN (WGAN) [5] is worth mentioning as it makes loss numbers meaningful and stabilizes the training. Later, a follow up model WGAN-GP [4] fixed some problems that occurred, therefore let WGAN-GP become one of the most used models as a first attempt for new datasets because of its stability.

Although GANs are able to sample from probability distribution to generate new similar data, but vanilla GAN is only used for unsupervised training. Therefore a simple yet notable extension, Conditional GAN (CGAN) [3], was presented. In the rest of the paper we will try to implement WGAN in conditional sense with CGAN and generate acceptable data.

II. BACKGROUND

A. Generative adversarial networks

The GAN framework estimates generative models via training two models simultaneously with an adversarial process. The generator G generates data from a source of noise, while the discriminator D tries to distinguish between true data samples and generated samples.

Formally, the game between the two models is the minimax objective:

$$\min_G \max_D \mathop{\mathrm{E}}_{x\sim P_{data}(x)}[\log(D(x))] + \mathop{\mathrm{E}}_{z\sim P_z(z)}[(1-D(G(z)))]$$

Where $P_z(z)$ is a defined input noise variable, $\mathrm{G}(\mathrm{z})$ is a mapping to data space, and $\mathrm{D}(\mathrm{x})$ outputs a single scalar which represents the probability that x is from true data instead generated by G.

B. WGAN and WGAN-GP

Knowing that vanilla GAN is not stable, and using JS divergence in training process might fail in simple circumstances and the numbers don't have obvious meanings, WGAN was proposed to solve the problems by substituting JS divergence with EM distance. The EM distance is defined as:

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

In simple words, it represents the amount of work needed to do to map the probability distribution from x to y, where we define $\prod(P_r, P_g)$ to be the set of joint distributions with marginal P_r and $\mathrm{P_g}$. Though it is ideal to simply calculate and minimize the EM distance, it cannot be obtained in practical training process. Instead, they use an approximation:

$$E_{x\sim P_r}[\mathrm{D(x)}] - E_{G(z)\sim P_g}[D(G(z))]$$

Where D must be a set of 1-Lipschitz function in order to let this approximation hold. To enforce the Lipschitz constraint, they clip the weights in the network within a compact space $[-\mathrm{c}, \mathrm{c}]$, but this approach is brutal and has some sequela.

WGAN-GP is a much more "softer" way of constraining D to satisfy the Lipschitz constraint by giving a penalty to the calculated loss, thus constraining the gradients. This gives the name to this approach, which is *Gradient Penalty*.

C. Conditional Batch Normalization

Batch Normalization (BN) [13] is a widely used technique that was designed to accelerate training of neural networks by reducing internal covariate shift. Defining a mini-batch $\mathrm{B} = \{\mathrm{F_i}, ., ., .\}_{i=1}^{N}$ of N examples, where F corresponds to input feature maps, BN normalizes at training time as follows:

$$\mathrm{BN}(\mathrm{F_{i,c,h,w}}|\gamma_c, \beta_c) = \frac{\gamma_c(F_{i,c,w,h} - E_b[F_{.,c,.,.}])}{\sqrt{Var_B[F_{.,c,.,.}] + \epsilon}} + \beta_c$$

Where γ and β are trainable scalars that maintain representational power of network, and ϵ is a constant damping factor for numerical stability.

Conditional Batch Normalization (CBN) [7,8] learns new parameters $\gamma_{\mathrm{i,c}}, \beta_{i,c}$ according to the label of the input $\mathrm{x_i}$, thus can control the activations based on $\mathrm{x_i}$.

Combined with ReLU activations, CBN allows to control the use of feature maps by scaling, shutting them off, and other operations. Therefore able to share the common feature maps across different labels, and can ignore or use certain feature maps.

III. Experiments

The architecture is trained on three datasets, MNIST, fashion-MNIST, and CIFAR-10. Each dataset are trained on different number of epochs but are identical in the same dataset. Lots of network architecture parameters were tried and the following work only shows the one that works the best..

A. Architecture Details

The architecture is mostly the same as DCGAN with minor changes. Below is the architecture for MNIST datasets, other datasets only differ in the filter count and the channel count, details can be found in the code.

Generator	**Discriminator**
Dense layer (4*4*256 units) BN, ReLU	Conv2d(filter 64, kernel 5x5, stride 2x2) Leaky ReLU
Transpose conv2d (filter 128, kernel 5x5, stride 2x2) BN, ReLU	Conv2d(filter 32, kernel 5x5, stride 2x2) Leaky ReLU
Transpose conv2d (filter 64, kernel 5x5, stride 2x2) BN, ReLU	Conv2d(filter 16, kernel 5x5, stride 2x2) Leaky ReLU
Transpose conv2d (filter 1,kernel 5x5, stride 2x2) Tanh	Conv2d(filter 64, kernel 4x4, stride 1x1) Leaky ReLU Dense (1 unit)

B. MNIST

MNIST is a widely used grayscale image dataset, consisted of ten number digits with different handwritten styles, the generated images after training are shown in Fig.1.

C. Fashion-MNIST

Fashion-MNIST is a grayscale image dataset with ten different classes consisted of clothing, shoes, and others. Alike MNIST, it has 60,000 training set images and 10,000 test set images. The generated images are shown in Fig. 2.

Notice that the features are more significant in the network that uses CBN, for example, the heel of the boots and the handle of the bags are identifiable.

D. CIFAR-10

CIFAR-10 is a widely used color image dataset with 50000 training images and 10000 test images, consisted of ten different classes like ship, frog, and others. The generated images are shown in Fig.3, and the result of training with normal batch normalization is omitted since it performs only worse.

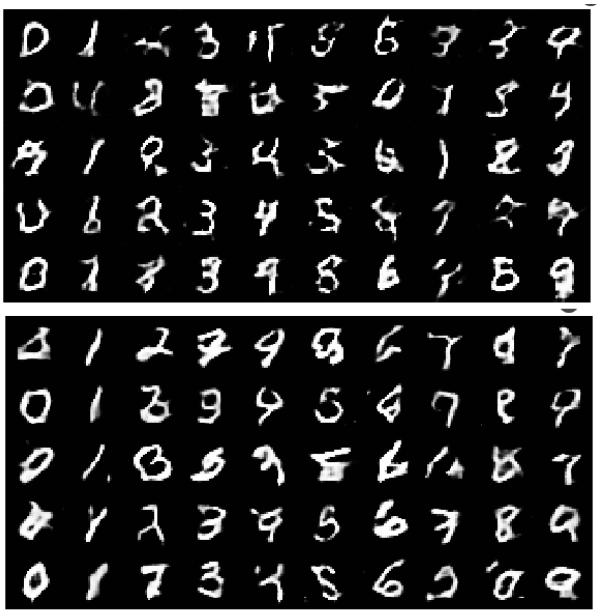

Fig.1 Generated images from trained WCGAN network. Each column corresponds to one number. Top: normal BN. Bottom: CBN.

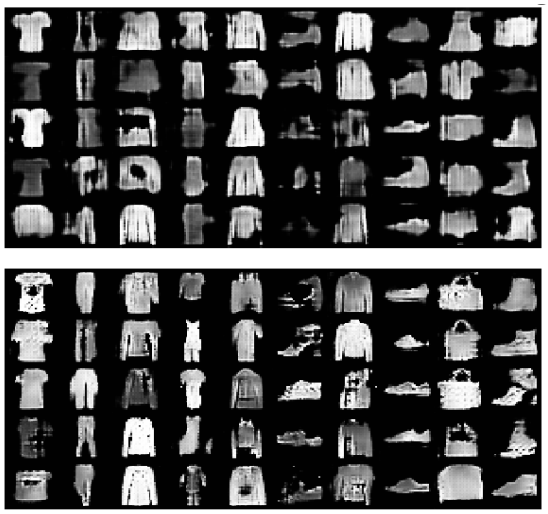

Fig. 2 Generated images from trained WCGAN network. Each column corresponds to one class. Top: normal BN. Bottom: CBN.

Fig. 3 Generated CIFAR-10 images with CBN

IV. Discussion

From the MNIST and fashion-MNIST datasets we can observe that conditional batch normalization does enhance the performance of network, especially the details like the handles if the bag class, and doesn't mix up digits that much like the image of non-conditional batch normalization. In the CIFAR-10 dataset, normal batch normalization generates unrecognizable images while CBN can make the images from different classes distinguishable. To conclude, conditional batch normalization does work as intended but still doesn't provide satisfying results.

The main purpose of this paper is to try whether the simple CGAN method still works or not in WGAN-GP settings since the original paper of WGAN-GP used residual blocks from ResNet [12] in the architecture and ACGAN [11] to condition CIFAR-10 images, but never mentioned anything else. From the results of my WCGAN implementation, I can conclude that the DCGAN structure may not work well even if applying CBN.

References

[1] M.Goodfellow, I., Pouget-Abadie, J., & Mirza, "Generative Adversarial Networks," 2014.

[2] I.Goodfellow, "NIPS 2016 Tutorial: Generative Adversarial Networks," 2016.

[3] M.Mirza andS.Osindero, "Conditional Generative Adversarial Nets," 2014.

[4] I.Gulrajani, F.Ahmed, M.Arjovsky, V.Dumoulin, andA.Courville, "Improved Training of Wasserstein GANs," 2017.

[5] M.Arjovsky, S.Chintala, andL.Bottou, "Wasserstein GAN," 2017.

[6] M.Arjovsky, S.Chintala, andL.Bottou, "Conditional Wasserstein generative adversarial networks," Icml, pp. 214–223, 2017.

[7] E.Perez, H.deVries, F.Strub, V.Dumoulin, andA.Courville, "Learning Visual Reasoning Without Strong Priors," Jul.2017.

[8] H.deVries, F.Strub, J.Mary, H.Larochelle, O.Pietquin, andA.Courville, "Modulating early visual processing by language," Jul.2017.

[9] V.Dumoulin, F.Visin, andG. E. P.Box, "A guide to convolution arithmetic for deep learning," 2018.

[10] A.Radford, L.Metz, andS.Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," in *International Conference on Learning Representations (ICLR)*, 2015.

[11] A.Odena, C.Olah, andJ.Shlens, "Conditional Image Synthesis With Auxiliary Classifier GANs," Oct.2016.

[12] K.He, X.Zhang, S.Ren, andJ.Sun, "Deep Residual Learning for Image Recognition," Dec.2015.

[13] S.Ioffe andC.Szegedy, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," Feb.2015.