CV Assignment 7: GANs

Chao Wang

Due Date: June 18, 2017

College of Information Science and Engineering, Ocean University of China, Qingdao, China
E-mail: chaowangplus@gmail.com

1 Introduction

In this assignment. You will implement Generative Adversarial Networks (GANs), and apply them to image generation on MNIST, CIFAR10 and LFW datasets. You will also explore models addressed in CV class to train GAN in a stable way and avoid graident unstable and mode missing problems. Finally, you will get a generative adversarial network to generate images that look like a training dataset!

The goals of this assignment are as follows:

- Understand how to train and implement a generative adversarial netwoks (GANs).
- Understand how to improve the training of GANs.
- Have a try on the most popular generative model.

2 Setup

You can work on the assignment in some popular deep learning frameworks. I highly recommend TensorFlow and PyTorch. See more details here: TensorFlow: http://www.tensorflow.org; PyTorch: http://www.pytorch.org.

2.1 Datasets

In this assignment, you will need three datasets for training your GAN models:

- MNIST (http://yann.lecun.com/exdb/mnist/).
- CIFAR10 (http://www.cs.toronto.edu/~kriz/cifar.html).
- LFW (http://vis-www.cs.umass.edu/lfw/).

2.2 Model zoo

Some models you can choose:

- GAN [1].
- DCGAN [2]
- WGAN [3].
- Improved WGAN [4].

2.3 Notes

- You can find all of the models on github.
- When your model seems not convergence. Reference to: https://github.com/soumith/ganhacks.
- For conditional generation: CGAN [5], InfoGAN [6].
- MNIST and CIFAR10 are embedded in Tensorflow and PyTorch. Here is an example of using CIFAR10 in PyTorch.

```
dataset = datasets.CIFAR10(root='data/cifar10', download=True,
                           transform=transforms.Compose([
                               transforms.Scale(32),
                               transforms.ToTensor(),
                               transforms.Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5)),
                           ])
                                    )

dataloader = torch.utils.data.DataLoader(dataset, batch_size=64,
                                         shuffle=True, num_workers=int(2))
```

3 Experiment

This part includes some details of the experiment:

1. Training at least three models (WGAN must be selected.) on all the datasets to generate images like a training dataset. Plot the training loss curve. Compare and analysis the gradient stability problem of your models. Compare the image quality of different models.

2. Generate **MNIST** digits conditioned on class label through one of your model and plot the training loss curve to analysis the efficiency of add label information to GAN. You should generate samples like Fig. 1.

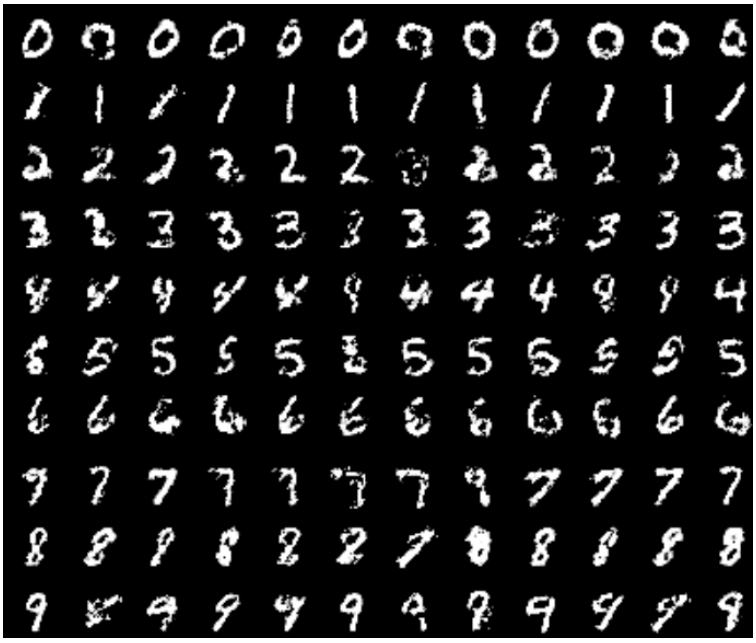

Fig. 1: Generated MNIST digits, each row conditioned on one label.

4 Submitting your work

At last you need to submit your work to *ouceecv@163.com* in a zip file called: **YourName_Assignment7.zip**. And it should include these parts:

1. A report with your analysis about your experiments.

2. Your result.

3. Code.

References

[1] Ian Goodfellow, Jean Pougetabadie, Mehdi Mirza, Bing Xu, David Wardefarley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *NIPS*, 2014.

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

[3] Martin Arjovsky, Soumith Chintala, and Lon Bottou. Wasserstein gan. *arXiv preprint*, 2017.

[4] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron Courville. Improved training of wasserstein gans. *arXiv preprint*, 2017.

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

[6] Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *NIPS*, 2016.