cycleGAN project report

Kaijie Wang&Hong Guan

1.outlines(contribution):

Hong Guan: cyclegan.py and some network dataset revise.

Kaijie Wang: network.py dataset.py and do the training get the output

2.Introduction:

Image-to-Image translation is a visual and image problem. Its goal is to use paired images as a training set, and let the machine learn the mapping from the input image to the output image. However, in many tasks, paired training data is not available.

We propose a method of learning from the source data domain X to the target data domain Y in the absence of paired data. Our goal is to use an adversarial loss function to learn the mapping G: $X \rightarrow Y$, making it difficult for the discriminator to distinguish between picture G (X) and picture Y. Because this kind of mapping is greatly restricted, we add an opposite mapping F: $Y \rightarrow X$ to the mapping G, making them paired, and adding a cycle consistency loss function (cycle consistency loss) to ensure F (G (X)) $\approx$ X (or vice versa).

3. Related work

The goal we want to achieve here is the transformation of MNIST TO SVHN, so we borrowed the working principle of cycleGAN source code on the Internet. We first

understood the role of each part in the source code. The simple cycleGAN needs the training part, the neural network part is the generator discriminator, the Loss function part and the data.

However, there is no one we need in the traditional training set, so we have to create our own training set

4. Method

Tools: pytroch&colab&tensorflow

Let cyclegan.py and network.py and dataset.py to the colab and use instruction

```
[ ]  !python  cyclegan_model.py  --mode=train
```

When finish training use instruction

```
[ ]  !python  cyclegan_model.py  --mode=test
```

Then if you want to see the image you can use

```
[ ]  from  IPython.display  import  Image
     Image(filename='output/A/0001.png')
```

or

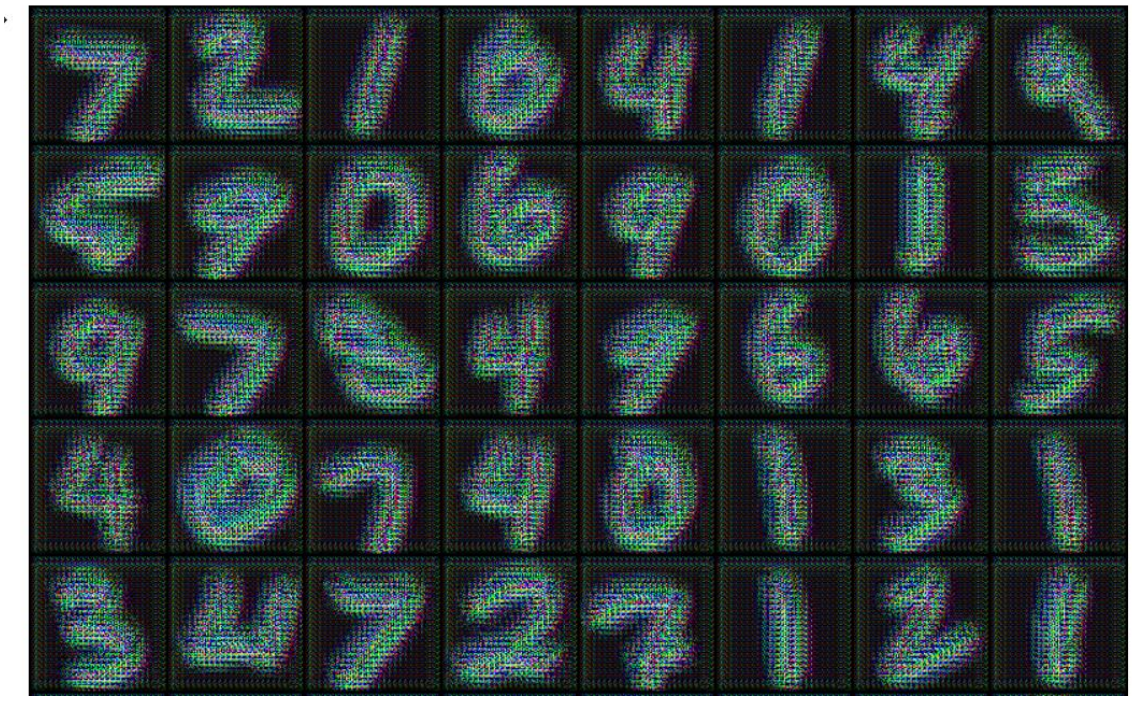
```
     from  IPython.display  import  Image
     Image(filename='output/B/0001.png')
```

The output A is fake MNIST output B is fake SVHN

5. Experiments

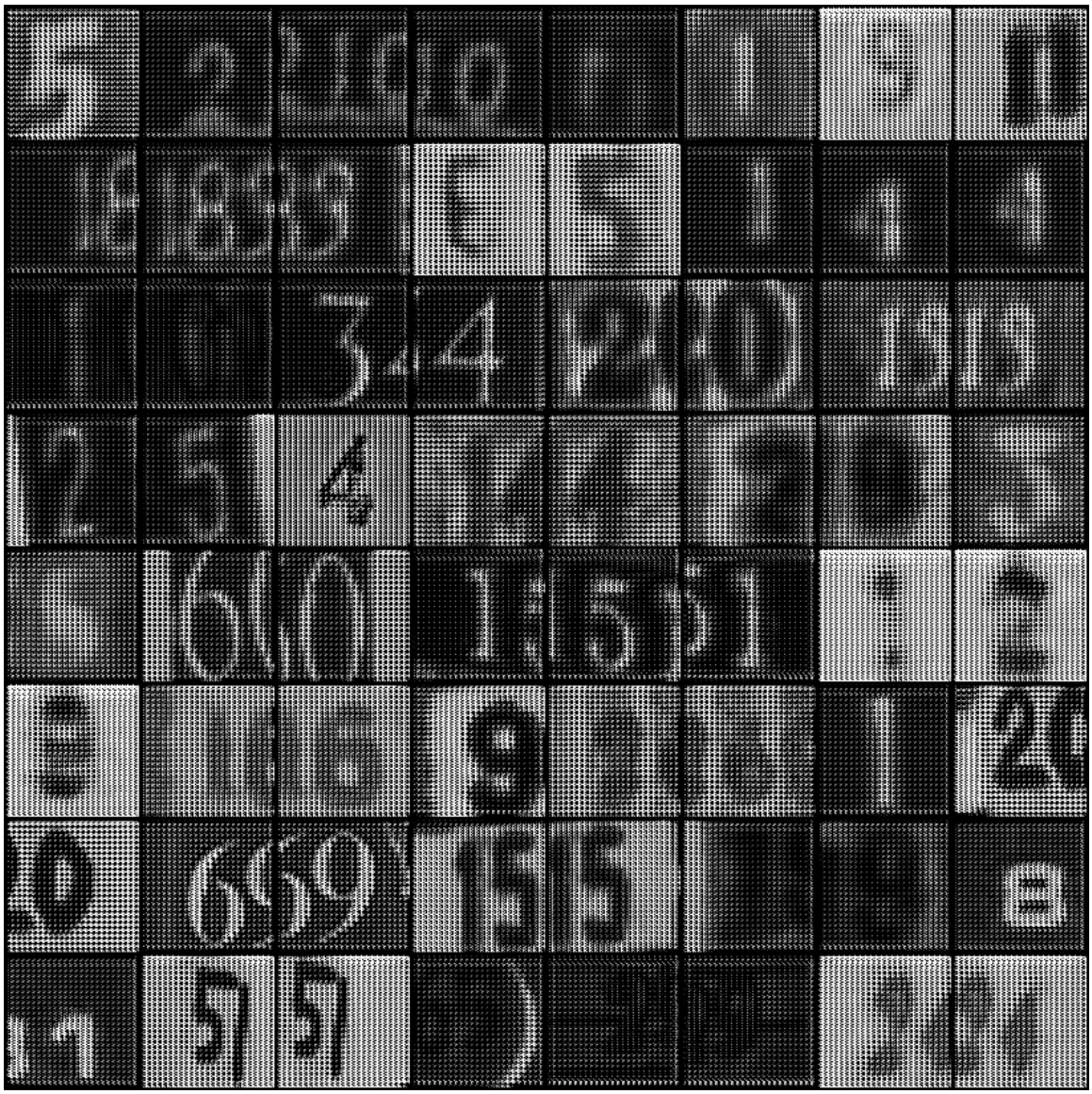
We made a lot of comparisons during the final run. We set up an enhancer called buffer in the code. As a comparison, we made several control groups. (first image is fake MNIST, second is fake SVHN)

Group 1:buffer size 32, train epoch 2, load size 100, batch size 64:

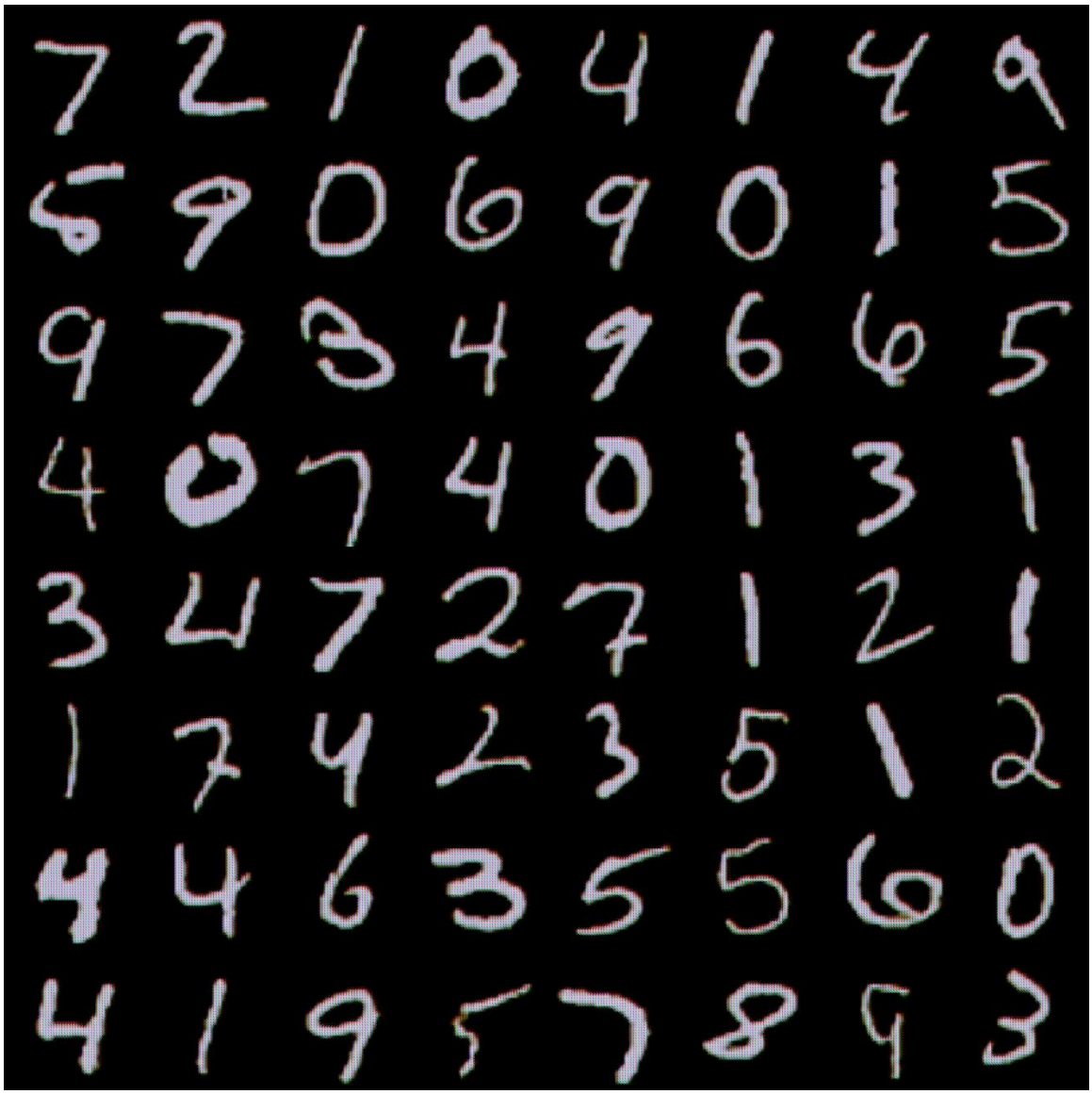


Groups 2: buffer size 64, train epoch 2, load size 100, batch size 64:

Groups 3: buffer size 64, train epoch 20, load size 100, batch size 64:



Groups 4: buffer size 240, train epoch 20, load size 200, batch size 64:



6.Conclusion:

We plot the loss vs. steps graph, and find that it converges quickly within 1 epoch, so it is useless to train many steps. So, we think that GAN cannot just look at the Loss function.

The key parameters are buffer size, load size and batch size.

Reference

[1] “Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks,” CycleGAN Project Page. [Online]. Available: https://junyanz.github.io/CycleGAN/. [Accessed: 29-Apr-2020].

[2] Yunjey, “yunjey/mnist-svhn-transfer,” GitHub, 27-May-2017. [Online]. Available: https://github.com/yunjey/mnist-svhn-transfer. [Accessed: 29-Apr-2020].