CPSC 8430 -HW3

Fangjian Li

April 2021

The codes are available at:
`https://github.com/FangjianLi/Deep-learning-HW3`

1 Generative Adversarial Networks (GANs)

The generative adversarial networks (GANs) Goodfellow et al. (2014) have been used to generate the new data points from different databases in the literature. In this project, three different forms of GANS, i.e., DCGAN, WGAN, and ACGAN, are applied to generate realistic images based on the cifar-10 image database. The cifar-10 (`https://www.cs.toronto.edu/~kriz/cifar.html`) contains 50000 training 32x32 RGB images with ten different classes which are airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck. The example images are shown in Fig. 1.

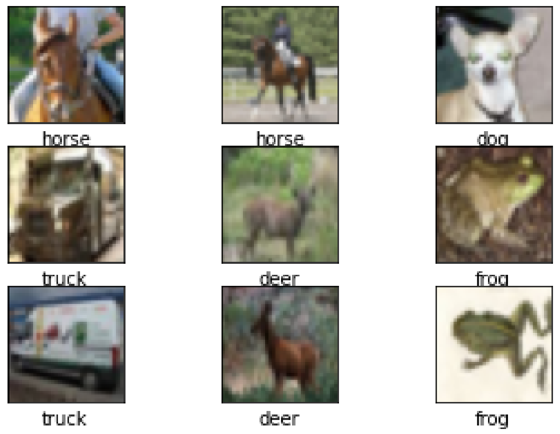


Figure 1: Examples of cifar-10 iamges

The structure of GANs is quite intuitive. The most basic GAN is composed of two components, i.e., generator (G) and discriminator (D). The generator takes the noises z as the input and generates the fake images. The discriminator takes the images (either fake $x \sim p_g$ or real $x \sim p_r$) as the input, and outputs the level of realism, i.e., 0 for totally fake and 1 for genuinely realistic. The structure of GAN is shown in Fig. 2.

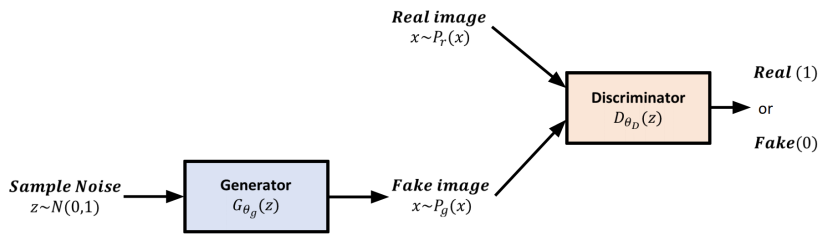


Figure 2: Diagram of the GAN

The generator and discriminator can be trained based on the following loss functions.

$$L_{\text{discrim}} = -E_{x\sim p_r}[\log(D(x))] - E_z[\log(1 - D(G(z)))] \quad (1)$$

$$L_{\text{gen}} = -E_z[\log(D(G(z)))] \quad (2)$$

2 Deep Convolutional Generative Adversarial Networks (DCGAN)

The deep convolutional generative adversarial networks use the convolutional neural network to build the generator and discriminator instead of the feedforward neural network adopted by the conventional GANs Radford et al. (2015). In this project, the DCGAN is first built to generate the realistic photos based on cifar-10 database. The detailed structures of generator and discriminator are shown in Fig. 3.

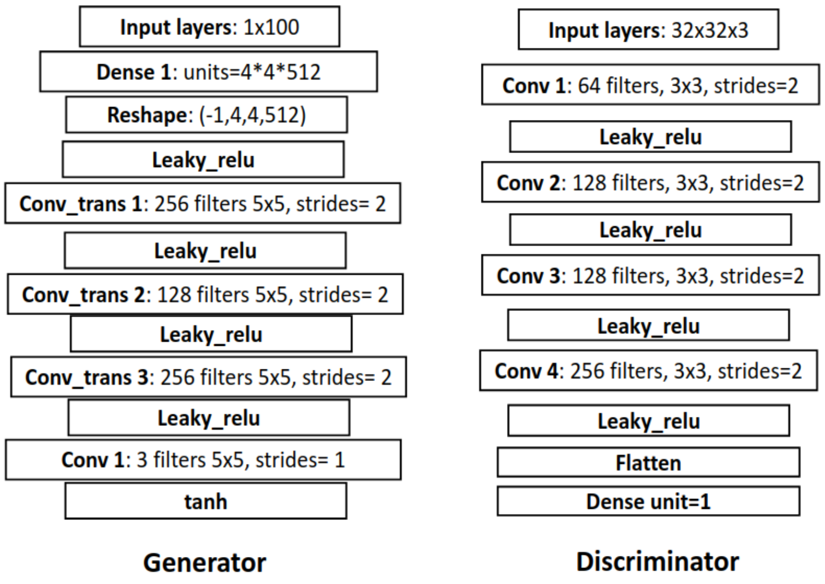

Input layers: 1x100
Dense 1: units=4*4*512
Reshape: (-1,4,4,512)
Leaky_relu
Conv_trans 1: 256 filters 5x5, strides= 2
Leaky_relu
Conv_trans 2: 128 filters 5x5, strides= 2
Leaky_relu
Conv_trans 3: 256 filters 5x5, strides= 2
Leaky_relu
Conv 1: 3 filters 5x5, strides= 1
tanh
Generator
Input layers: 32x32x3
Conv 1: 64 filters, 3x3, strides=2
Leaky_relu
Conv 2: 128 filters, 3x3, strides=2
Leaky_relu
Conv 3: 128 filters, 3x3, strides=2
Leaky_relu
Conv 4: 256 filters, 3x3, strides=2
Leaky_relu
Flatten
Dense unit=1
Discriminator

Figure 3: Structure of the DCGAN

The neural network is built in **Tensorflow**. The functions **tf.layers.conv2d** and **tf.layers.conv2d_transpose** are mainly used to build the model. The training parameters are shown in the Table. 1. The codes can be found in `https://github.com/FangjianLi/Deep-learning-HW3/tree/main/DCGAN` .

Table 1: DCGAN training configuration

Optimizer	Adam	**# of epochs**	200
Learning rate (gen)	2e-4	**Learning rate (discrim)**	2e-4
Batch size	100	**Kernel initializer**	truncated normal

The generated image examples are shown in the following figure.

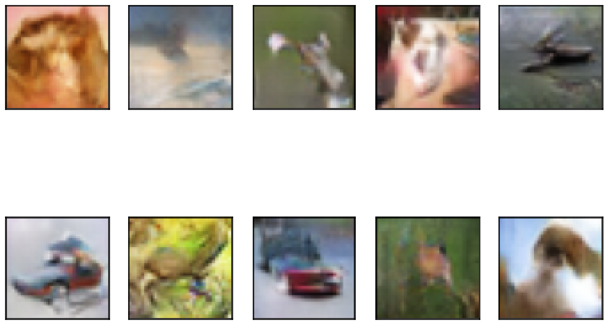


Figure 4: Example of DCGAN generated pics

The **FID score** of DCGAN is obtained as **42.9**

3 Wasserstein Generative Adversarial Networks (WGAN)

The conventional GAN loss function mentioned in Eqns. (1)-(2) can be proved to minimize the Jensen-Shanon (JS) divergence between generated image distribution and real image distribution. However, when there is no overlap between these two distributions, the JS divergence is always the same. This property is not good for the gradient-based network training. To address this problem, the Wasserstein distance is used to quantify the distribution difference Arjovsky et al. (2017). Via some derivations, the loss functions of the generator and discriminator become:

$$L_{\text{discrim}} = -E_{x\sim p_r}[D(x)] + E_z[D(G(z))] \tag{3}$$
$$L_{\text{gen}} = -E_z[D(G(z))] \tag{4}$$

In addition, the weights of the neural network are also clipped in order to satisfy the k-Lipschitz property.

In this project, the WGAN is built based on **Tensorflow**. The network structures are same as the DCGAN shown in the Section 2. The training configuration is shown in the Table 2. The codes can be found in `https://github.com/FangjianLi/Deep-learning-HW3/tree/main/WGAN`.

Table 2: WGAN training configuration

Optimizer	Adam	**# of epochs**	245
Learning rate (gen)	2e-4	**Learning rate (discrim)**	1e-4
Batch size	100	**Kernel initializer**	truncated normal

The generated image examples are shown in the following figure.

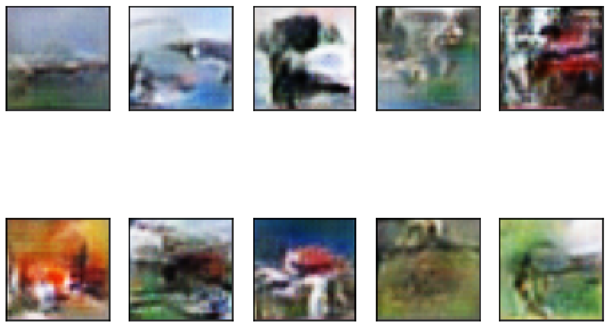


Figure 5: Example of WGAN generated pics

The **FID score** of WGAN is obtained as **82.5**

4 Auxiliary Classifier Generative Adversarial Networks (ACGAN)

On top of the structure of the conventional GAN discriminator, an classifier that can output the class of the pictures is added to improve the performance Odena et al. (2017). To achieve this structure, the generator takes the fake label, l, as the additional input. The discriminator still take the images as the input. Because of the additional classifier, the discriminator can also predict the class of the input image. The structure of the ACGAN is shown in Fig. 6.

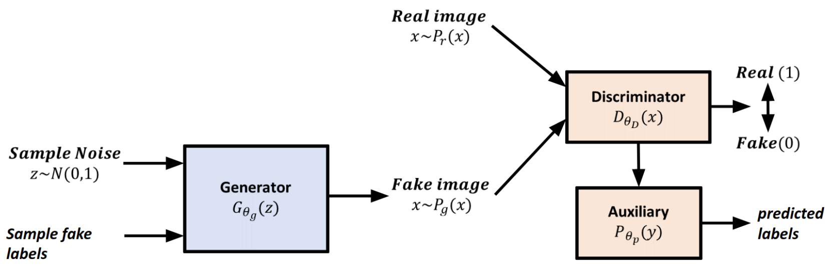
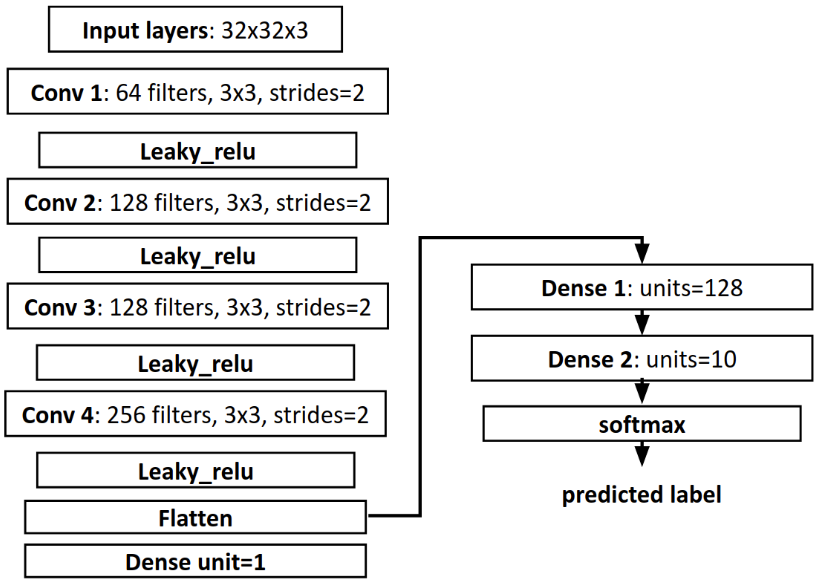


Figure 6: Diagram of the ACGAN

In this project, the ACGAN is built by using **Tensorflow**. The generator network structure is same as DCGAN. The structure of discriminator is shown in the following figure.



Input layers: 32x32x3
Conv 1: 64 filters, 3x3, strides=2
Leaky_relu
Conv 2: 128 filters, 3x3, strides=2
Leaky_relu
Conv 3: 128 filters, 3x3, strides=2
Leaky_relu
Conv 4: 256 filters, 3x3, strides=2
Leaky_relu
Flatten
Dense unit=1
Dense 1: units=128
Dense 2: units=10
softmax
predicted label

Figure 7: Structure of the ACGAN discriminator

Since new structure is introduced, the loss functions of generator and discriminator are also changed as follows.

$$L_{\text{discrim}} = -E_{x\sim p_r}[\log D(x)] - E_z[\log(1-D(G(z)))] - E_{x\sim p_r}[\log P(c|x)] - E_{z,l}[\log P(c|G(z,l))] \tag{5}$$

$$L_{\text{gen}} = -E_z[\log D(G(z))] - E_{z,l}[\log P(c|G(z,l))] \tag{6}$$

The training parameters are same as the DCGAN in Section 2. The codes can be found in `https://github.com/FangjianLi/Deep-learning-HW3/tree/main/ACGAN`. The generated image examples are shown in the following figure.

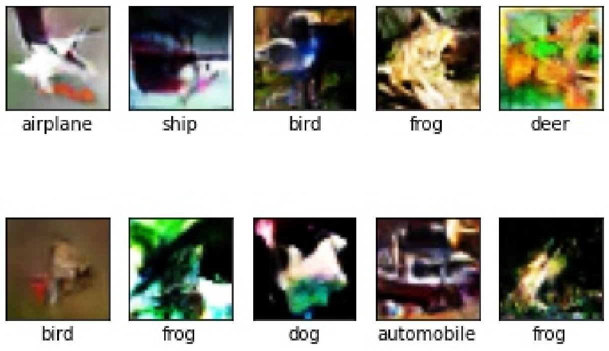


Figure 8: Example of ACGAN generated pics

The **FID score** of ACGAN is obtained as **45.9**

5 The comparison between DCGAN, WGAN, and ACGAN

In this project, three GANs, i.e., DCGAN, WGAN, and ACGAN, are built to generate the images based on the cifar-10 dataset. The **FID score** for DCGAN, WGAN, and ACGAN are **42.9**, **80.2**, and **45.9** respectively. As can be seen, based on the hyperparameters adopted in the above sections, the performance of ACGAN is similar to DCGAN. ACGAN can also output an image classifier. Based on the chosen hyperparameters, the performance of WGAN is worse than the other two methods. The generated pictures are less realistic. As the loss is quantified by the Wasserstein distance, the hyperparameters and structure of WGAN might need to be fine-tuned in order to achieve better performance. Correspondingly, the ten best pictures in this project should be the pics generated by DCGAN, as shown in Fig. 4.

References

Arjovsky, Martin, Soumith Chintala, and Léon Bottou (2017), “Wasserstein generative adversarial networks.” In *International conference on machine learning*, 214–223, PMLR.

Goodfellow, Ian J, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio (2014), “Generative adversarial networks.” *arXiv preprint arXiv:1406.2661*.

Odena, Augustus, Christopher Olah, and Jonathon Shlens (2017), “Conditional image synthesis with auxiliary classifier gans.” In *International conference on machine learning*, 2642–2651, PMLR.

Radford, Alec, Luke Metz, and Soumith Chintala (2015), “Unsupervised representation learning with deep convolutional generative adversarial networks.” *arXiv preprint arXiv:1511.06434*.