Facial Expression Generation Using DCGAN

1 Introduction

Now more and more people like to use their mobile phones to record the beautiful moments around them, and they are also good at taking pictures of small accidents in life. With the increase in the number of photos taken, people's requirements for photo edition are also slowly increasing. Sometimes these small accidents can only be captured accidentally, but to achieve better results, people need to edit these photos. For example, editing people's facial expressions on photos. And this function is not so popular for current editing software. Therefore, we can train a machine learning model to generate human facial expressions through the existing dataset of a large number of human expressions. After searching for good methods to process human facial image, it is recommended by combining deep convolutional neural network (CNN) and Generative Adversarial Networks (GAN) to form a high performance machine learning model – DCGAN (Goodfellow et al., 2014; Radford et al., 2016). The performance of the face generation using generator and discriminator is powerful. Therefore, it is achievable to use the same method to generate human facial expressions.

To achieve what we expect, the model need to recognize different feature on human's face. While keeping the basic features of the face unchanged, different expressions can be generated on the same face by generating different mouth shapes, eyebrows, or some other facial features. Furthermore, if the model can learn more small changes on human's face, it can help generate a more natural face expression, but it need more studying. To achieve it, the choice of dataset and structure of DCGAN will be the most important part in this problem. In this project, we use Fer 2013 dataset to train 7 kinds of facial expression, angry, surprise, happy, disgust, fear, neural, and sad.

2 Related Work

The main idea of our method is related to Radford's paper, Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks (Radford et al., 2016). In his paper, he explains the detail structure of the generator and discriminator and points out the main difference between the normal GAN and DCGAN. Further, there are some papers related to reproduce missing pixels in the missing area in an image using DCGAN. Similarly, those papers use DCGAN to train a model learning the attributes of the image and reproduce those missing pixels to complete the image (Guo & Liu, 2020; Xu et al., 2020). Following those papers, the DCGAN can achieve the facial expression generation purpose, and we will build the DCGAN network based on the structure and modify it according to the dataset we use in experiment.

3 Method

In this project, we use the DCGAN method to train the model. As shown in Radford's paper, the basic structure will be like this.

- Generator

  ConvTranspose2d(100, 512, kernel size=(4, 4), stride=(1, 1), bias=False)
  BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track running stats=True)
  ReLU(inplace=True)
  ConvTranspose2d(512,256, kernel size=(4,4), stride=(2,2), padding=(1,1), bias=False)

BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track running stats=True)
ReLU(inplace=True)
ConvTranspose2d(256,128, kernel size=(4,4), stride=(2,2), padding=(1,1), bias=False)
BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track running stats=True)
ReLU(inplace=True)
ConvTranspose2d(128,64, kernel size=(4,4), stride=(2,2), padding=(1,1), bias=False)
BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track running stats=True)
ReLU(inplace=True)
ConvTranspose2d(64,3, kernel size=(4,4), stride=(2,2), padding=(1,1), bias=False)
Tanh()

- Discriminator

  Conv2d(3,64, kernel size=(4,4), stride=(2,2), padding=(1,1), bias=False)
  LeakyReLU(negative slope=0.2, inplace=True)
  Conv2d(64,128, kernel size=(4,4), stride=(2,2), padding=(1,1), bias=False)
  BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track running stats=True)
  LeakyReLU(negative slope=0.2, inplace=True)
  Conv2d(128,256, kernel size=(4,4), stride=(2,2), padding=(1,1), bias=False)
  BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track running stats=True)
  LeakyReLU(negative slope=0.2, inplace=True)
  Conv2d(256,512, kernel size=(4,4), stride=(2,2), padding=(1,1), bias=False)
  BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track running stats=True)
  LeakyReLU(negative slope=0.2, inplace=True)
  Conv2d(512, 1, kernel size=(4, 4), stride=(1, 1), bias=False)
  Sigmoid()

This structure is for 64*64 picture with 3 channels. For the Fer 2013 dataset, which contains picture with 48*48 and 1 channel. With respect to it, the first and last layer of generator need to be changed. All dataset will be trained with batch size equals to 128. We use Adam optimizer for generator and discriminator and set the beta parameter to 0.5. We use binary cross entropy for my loss function.

GAN is known to be a good model for learning representation from a great amount of unlabeled image. However, it is also unstable during training, resulting in generator that output a meaningless image. So, to get more reasonable output, DCGAN can achieve it with some changed compared to original GAN model.

In DCGAN, it doesn't use pooling layer. Instead, it uses strided convolution, allowing the convolution layer to learn its own upsampling in generator and downsampling in discriminator.

Also, it applys batch normalization after all convolution layer, except the output layer in generator and input layer in discriminator. Applying batch normalization can help DCGAN stabilize learning process. Not applying batch normalization after the output of generator and input of discriminator is to prevent oscillation.

Besides, it applys relu layer for all activation layer except the output layer of discriminator, where it uses Sigmoid layer for outputing a probability. And for generator, it applys Tanh layer after output layer, because using bounded activation function help the model learn quickly. It use leaky relu layer in discriminator instead of using normal relu layer. Leaky relu is proved to work well for higher resolution modeling (Radford et al., 2016).

4 Experiments

First, before using Fer 2013 dataset to train the model, we use Celeb A dataset to test the performance of DCGAN model. Celeb A dataset is large-scale face attributes dataset with more than 200k celebrity images. It is a recommended dataset for DCGAN training. The image in this dataset is 64*64 with 3 channels. Although there are many attributes for each image, we only use the image itself to train the model. For each datset, we have setup a fixed vector, which is a 100*1 vector, for performing the generator output.

Below is the training image sample in Celeb A dataset. These image are all good training image that mainly shows human face with different background.

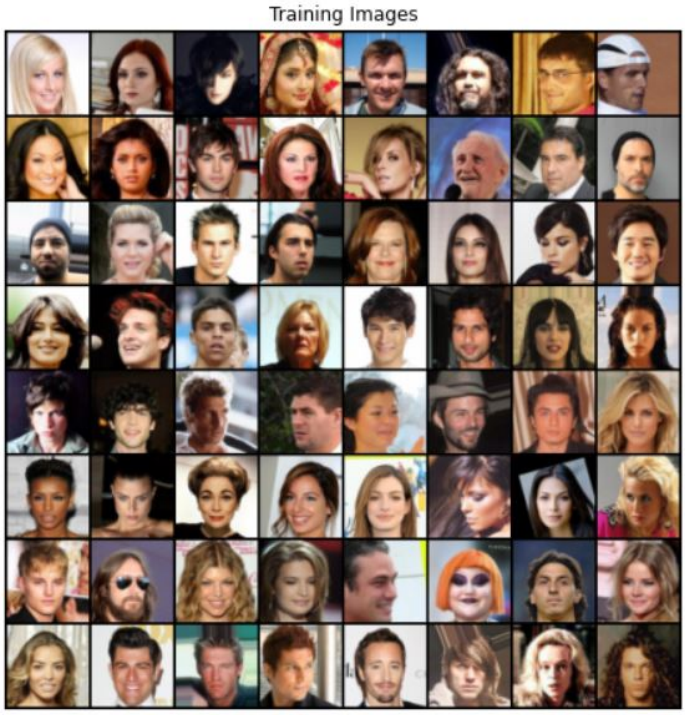

Figure 1: Celeb A dataset sample image

For this dataset, the epochs are set to 25, to have good performance. Below is the loss plot and the final output of the generator.

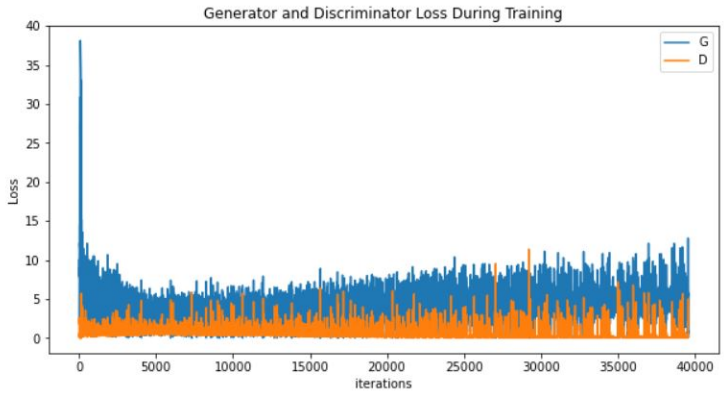

Figure 2: Celeb A dataset loss

Figure 3: Celeb A dataset generator output

Although the loss of the DCGAN is still oscillated, the average loss is low for generator and discriminator. And we can notice that fake images generated by generator are similar to real image. We can clearly distinguish the organs and expressions on the human face, and at the same time, the whole picture appears natural and there is no obvious unreasonableness. So, it proves that DCGAN can achieve our expectation.

Now, we use Fer 2013 dataset to train the DCGAN model. Fer 2013 contains more than 30k examples of 48*48 pixel grayscale images of faces. These images are categorized into 7 different group based on their facial expression. In order to generate specific facial expressions, we will train these seven expressions separately. After training separately, each expression has only about 5k samples. In order to have better training performance, I adjusted the epochs to 50. The following is the training sample of the FER2013 data set.

Figure 4: Fer 2013 image sample

Below is the loss and final generated output. From the loss plot we can know that it still has oscillation in loss, but the average loss is lower than the one in Celeb A dataset. But the fake images generated by the generator is not good. As we can see, fake images only have a rough outline of the face, and most of them are still blurred. Some complete pictures also have the problem of distorted facial expression. In general, these pictures can not achieve the effect of generating natural expressions.

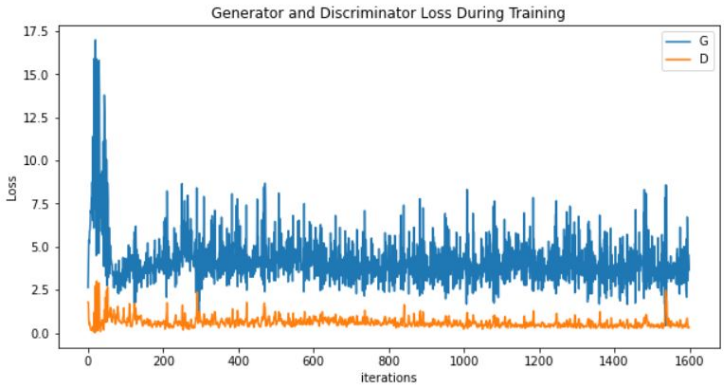

Figure 5: Fer 2013 loss

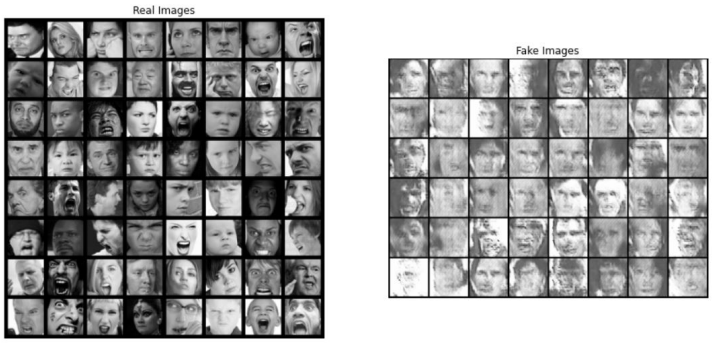

Figure 6: Fer 2013 generator output

In general, the DCGAN model is very effective in processing unlabeled images and finding their representations, just like the output shown in the Celeb A dataset. However, for the FER 2013 dataset, its performance is not good. There may be several reasons for this. One is that the images in the dataset are all grayscale, which is easy to mix with the background of the image, which makes it impossible to learn the facial features in the image well. The second is that there are too few samples in the dataset. Compared with the 200k samples in CELEBA, the single expression class sample of FER 2013 is only 5k. Too few samples also lead to poor training results. In order to obtain a better training effect, the later experiment will find a data set with a larger target sample size, and will focus on the generation of facial expressions in color pictures.

References

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., & Bengio, Y. (2014). Generative Adversarial Networks. ArXiv:1406.2661 [Cs, Stat]. http://arxiv.org/abs/1406.2661

Guo, J., & Liu, Y. (2020). Attributes guided facial image completion. Neurocomputing, 392, 60–69. https://doi.org/10.1016/j.neucom.2020.02.013

Radford, A., Metz, L., & Chintala, S. (2016). Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. ArXiv:1511.06434 [Cs]. http://arxiv.org/abs/1511.06434

Xu, S., Zhu, Q., & Wang, J. (2020). Generative image completion with image-to-image translation. Neural Computing & Applications, 32(11), 7333–7345. https://doi.org/10.1007/s00521-019-04253-2