Lab6 - Let's Play GANs

Shih-Po, Lee

- Deadline: May 26, 2020 11:59 a.m.
- Demo Date: May 26, 2020
- Format: Experimental report (.pdf) and source code (.py). Zip all files in one file and name it like DLP_LAB6_yourstudentID_name.zip. e.g. DLP_LAB6_0756051_Robert.zip.

1 Lab Description

In this lab, you need to implement a conditional GAN to generate synthetic images according to multi-label conditions. Recall that we have seen the generation capability of VAE in the previous lab. To achieve higher generation capacity, especially in computer vision, generative adversarial network (GAN) is proposed. In this lab, given a specific condition, your model should generate the corresponding synthetic images (see in Fig. 1). For example, given "red cube" and "blue cylinder", your model should generate the synthetic images with red cube and blue cylinder and meanwhile, input your generated images to a pre-trained classifier for evaluation.

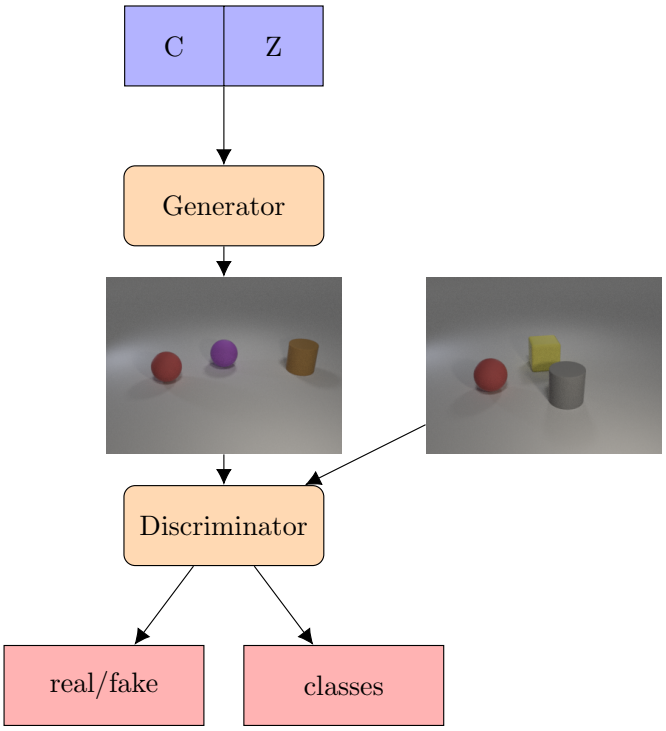

Figure 1: The illustration of cGAN.

2 Requirements

1. **Implement a conditional GAN**

- Choose your cGAN architecture.
- Design your generator and discriminator.
- Choose your loss functions.
- Implement the training function, testing function, and data loader.

2. **Generate the synthetic images**
   - Evaluate the accuracy of test.json
   - Evaluate the accuracy of new_test.json which will be released before demo.

3 Implementation Details

1. **Architecture of conditional GAN**

   There are Variants of conditional GAN architectures. You can choose one of the models in the following list or a hybrid version as your conditional GAN structure
   - conditional GAN
   - Auxiliary GAN
   - Projection discriminator

2. **Design of generator and discriminator.**

   Many GAN structures are developed to generator high quality images. Basically, all GAN architecture are based on DCGAN which is mainly composed of convolution neural network. You can adopt one of the structures or a hybrid
   - DCGAN
   - Super resolution GAN
   - Self-attention GAN
   - Progressive growing of GAN

3. **Training loss functions**

   Besides the significant progress of GAN architectures, the training function is also play an important role. Your loss function should combine with your choice of cGAN as it will take a part in your training function, e.g., auxiliary loss. Here are the reference training loss functions version in the following list
   - GAN
   - LSGAN
   - WGAN
   - WGAN-GP

4. **Other implementation details**
   - Your can ignore previous suggestion and choose any GAN architecture you like.
   - Except for the provided files, you cannot use other training data. (e.g. background image)
   - Use the function of a pre-trained classifier, *eval*(images, labels) to compute accuracy of your synthetic images.
   - Use *make_grid*(images) and *save_image*(images, path) (from torchvision_utils import save_image, make_grid) to save your image (8 images a row, 4 rows).
   - The same object will not appear twice in an image.
   - The normalization of input images is transforms.Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5)). You should not apply normalization on your generated images but apply de-normalization while generating RGB images.

4 Dataset Descriptions

You can download the dataset.zip file from new e3, classifier_weight.pth and iclevr.zip from lab6 in open source google drive. There are 5 files in the dataset.zip: readme.txt, train.json, and test.json, object.json, and evaluator.py. All the details of the dataset are in the readme.txt.

5 Scoring Criteria

1. Report (40%)
   - Introduction (5%)
   - Implementation details (15%)
     - Describe how you implement your model, including your choice of cGAN, model architectures, and loss functions. (10%)
     - Specify the hyperparameters (learning rate, epochs, etc.) (5%)
   - Results and discussion (20%)
     - Show your results based on the testing data. (5%)
     - Discuss the results of different models architectures. (15%) For example, what is the effect with or without some specific loss terms, or what kinds of condition design is more effective to help cGAN.
2. Demo (60%)
   - Classification accuracy on test.json and new_test.json. (20% + 20%)

Accuracy	Grade
score ≥ 0.8	100%
$0.8 >$ score ≥ 0.7	90%
$0.7 >$ score ≥ 0.6	80%
$0.6 >$ score ≥ 0.5	70%
$0.5 >$ score ≥ 0.4	60%
score < 0.4	0%

   - Questions (20%)