Lab 6: Let's play GANs!

- **Lab objective**

In this lab, you need to implement a conditional GAN to generate synthetic images according to different conditions.

- **Important Date**

1. Deadline: 5/26 (Tue.) 11:59 a.m.
2. Demo date: 5/26 (Tue.)

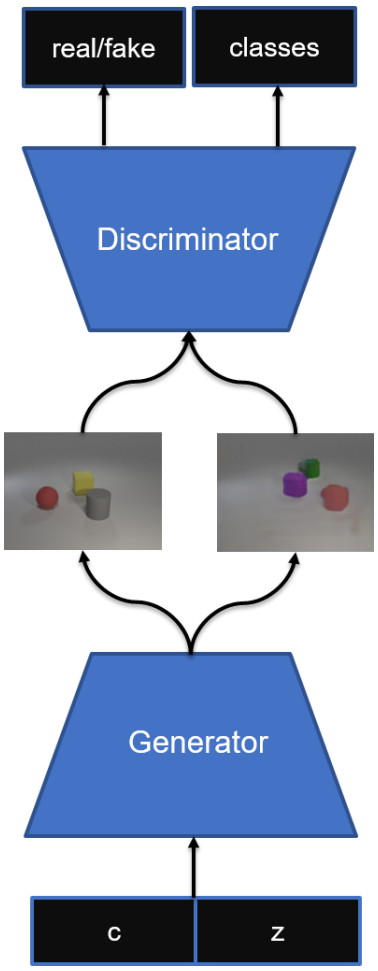
- **Format**

1. Experimental Report (.pdf) and Source code

Notice: zip all files in one file and name it like DLP_LAB6_your studentID_name.zip. e.g. DLP_LAB6_0756051_李仕柏.zip

- **Lab Description**

In the previous lab, we have seen the generation capability of VAE. To achieve higher generation quality, especially in computer vision, generative adversarial network has been proposed and variants of GAN are widely studied and applied to different applications such as image synthesis and style transfer. In this lab, given a specific condition, your model should generate the corresponding synthetic images. For example, given "red cube" and "blue cylinder", your model should generate the synthetic images with red cube and blue cylinder and meanwhile, the results should get a high accuracy by a pre-trained classifier. The figure below is the illustration of cGAN.

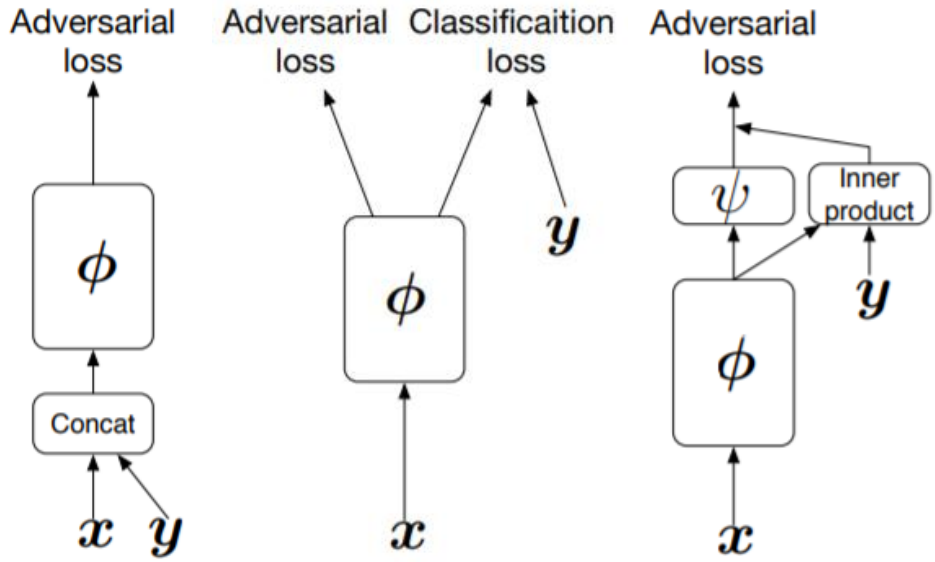


- **Requirements**

1. Implement a conditional GAN.
   A. Choose your cGAN architecture.
   B. Design your generator and discriminator.
   C. Choose your loss functions.
   D. Implement the training function, testing function, and dataloader.
2. Generate the synthetic images based on test.json and new_test.json (will be released before demo), compute the accuracy, and show the results while demo.

- **Implementation details**

1. Variants of conditional GAN.



From left to right are cGAN [1], ACGAN [2], and projection discriminator [3] respectively (figure from [3]). You can choose one of them or a hybrid version as your cGAN structure.

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 version in the following list
   - DCGAN [4]
   - Super resolution GAN [5]
   - Self-Attention GAN [6]
   - Progressive growing of GAN [7]
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 [12].
   - GAN [8]
   - LSGAN [9]
   - WGAN [10]
   - WGAN-GP [11]

4. Other implementation details
   - You can ignore previous suggestion and can choose any GAN architecture you like.
   - Use the function of a pretrained 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)

- ## Dataset Descriptions

  You can download the dataset.zip file from new e3, iclevr.zip and classifier_weight.pth 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.

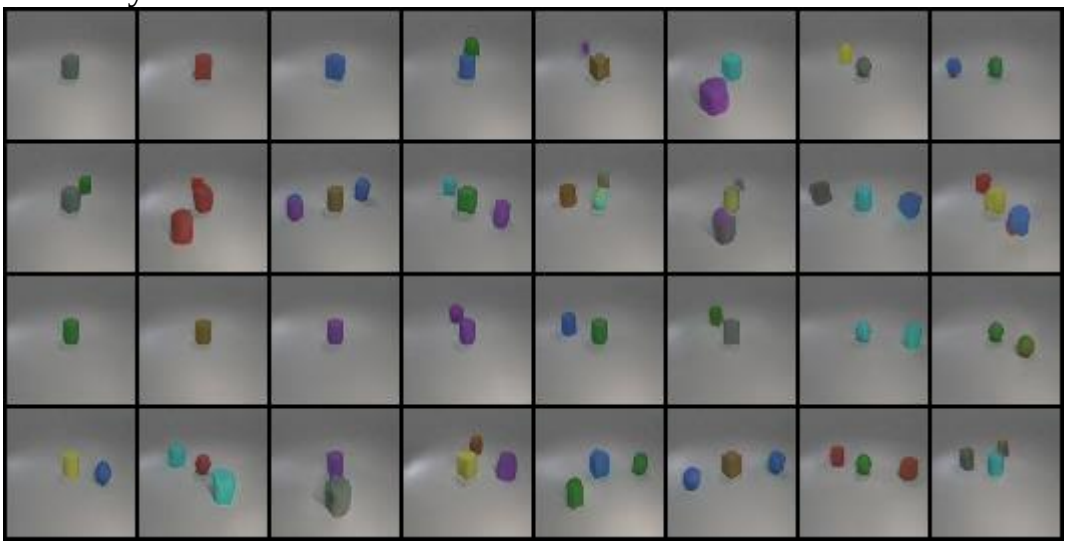
- ## Scoring Criteria

1. Report (40%)
   - Introduction (5%)
   - Implementation details (15%)
     A. Describe how you implement your model, including your choice of cGAN, model architectures, and loss functions. (10%)
     B. Specify the hyperparameters (learning rate, epochs, etc.) (5%)
   - Results and discussion (20%)
     A. Show your results based on the testing data. (5%)
     B. 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%)
   A. Classification accuracy on test.json and new_test.json. (20% + 20%)
      - 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%

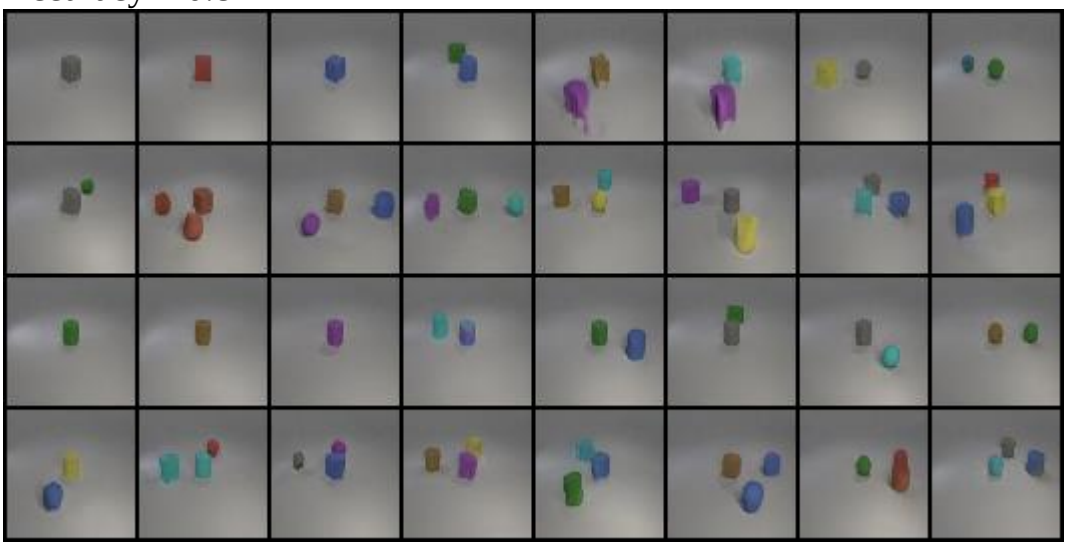
   B. Questions (20%)

- **Output examples**

1. Accuracy = 0.667



2. Accuracy = 0.847



- **Reference**

[1] cGAN: https://arxiv.org/abs/1411.1784
[2] ACGAN: https://arxiv.org/abs/1610.09585
[3] Projection discriminator: https://arxiv.org/abs/1802.05637
[4] DCGAN: https://arxiv.org/abs/1511.06434
[5] Super resolution GAN: https://arxiv.org/pdf/1609.04802
[6] Self-Attention GAN: https://arxiv.org/abs/1805.08318
[7] Progressive growing of GAN: https://arxiv.org/abs/1710.10196
[8] GAN: https://arxiv.org/abs/1406.2661
[9] LSGAN: https://arxiv.org/abs/1611.04076
[10] WGAN: https://arxiv.org/abs/1701.07875
[11] WGAN-GP: https://arxiv.org/pdf/1704.00028
[12] loss functions: http://www.twistedwg.com/2018/10/05/GAN_loss_summary.html