NYCU 2022 Spring DLP Lab7 – Let's Play GANs

TA 陳鵬宇

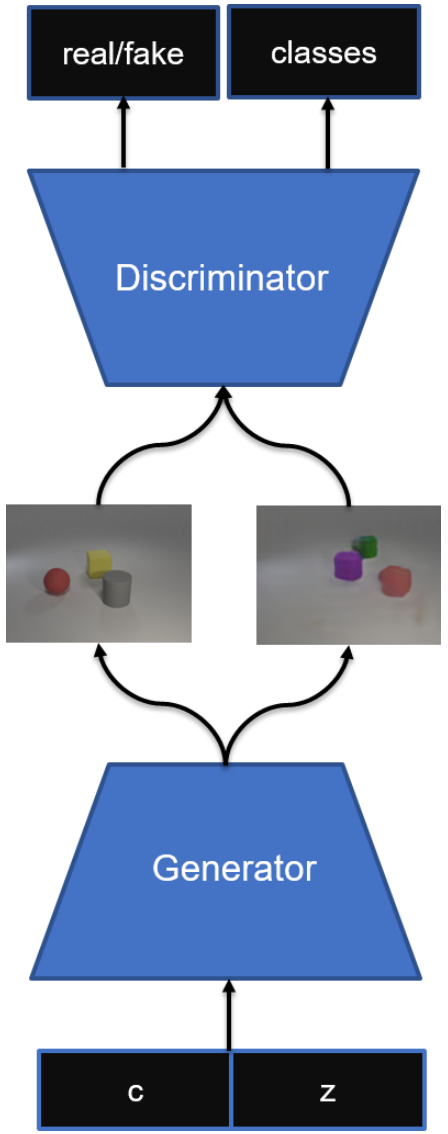
May 17, 2022

Outline

- Lab Objective
- Important Date
- Lab Description
- Scoring Criteria

Lab Objective

- In this lab, you need to implement a conditional GAN and generate synthetic images based on multi-labels conditions
- Example of labels:
  - [“cyan cylinder”, “red cube”], [“green sphere”], …



Important Date

- Experiment Report Submission Deadline: 6/7 (Tue.) 11:59 a.m.
- Demo date: 6/7 (Tue.)
- Zip all files in one file
  - Report (.pdf)
  - Source code
- Name it like ”DLP_LAB7_yourstudentID_name.zip”
  - Example: “DLP_LAB7_309551113_陳鵬宇.zip”
- -5% to your score if you do not follow the format

Lab Description

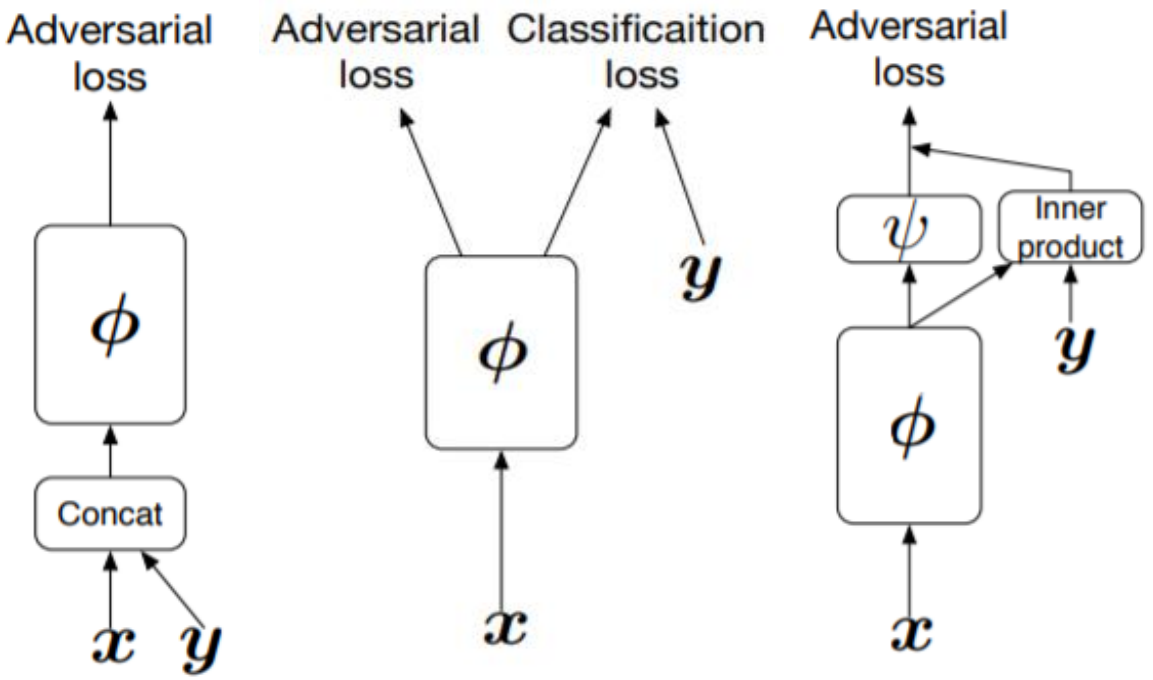
- Implementation details
  - Choose your conditional GAN architecture
  - Design your generator and discriminator
  - Choose your loss function

Lab Description – Choice of cGAN

- Generator
  - Concatenation, multiplication, batch normalization, etc.

- Discriminator
  - Conditional GAN
  - InfoGAN
  - Auxiliary GAN
  - Projection discriminator

- Hybrid version



Lab Description – Design of GAN

- De-convolution layers
- Basic block
- Bottleneck block
- Residual
- Self-attention
- Again, hybrid version
- E.g.
  - DCGAN
  - SA-GAN
  - Progressive GAN

Lab Description – Choice of loss functions

- GAN loss function
  - $L_D^{GAN} = E[\log(D(x))] + E[\log(1 - D(G(z)))]$

    $L_G^{GAN} = E[\log(D(G(z)))]$
- LSGAN
- WGAN
- WGAN-GP

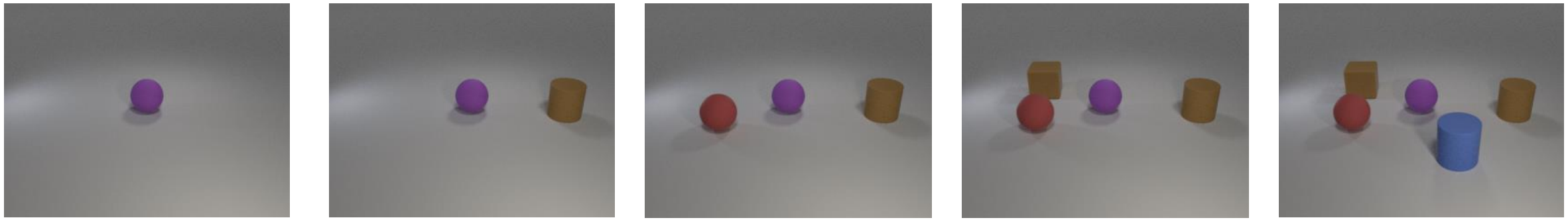
- Combine with your choice of cGAN

Lab Description – Other details

- You can use any GAN architecture your like
- Use the function of a pretrained classifier, **eval**(images, labels), to compute accuracy of your synthetic images.
  - Labels should be one-hot vector. E.g. [[1,1,0,0,…],[0,1,0,0,…],…]
  - Images should be all generated images. E.g. (batch size, 3, 64, 64)
- 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 resolution of input for pretrained classifier is 64x64. You can degisn your own output resolution for generator and resize it.
- Some tips: https://github.com/soumith/ganhacks.

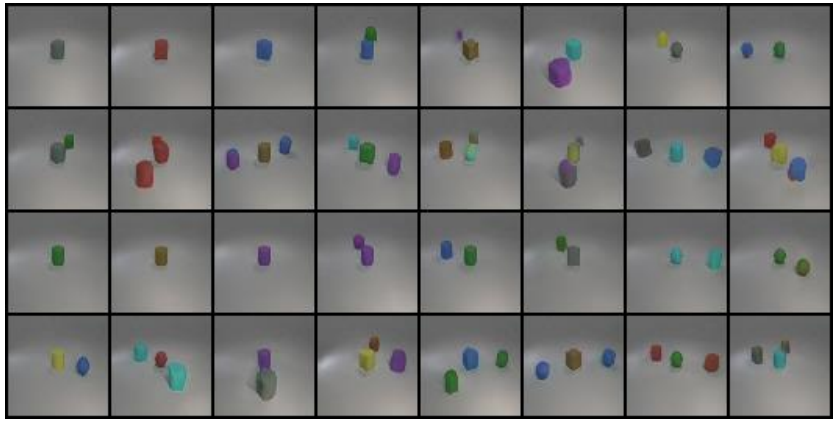
Lab Description - Dataset

- Provided files
  - readme.txt, train.json, test.json, object.json, iclevr.zip, evaluator.py, checkpoint.pth
- iclver.zip
  - On the open source google drive
- object.json
  - Dictionary of objects
  - 24 classes

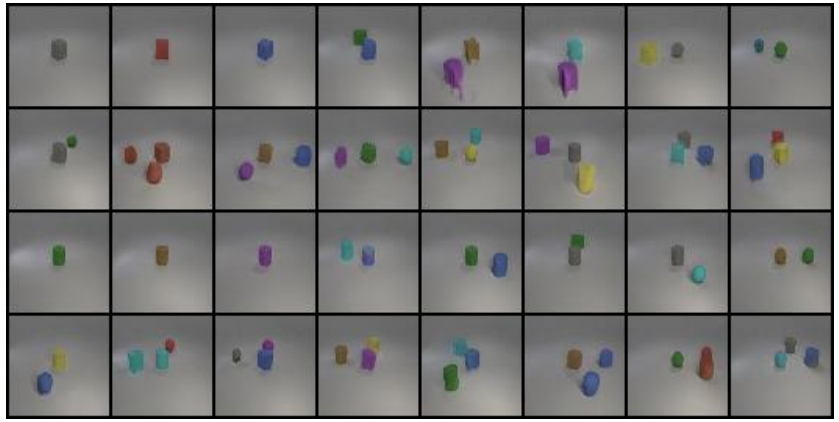


Lab Description – Output examples

Accuracy: 0.667



Accuracy:0.847



Lab Description – Requirements

- Implement training, testing functions, and dataloader
- Choose your cGAN architecture
- Design your generator and discriminator
- Choose your loss functions
- Output the results based on test.json and new_test.json (will be released before demo)

Scoring Criteria

- 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

Scoring Criteria

- Demo (60%)
  - 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%
  - Questions (20%)