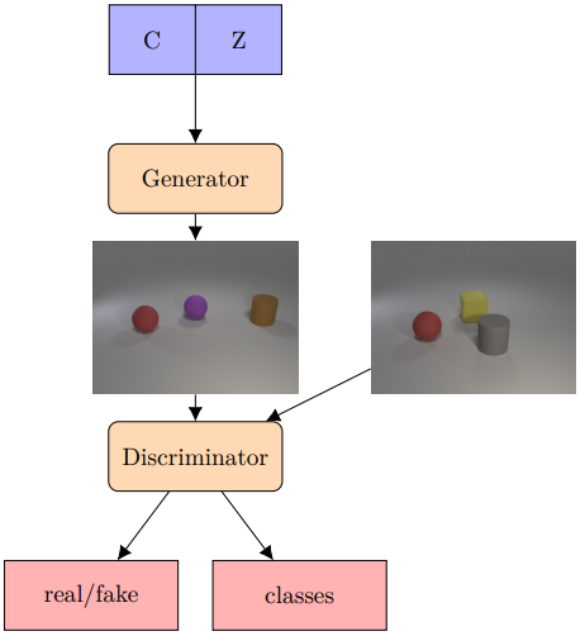
DL Lab5 report - Let's Play GANs

匯出pdf之後版面變得有點醜QQ
助教不介意的話可以直接進HackMD看 謝謝～～～
https://hackmd.io/@ZdXM6gDQSTGTtZGr5jTo4Q/SJwwaZYRc (https://hackmd.io/@ZdXM6gDQSTGTtZGr5jTo4Q/SJwwaZYRc)

Introduction

In this lab, we need to implement a conditional GAN to generate synthetic images according to multi-label conditions. Given a specific condition, the model should generate the corresponding synthetic images. For example, given "red cube" and "blue cylinder", the model should generate the synthetic images with red cube and blue cylinder.



Requirments

- Implement a conditional GAN
- Generate the synthetic images

Dataset

- object.json
  A dictionary file that contains the number of objects and the idexes.
  There are totally 24 objects in i-CLEVR datasets with 3

shapes and 8 colors.

- train.json

  A dictionary where keys are filenames and values are objects.
  For example:
  {“CLEVR_train_001032_0.png”: [“yellow sphere”],
  “CLEVR_train_001032_1.png”: [“yellow sphere”, “gray cylinder”],
  “CLEVR_train_001032_2.png”: [“yellow sphere”, “gray cylinder”, “purple cube”], ... }
  One image can include objects from 1 to 3.
  The number of training data is 18009.

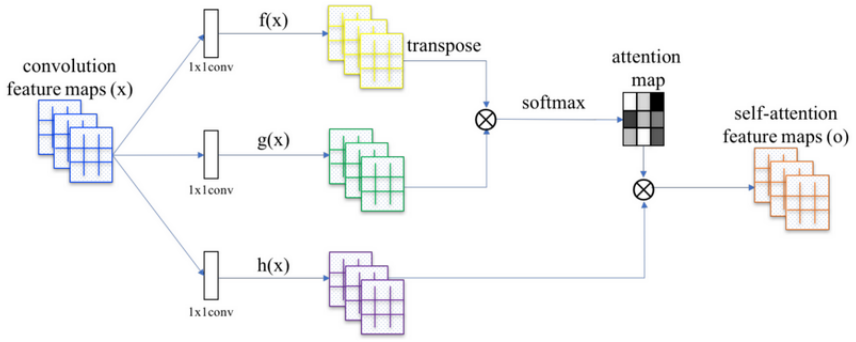
- test.json

  A list where each element includes multiple objects.
  For example:
  [[‘gray cube’],
  [‘red cube’],
  [‘blue cube’],
  [‘blue cube’, ‘green cube’], ...]
  The number of testing data is 32.

Implementation details

Describe how you implement your model

- Choice of cGAN: Self-attention GAN



- Model architecture

attn2.value_conv

weight〈64×64×1×1〉
bias〈64〉

attn2.key_conv

weight〈8×64×1×1〉
bias〈8〉

attn2.query_conv

weight〈8×64×1×1〉
bias〈8〉

attn2

gamma〈1〉

attn1.value_conv

weight〈128×128×1×1〉
bias〈128〉

attn1.key_conv

weight〈16×128×1×1〉
bias〈16〉

attn1.query_conv

weight〈16×128×1×1〉
bias〈16〉

attn1

gamma〈1〉

l5

weight〈64×3×4×4〉
bias〈3〉

l4.1

weight〈64〉
bias〈64〉
running_mean〈64〉
running_var〈64〉
num_batches_tracked = 353628

l4.0

weight〈128×64×4×4〉
bias〈64〉

l3.1

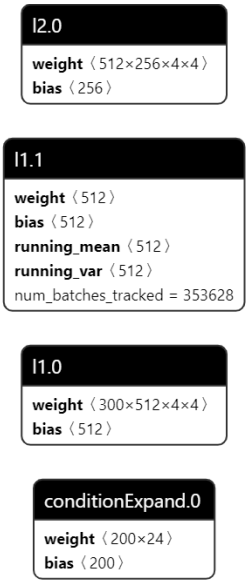
weight〈128〉
bias〈128〉
running_mean〈128〉
running_var〈128〉
num_batches_tracked = 353628

l3.0

weight〈256×128×4×4〉
bias〈128〉

l2.1

weight〈256〉
bias〈256〉
running_mean〈256〉
running_var〈256〉
num_batches_tracked = 353628



- Loss function
  SAGAN使用Hinge loss，目的是增加對分類影響較大的數據點的權重，減少與分類關係較小的數據點的權重

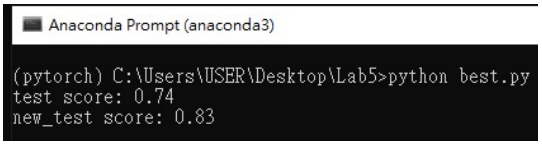
$$L_D = E[max(0, 1 - D(x))] + E[max(0, 1 + D(G(z)))]$$
$$L_G = -E[D(G(z))]$$

Specify the hyperparameters (learning rate, epochs, etc.)

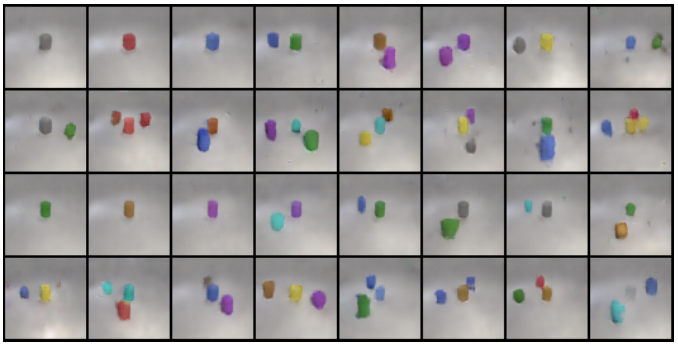
learning rate = 1e-4
epochs = 300
batch size = 64
random seed = 1

Results and discussion

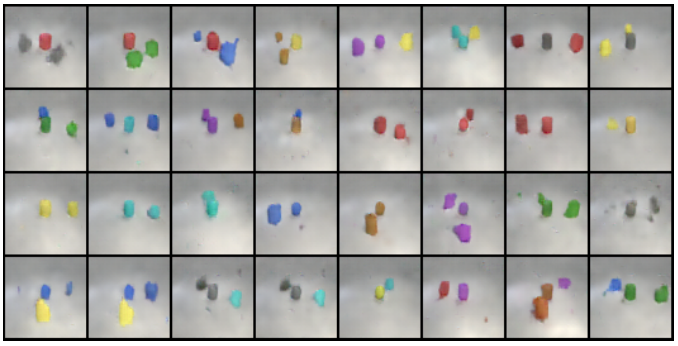
Show your results based on the testing data



- test.json



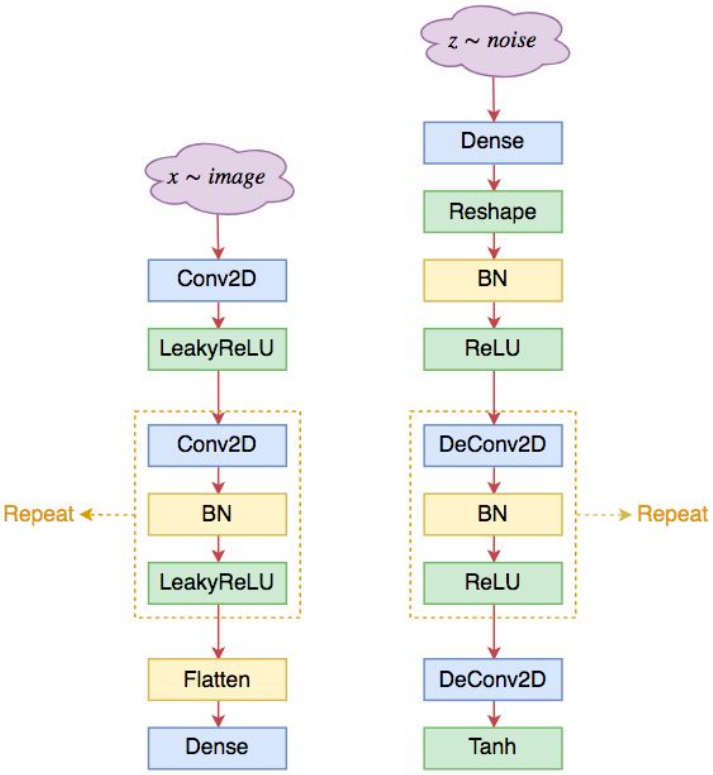
- new_test.json



Discuss the results of different models architectures

這次的lab我嘗試過DCGAN和SAGAN，上面的結果是使用SAGAN實作出來的。因為在使用DCGAN的時候，score最高只能到0.69就上不去了，所以就再試了SAGAN，發現結果比較好。
SAGAN和DCGAN之間的改動其實並不大，標準的SAGAN只是在普通的DCGAN架構中間插入self attention層。
DCGAN的模型架構大致如下：

1. generator和discriminator均不採用pooling layer，採用convolution layer，其中discriminator採用Conv2D，而generator採用DeConv2D
2. 在generator和discriminator上均使用batch normalization
3. 在generator除輸出層外的所有層上使用ReLU，而輸出層使用Tanh
4. 在discriminator的所有層上使用LeakyReLU
5. convolution layer之後不使用fully connected layer
6. discriminator的最後一個卷積層之後也不用global pooling，而是直接flatten



另外，使用DCGAN時我使用的loss function是binary cross entropy。