Lab5 : Let’s Play GANs

311511043 李承翰

1. Introduction

在這一次的lab當中我們要實作出conditional GAN, 這部分跟上一次的lab有相似的地方，也就是要藉由著將標籤輸入到網路當中，讓我們可以決定出要生成怎麼樣的資料，而我們使用ICLEVR的圖片當作dataset，在這一個dataset當中總共有24種幾何物體，也因此這次我們的conditional dim這一個變數將會是24

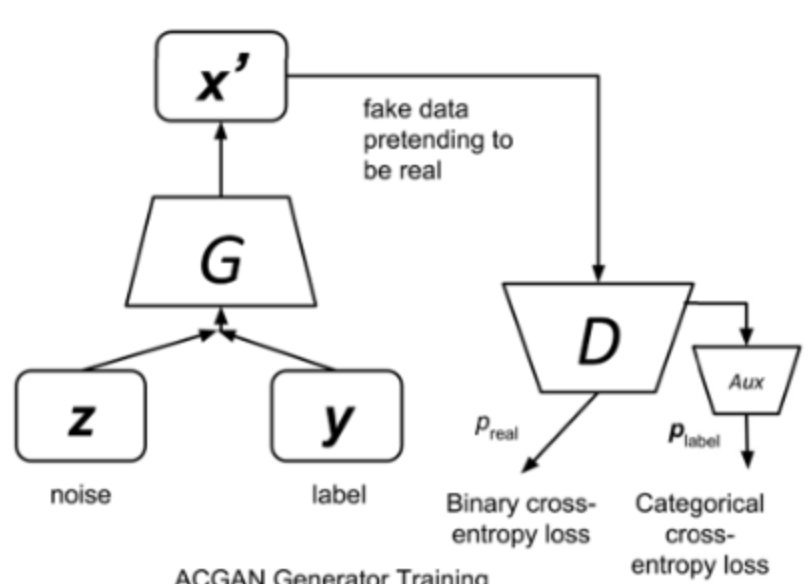
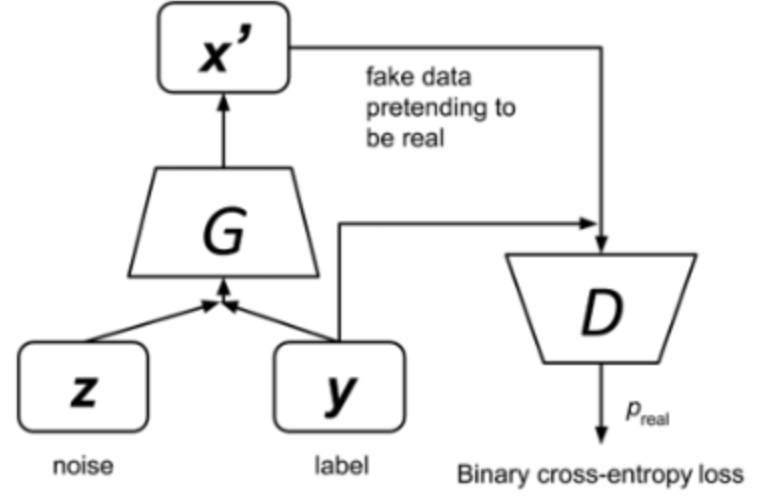
而GAN的全名是Generative Adversarial Network，也就是生成對抗網路，在這一種網路模型當中有兩個子模型，分別是generator(生成網路)、以及discriminator(判別網路)，這兩個模型互相的訓練，當generator生成了新的資料後，交由discriminator判別這個被生成的資料像不像真實的資料，並將這個結果回饋給generator，使其改善。藉由這樣一來一往的訓練，generator的目標就是想盡辦法使discriminator無法辨別是否是生成的資料，而discriminator就是要盡可能地將生成的資料與真實資料區分開來。

1. Implementation details
2. Describe how you implement your model(choice of cGAN, model architecture, loss functions)
3. Choice of cGAN

在這次的lab我嘗試了兩種cGAN, 分別是DCGAN以及ACGAN

1. Model architecture
   1. ACGAN

ACGAN的全稱是Auxiliary Classifier GAN，ACGAN的主要架構與普通的cGAN類似，只是她針對discriminator做了一些改良，除了產生一個針對生成圖片的相似度評分以外，discriminator還多了一個classifier用以計算classifier loss



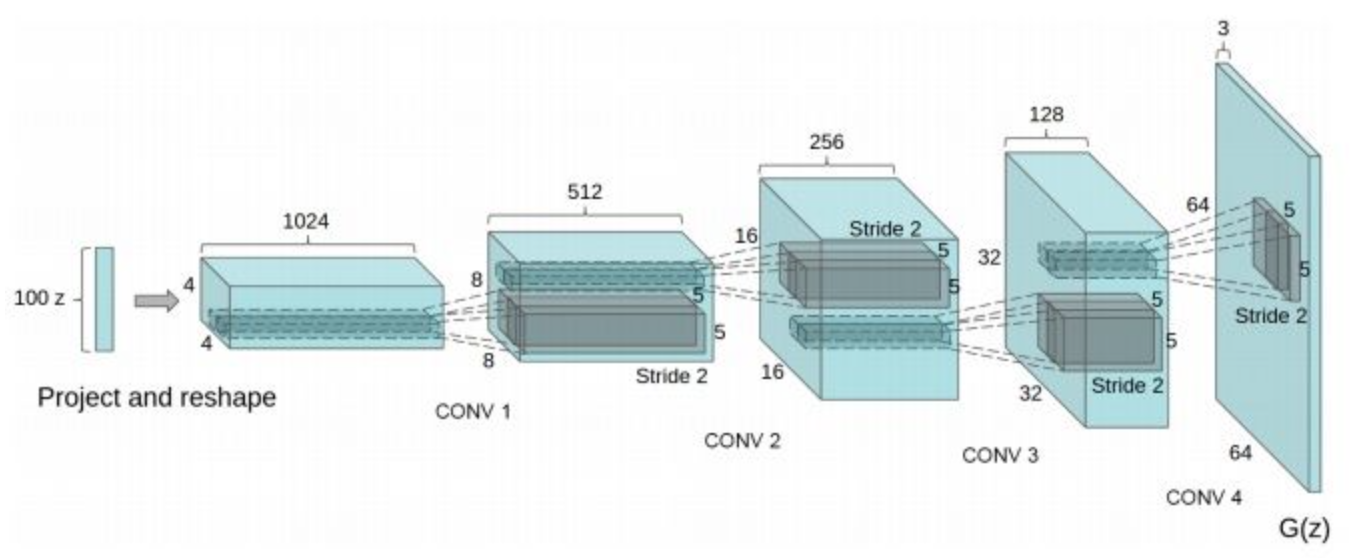
左圖是普通的cGAN架構，右圖是ACGAN的架構

在ACGAN的理論當中，作者認為如果我們讓模型去執行本職以外的任務，那麼應該可以提高原始的任務的性能，因此在discriminator旁邊多了一個Aux的方塊，其任務就是classifier，作為本職以外的任務，她需要做的事情是針對生成的圖像進行分類，而本職任務就是生成圖片。

* 1. DCGAN

DCGAN是一種改良版本的GAN，不同於普通GAN使用全連接層，DCGAN使用了CNN來取代之。

Discriminator使用了正常的CNN，只是將池化層都捨棄，而Generator使用的是反捲積層並且，在不管是discriminator或者Generator都使用了Batch Normalization，這麼一來就可以解決在初期的時候初始化隨機性帶來的影響，提升整體訓練的穩定性。



DCGAN的generator架構

1. Loss functions
   1. ACGAN

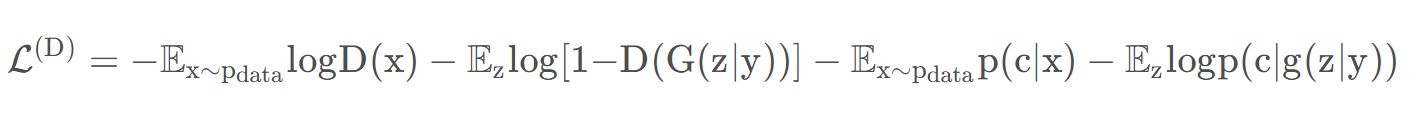
在generator的部分，loss function應為

一張含有 文字 的圖片

自動產生的描述

在實作當中，我們使用pytorch的 BCELoss來執行

在discriminator當中，loss function應為



針對圖片真偽度的評分，我們使用BCELoss

classifier的部分則是使用CrossEntropyLoss

但是在實驗中發現，因為Pytorch的CrossEntropyLoss當中含有softmax，對模型表現造成了不太好的影響，因此後來我也改用BCELoss

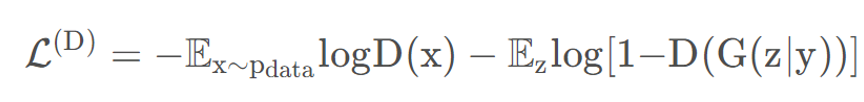
* 1. DCGAN

在generator的部分是

一張含有 文字 的圖片

自動產生的描述

也因此與ACGAN相同，我們使用BCELoss

而在discriminator端，我們沒有Aux的輔助任務，因此是: 

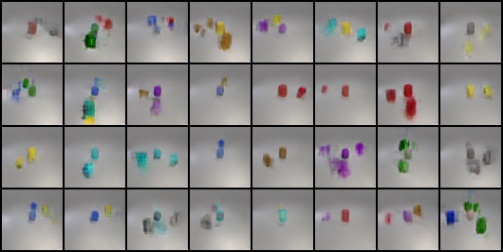
而實作中就使用BCELoss

1. Specify the hyperparameters
2. Learning rate for generator: 0.0001
3. Learning rate for discriminator:0.0002
4. Optimizer for generator: adam, betas = [0.5,0.999]
5. Optimizer for discriminator: sgd, momentum = 0.9
6. Batch size = 32
7. Epoch = 300
8. Dropout rate 0.5
9. Result and Discussion
10. Show your results based on testing data( including images predicted)

一張含有 文字 的圖片

自動產生的描述

|  |  |  |
| --- | --- | --- |
|  | ACGAN | DCGAN |
| Best acc(test.json) | 81.944 | 73.12 |
| Best acc(new\_test.json) | 83.33 | 75.66 |

 一張含有 團體, 一對, 差異, 彩色 的圖片

自動產生的描述

ACGAN new\_test.json ACGAN test.json

1. Discuss the results of different models architectures

在我嘗試的兩種模型當中有DCGAN以及ACGAN，而在討論A的圖表當中我們可以看到，ACGAN跟DCGAN相比，是ACGAN的效果表現較好，這也符合一開始我的預期，畢竟ACGAN除了生成圖片的任務以外，還有一個輔助性的任務，以期可以提高他生成圖片這一個任務的正確率。

而在loss fcn的差異當中，本來在原論文中，ACGAN是使用CrossEntropy來當中他輔助任務的損失函數，只是因為在pytorch當中將argmax也包含在了CrossEntropy當中，這使得本來有明確分類的資料標籤反而失去了其意義，因此造成了訓練結果不佳，所以我將CrossEntropy換成了BinaryCrossEntropy來解決這一個問題。而DCGAN的部分，我有觀察到在這次模型當中如果不是使用adam這一個optimizer，而是使用RMSProp那可以有效的改善模型不穩定的問題。而在助教提供的github當中我也使用了其中的幾個tips來試圖讓模型的表現更好，例如:

* + - 1. 分別對generator及discriminator使用不同的learning rate
      2. 針對discriminator使用sgd的optimizer
      3. 以一定的機率去交換fake label還有true label