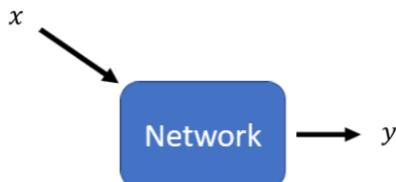


Introduction

Generator

接下来要进入一个,新的主题 我们要讲生成这件事情

到目前為止大家学到的network,都是一个function,你给他一个X就可以输出一个Y



我们已经学到各式各样的, network架构,可以处理不同的X 不同的Y

我们学到输入的X

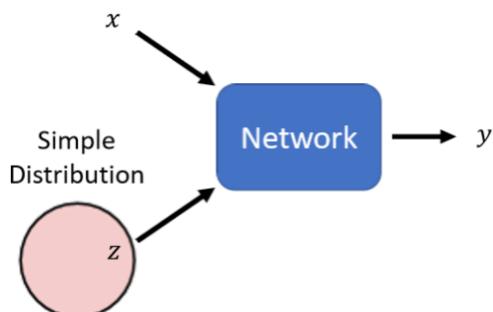
- 如果是一张图片的时候怎麽办
- 如果是一个sequence的时候怎麽办

我们也学到输出的Y

- 可以是一个数值
- 可以是一个类别
- 也可以是一个sequence

接下来我们要进入一个新的主题,这个新的主题是要把network,当做一个generator来用,我们要把network拿来做生成使用

那把network拿来,当作generator使用,他特别的地方是现在network的输入,会加上一个**random的variable,会加上一个Z**



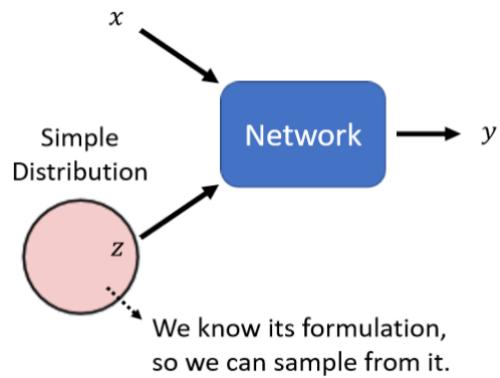
这个Z,是从某一个,distribution sample出来的,所以现在network它不是只看一个固定的X得到输出,它是同时看X跟Z得到输出

network怎麽同时看X跟Z,有很多不同的做法,就看你怎样设计你的network架构

- 你可以说X是个向量,Z是个向量 两个向量直接接起来,变成一个比较长的向量,就当作network的input
- 或者是你的X跟Z正好长度一模一样,把它们相加以后,当做network的input
- 等等

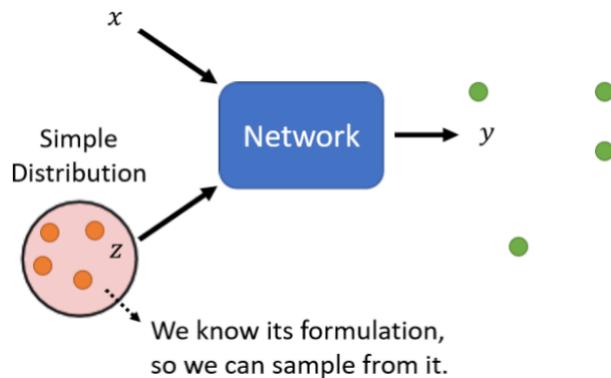
Z特别的地方是 它是不固定的,每一次我们用这个network的时候,都会随机生成一个Z,所以Z每次都不一样,它是从一个distribution裡面,sample出来的

这个distribution,这边有一个限制是,它必须够简单,够简单的意思是,我们知道它的式子长什麼样子,我们可以从这个distribution,去做sample

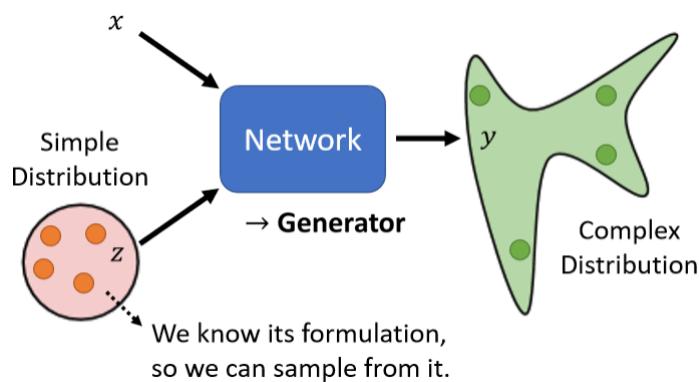


举例来说这个distribution,可以是一个function distribution,你知道function distribution的式子,你知道怎麽从,gaussian distribution做sample

它也可以是uniform distribution,那uniform distribution,的式子你一定知道,你也知道怎麽从,uniform distribution做sample,所以这一个distribution,的形状你自己决定,但你只要记得说它是简单的,你能够sample它的就结束了



所以每次今天,有一个X进来的时候,你都从这个distribution,裡面做一个sample,然后得到一个output,随著你sample到的Z不同,Y的输出也就不一样,所以这个时候我们的network输出,不再是单一个固定的东西,而变成了一个复杂的distribution,同样的X作为输入,我们这边每次sample到,不一样的东西,通过一个复杂的network转换以后,它就会变成一个复杂的分布,你的network的输出,就变成了一个distribution



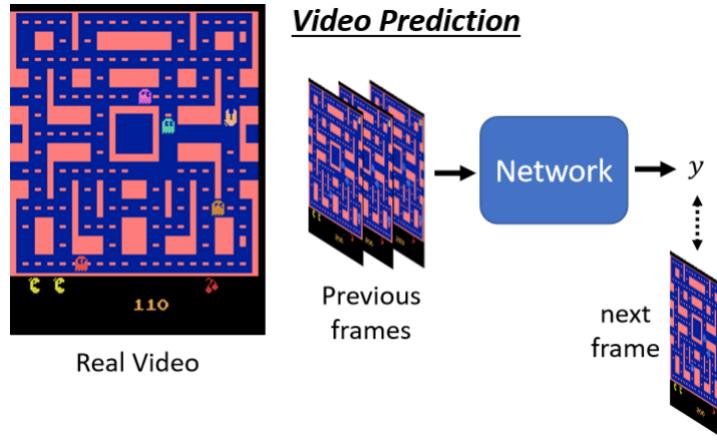
那这种可以输出,一个distribution的network,我们就叫它generator

Why We need Distribution

在讲怎麽训练出generator之前,我们第一个想要回答的问题是,為什麼我们需要generator输出是一个分布? 输入X输出Y,这样固定的输入跟输出关係不好吗?

所以以下就举一个例子来跟你说明,為什麼输出有时候需要是一个分布

这边举的例子,是video prediction,就是给机器一段的影片,然后它要预测接下来会发生什麼事情



那这个例子,是我从上面这个,github的连结 https://github.com/dyelax/Adversarial_Video_Generation 找到的,那在这个连结裡面 它要做的事情,是去预测小精灵这个游戏,接下来的游戏画面会长什麼样子

video prediction,那你就给你的network过去的游戏画面,然后它的输出就是新的游戏画面,下一秒的下一个时间点的游戏画面

有人可能会问说怎麽输出一张图片?

这个一张图片就是一个很长的向量,所以你只要让你的network,可以输出一个很长的向量,然后这个向量整理成图片以后,跟你的目标越接近越好

其实在这个github裡面,它不是直接把整个画面当做输入,它是从画面裡面只切一小块当做输入,就它把这个整个画面切成很多块,然后分开来处理,不过我们这边為了简化说明,就当作network是一次,输入一整个这样的画面

如果你用我们学过的network training的方法,Supervise learning train下去,你得到的结果可能会是这个样子,这个是机器预测出来的结果(第三部分 Non-Adversarial), 所以你看有点模模糊糊的,而且中间还会变换角色,神奇的是那个小精灵,走著走著 居然就分裂了,它走到转角的时候,看它走到转角的时候就分裂成两隻了,走到这边又再分裂成两隻了,有时候走著走著还会消失

因为今天对这个network而言,在训练资料裡面同样的输入,有时候同样的转角

- 有时候小精灵是往左转
- 有时候小精灵是往右转,

这样这两种可能性,同时存在你的训练资料裡面

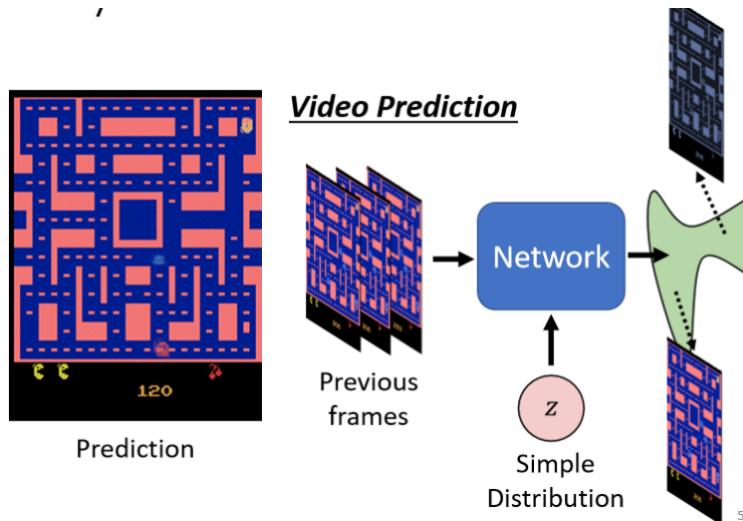
你在训练的时候,今天你的network,得到的训练的指示是,给定这一段输入,那今天得到这一笔训练资料,那它就要学到给定这段输入,输出应该要往右转,给定这一些训练资料,有时候你会看到的是向左转,那机器就会学到给定这一段输入,它要向左转

所以你的network,学到的就是两面讨好,

因为它需要得到一个输出,这个输出同时距离向左转最近,同时也距离向右转最近,那怎麽样同时距离向左转最近,向右转最近,也许就是同时向左也向右转

所以你的network,它就会得到一个错误的结果,向左转是对的 向右转也是对的,但是同时向左向右转 反而是不对的,

那有什麼样的可能性,可以处理这个问题,一个可能性就是,让机器的输出是有机率的,让它不再是输出单一的输出,让它输出一个机率的分佈



当我们给这个network,一个distribution的时候,当我们给这个network input,加上一个Z的时候,它的输出就变成了一个distribution,它的输出就不再是固定的

我们希望训练一个network,它可以知道说它的输出的这个分佈,包含了向左转的可能,也包含向右转的可能。举例来说假设你选择你的Z,是一个比如说,binary的random variable,它就只有0跟1 那可能各佔50%,也许你的network就可以学到说,Z sample到1的时候就向左转,Z sample到0的时候就向右转,就可以解决,这个世界有很多不可预测的东西,的状况

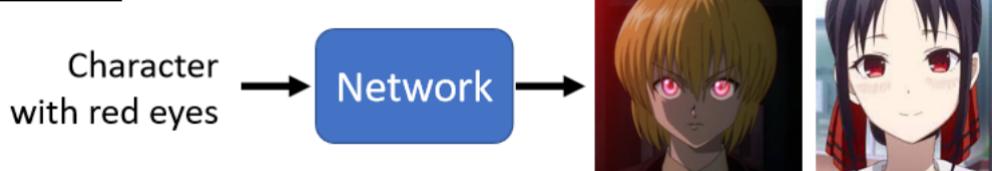
那什麼时候我们会特别需要,处理这种问题,什麼时候,我们会特别需要这种generator的model,当我们的任务需要一点创造力的时候。任务需要一点创造力这件事情,是比较拟人化的讲法,更具体一点的讲法就是,我们想要找一个function,但是同样的输入有多种可能的输出,而这些不同的输出都是对的,

举例来说,画图这件事情,可能就需要一些创造力

(The same input has different outputs.)

- Especially for the tasks needs “creativity”

Drawing



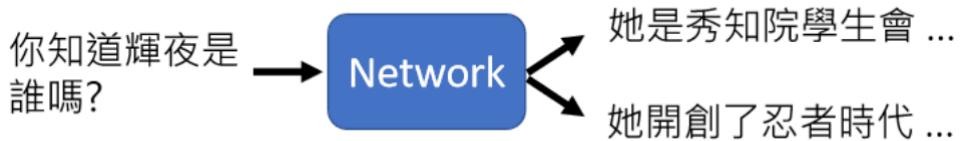
举例来说假设叫一个人,画一个红眼睛的角色,那每个人可能画出来,或者心裡想的动画人物可能都不一样,有哪些角色是红眼睛的

举例来说库拉皮卡是红眼睛的,它是窟卢塔族的,窟卢塔族的愤怒以后,就会有火红眼,那辉夜也是红眼睛的,那因为这个库拉皮卡从黑暗大陆回来以后 就用他在黑暗大陆得到的资源成立了四宫集团,辉夜其实是库拉皮卡的子孙,那她那一代火红眼变成是显性的,不用生气火红眼也会显现出来,所以库拉皮卡跟辉夜,他们都有火红眼,所以同样要画一个红眼睛的角色,每个人心裡想像的红眼睛的角色,都是不一样的

那这个时候,我们就需要让机器能够,让我们的model,能够output一个distribution

那还有什麼样的例子,会需要用到创造力,举例来说 对话这件事情

Chatbot



举例来说假设你跟另外一个人说,你知道辉夜是谁吗,其实有很多不同的答案对不对

辉夜她是秀知院的学生会会长,最近跟会长做了一些不可描述的事情,我没有说是什麼事情,所以也不算是爆雷,但是我们知道说辉夜,其实还有另外一个成就,其实虽然说辉夜姬,这个动画还没有完结,但是他的后传其实已经演完了,后来白银英年早逝所以辉夜,就把自己的头髮染白,她為了想要离开伤心的地球,就坐著太空船到另外一个星球,另外一个星球也有一些原始的生命,那些生命跟人类长得也是挺像的,他们还过著农耕的生活,然后辉夜看到其中一个小国的国王,长得跟白银有点像,所以她就跟那个小国国王在一起,她们生下了一个小孩就是六道仙人,於是就开创了忍者时代,真是可喜可贺 可喜可贺。所以我们今天学到什麼事,我们今天就是学到说,这个辉夜大小姐这个动画,它真是一个了不起的动画,它其实是在两部大长篇中间的小品,它既是猎人的后传,也是火影忍者的前传,这三个故事是可以串在一起的,就是這麼回事

所以我们对机器说一句话,问它说辉夜是谁,其实每个人也可能都有不同的答案,这个时候我们就需要,generative的model

Generative Adversarial Network (GAN)

generative的model,其中一个非常知名的,就是generative adversarial network,它的缩写 是GAN,那我们这一堂课主要就是介绍,generative adversarial network,发音就是 gàn

它其实有很多各式各样的变形,你可以在网路上找到,一个GAN的[动物园](#),找到一个GAN的zoo

All Kinds of GAN ... <https://github.com/hindupuravinash/the-gan-zoo>

GAN	<ul style="list-style-type: none"> • SeUDA - Semantic-Aware Generative Adversarial Nets for Unsupervised Domain Adaptation • SG-GAN - Semantic-aware Grad-GAN for Virtual-to-Real Urban Scene Adaption (github) • SG-GAN - Sparsely Grouped Multi-task Generative Adversarial Networks for Facial Attr. • SGAN - Texture Synthesis with Spatial Generative Adversarial Networks • SGAN - Stacked Generative Adversarial Networks (github) • SGAN - Steganographic Generative Adversarial Networks • SGAN - SGAN: An Alternative Training of Generative Adversarial Networks • SGAN - CT Image Enhancement Using Stacked Generative Adversarial Networks and Tr Segmentation Improvement • sGAN - Generative Adversarial Training for MRA Image Synthesis Using Multi-Contrast • SiftingGAN - SiftingGAN: Generating and Sifting Labeled Samples to Improve the Remote Classification Baseline in vitro • SiGAN - SiGAN: Siamese Generative Adversarial Network for Identity-Preserving Face H • SimGAN - Learning from Simulated and Unsupervised Images through Adversarial Trai • SisGAN - Semantic Image Synthesis via Adversarial Learning
ACGAN	
BGAN	
CGAN	
DCGAN	
EBGAN	
fGAN	
GoGAN	
...	

Mihuela Rosca, Balaji Lakshminarayanan, David Warde-Farley, Shakir Mohamed, "Variational Approaches for Auto-Encoding Generative Adversarial Networks", arXiv, 2017

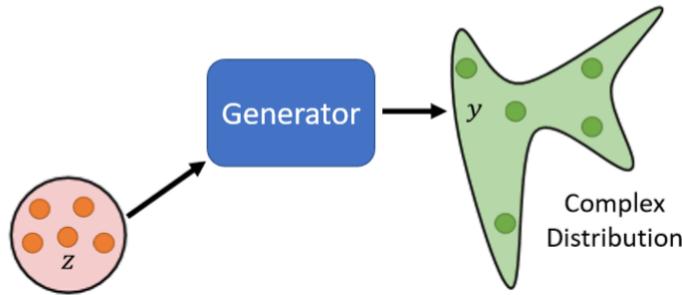
²We use the Greek α prefix for α -GAN, as AEGAN and most other Latin prefixes seem to have been taken <https://deephunt.in/the-gan-zoo-79597dc8c347>.

那个GAN的动物园裡面,收集了超过五百种以上的GAN,每次有人发明了,一个新的GAN的时候,他就会在前面加一个英文的字母,但是英文的字母是有限的,很快的英文的字母就被用尽了

举例来说在GAN的动物园裡面,至少就有六种的SGAN,它们都是不同的东西,但它们通通被叫做SGAN,甚至还发生了的状况,有一篇paper他提出来的叫做,"Variational auto-encoding GAN",照理说应该所写成,AEGAN或者是AGAN,但是作者加了一个註解说,哎呀AEGAN被别人用了,所有的英文字母,看起来都被别人用了,我只好叫做 α -GAN

我们现在要举的例子,就是要让机器生成动画人物的,二次元人物的脸,我们举的例子是Unconditional的 generation,unconditional generation,就是我们这边先把X拿掉

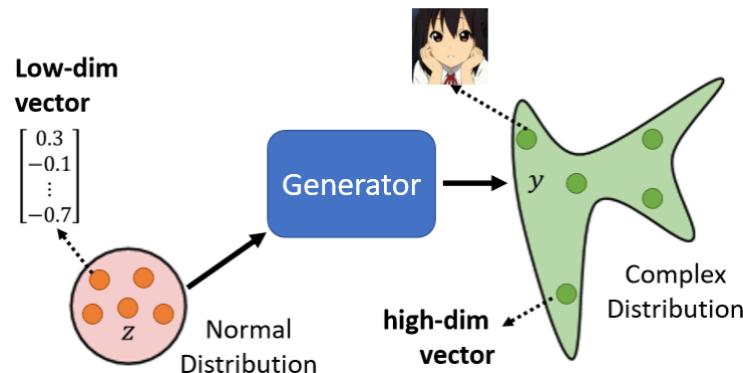
- Unconditional generation



那之后我们在讲到conditional, generation的时候,我们会再把X加回来,这边先把X拿掉,所以我们的 generator它输入就是Z,它的输出就是Y

那输入的这个Z是什麼

- Unconditional generation



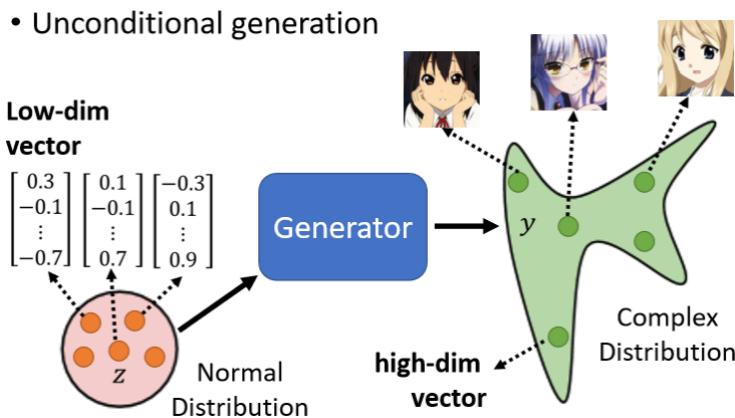
我们都假设Z是从一个normal distribution里sample出来的向量,那这个向量通常会是一个,low-dimensional的向量,它的维度其实是你自订的,你自己决定的,那通常你就订个50, 100,的大小,它是你自己决定的

好你从这边Z,你从这个normal distribution,裡面sample一个向量以后,丢到generator裡面,Generator就给你一个对应的输出,那我们希望对应的输出,就是一个二次元人物的脸

那到底generator要输出,怎麽样的东西,才会变成一个二次元人物的人脸,其实这个问题没有你想像的那麼困难

一张图片就是一个非常高维的向量,所以generator实际上做的事情,就是產生一个非常高维的向量,举例来说 假设这是一个64X64,然后彩色的图片,那你的generator输出就是64X64X3,那麼长的向量 把那个向量整理一下,就变成一张二次元人物,这个就是generator要做的事情

当你输入的向量不同的时候,你的输出就会跟著改变,所以你从这个,normal distribution裡面,Sample Z出来 Sample到不同的Z,那你输出来的Y都不一样,那我们希望说不管你这边sample到什麼Z,输出来的都是动画人物的脸



那讲到这边可能有同学会问说,这边為什麼是normal distribution,不能是别的吗?

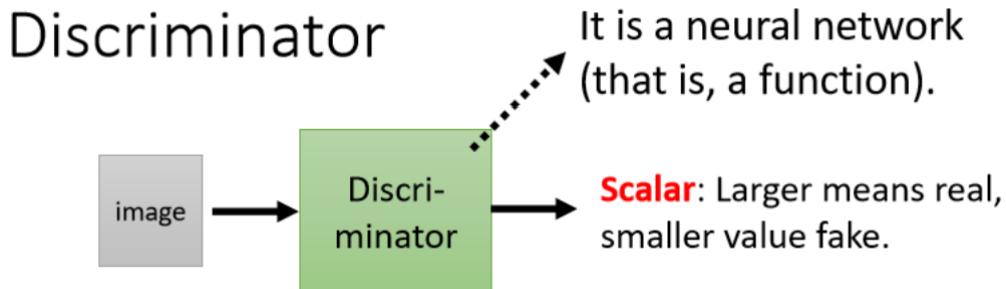
可以是别的,这边选别的你其实也会问同样的问题,就是了,那我(李宏毅本人)的经验是不同的distribution之间的差异,可能并没有真的非常大,不过你还是可以找到一些文献,试著去探讨不同的distribution之间,有没有差异

但是这边其实你只要选一个,够简单的distribution就行,因為你的generator会想办法,把这个简单的distribution,对应到一个复杂的distribution,所以你可以把选择,distribution这件事情,交给你的generator来处理,那这边我们在等一下的讨论裡面,都假设是一个,normal distribution

Discriminator

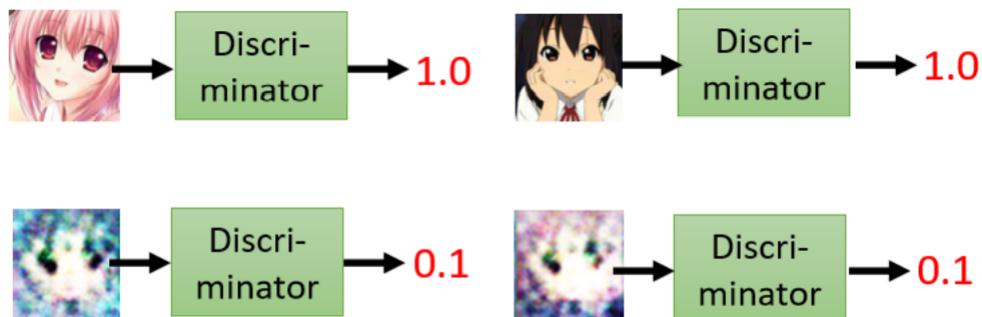
在GAN裡面,一个特别的地方就是,除了generator以外,我们要多训练一个东西,叫做discriminator

discriminator它的作用是,它会拿一张图片作为输入,它的输出是一个数值,这个discriminator本身,也是一个neural network,它就是一个function



它输入一张图片,它的输出就是一个数字,它输出就是一个scalar,这个scalar越大就代表说,现在输入的这张图片,越像是真实的二次元人物的图像

举例来说



这个是二次元人物的头像,那就输出1 假设1是最大的值,那这个也是画得很好的就输出1,这个不知道在画什麼就输出0.1,这个不知道在画什麼就输出0.1

所以generator,它是个neural network,Discriminator,也是个neural network,他们的架构长什麼样子,你完全可以自己设计,你可以用CNN 你可以用,transformer 都可以,只要你能够產生出你要的输入输出,就可以了

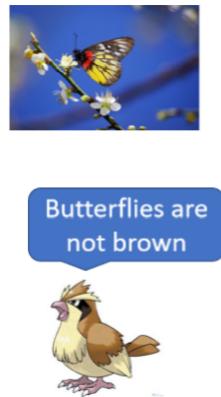
那在这个例子裡面,像discriminator,因為输入是一张图片,你很显然会选择CNN对不对,CNN在处理影像上有非常大的优势,既然输入是一张图片,那你的discriminator很有可能,裡面会有大量的CNN的架构,那至於实际上要用什麼样的架构,完全可以自己决定

Basic Idea of GAN

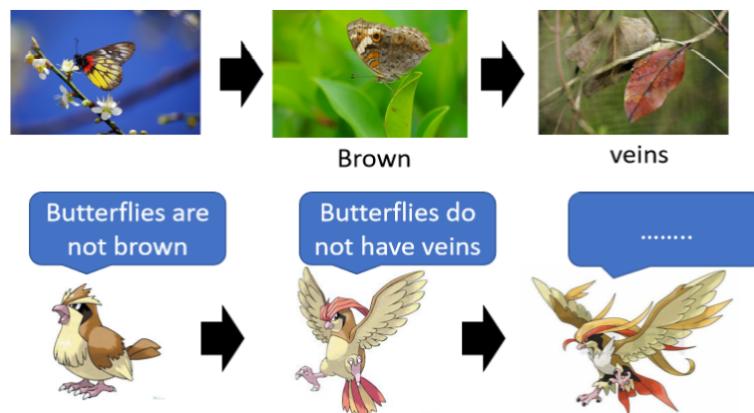
為什麼要多一个discriminator,这边就讲一个故事,这个故事跟演化是有关的



这不是一片枯叶,它其实枯叶蝶的拟态,那枯叶蝶长得跟枯叶非常像,它可以躲避天敌,那枯叶蝶的祖先,其实也不是长得像枯叶一样,也许他们原来也是五彩斑斓,但為什麼他们变得长得像枯叶一样,是因为有天择的压力



这个不是普通的麻雀 这个是波波,这个波波会吃枯叶蝶的祖先,在天择的压力之下,枯叶蝶就变成棕色的
因为波波它只会吃彩色的东西,它看到彩色的东西知道是蝴蝶,就把它吃掉,那看到棕色的东西,那个波波就觉得是枯叶就可以骗过它,所以枯叶蝶的祖先,在天择的压力之下,颜色就变成是棕色的

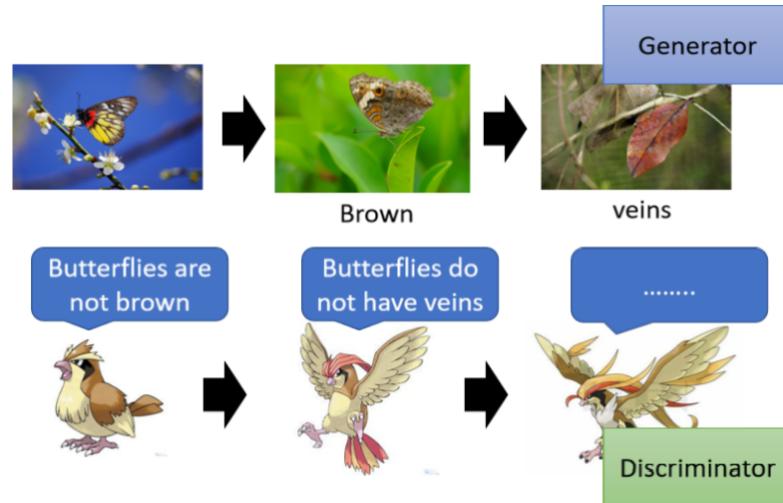


但是波波也是会演化的,所以波波為了要吃到这些枯叶蝶,你有偽裝成枯叶的枯叶蝶,所以它也进化了,波波进化以后就是比比鸟这样

比比鸟,它在判断一个蝴蝶能不能吃的时候,是用比较高明的手段,它不会只看顏色 它会看它的纹路,它知道说没有叶脉的是蝴蝶,有叶脉的才是真正的枯叶

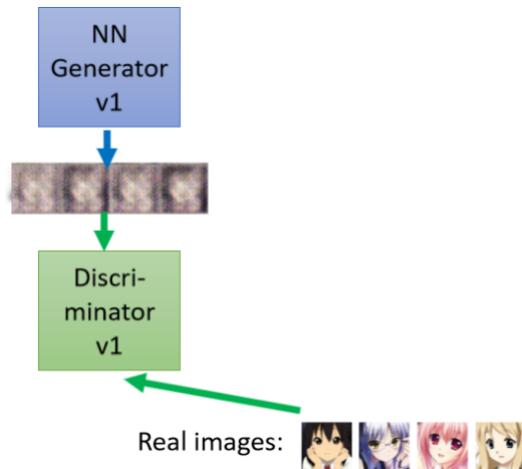
在天择的压力之下,枯叶蝶就產生了拟态 產生了叶脉,想要骗过比比鸟,但是比比鸟它也有可能会再进化,比比鸟进化是什麼,比比鸟进化就是大比鸟

这个就是大比鸟,那大比鸟可能可以分辨,这个枯叶蝶跟枯叶的不同



那这个是演化的故事,对应到GAN 枯叶蝶就是generator,那它的天敌就是discriminator,

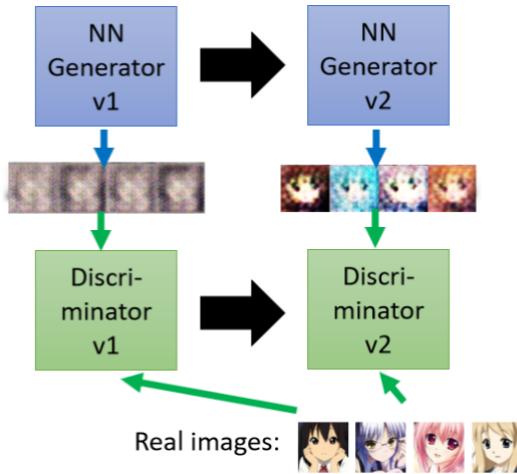
那现在我们generator要做的事情,是画出二次元的人物,那generator怎麼学习画出二次元的人物,它学习的过程是这样子



第一代的generator它的参数几乎是,它的参数完全是随机的,所以它根本就不知道,要怎麼画二次元的人物,所以它画出来的东西就是一些,莫名其妙的杂讯

那discriminator接下来,它学习的目标是,要分辨generator的输出,跟真正的图片的不同,那在这个例子裡面可能非常的容易,对discriminator来说它只要看说,图片裡面有没有两个黑黑的圆球,就是眼睛,有眼睛就是真正的二次元人物,没有眼睛就是generator,產生出来的东西

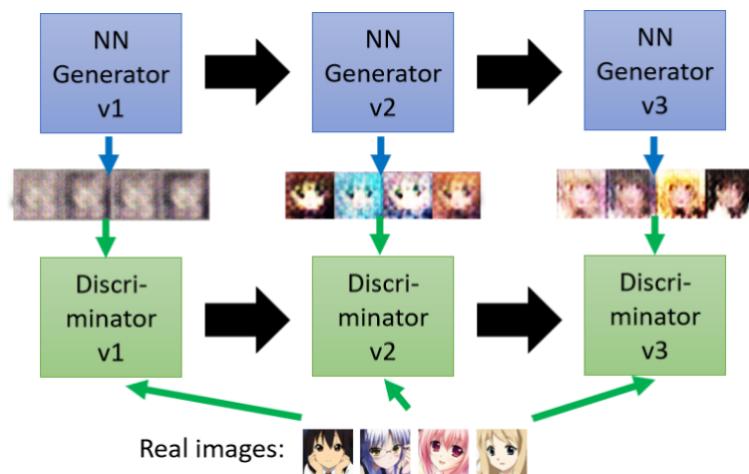
接下来generator就调整它的裡面的参数,Generator就进化了,它调整它裡面的参数 它调整的目标,是为了要骗过discriminator,假设discriminator,判断一张图片是不是真实的依据,看的是有没有眼睛,那generator就產生眼睛出来,给discriminator看



所以generator產生眼睛出来,然后他可以骗过第一代的discriminator,但是discriminator也是会进化的,所以第一代的discriminator,就变成第二代的discriminator,第二代的discriminator,会试图分辨这一组图片,跟真实图片之间的差异,它会试图去找出这两者之间的差异

它发现说,这边產生的图片都是比较简单的,举例来说都没有头髮也没有嘴巴,那这些图片是有头髮的也有嘴巴

接下来第三代的generator,就会想办法,去骗过第二代的discriminator,既然第二代的discriminator是看,有没有嘴巴来判断是不是真正的二次元人物,那第三代的generator就会把嘴巴加上去



那discriminator也会逐渐的进步,它会越来越严苛,然后期待discriminator越来越严苛,Generator產生出来的图片,就可以越来越像二次元的人物,那因为这边有一个generator,有一个discriminator,它们彼此之间是会互动

最早这个GAN是,Ian Goodfellow propose的,14年这个GAN的paper,是发表在14年的arxiv,那最早在这个GAN的原始的,paper裡面,把generator跟discriminator,当作是敌人

如果你有看很多网路文章的话,它都会举例说,啊generator是假钞的啊,然后discriminator是警察啊,警察要抓做假钞的人啊,假钞就会越来越像,警察就会越来越厉害等等

因为觉得generator,跟discriminator中间有一个,对抗的关係,所以就用了一个,adversarial这个字眼,Adversarial就是对抗的意思,但是至於generator跟discriminator,他们是不是真的在对抗,这只是一个拟人化的说法而已,

所以generator,跟discriminator的关係啊,用动画来说就是写作敌人唸做朋友,就跟进藤光还有塔矢亮一样,或者是跟Naruto跟Sasuke一样



Algorithm

以下就是正式来讲一下,这个演算法实际上是长什麼样子,generator跟discriminator,他们就是两个network

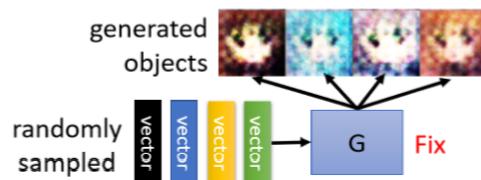
network在训练前,你要先初始化它的参数,所以我们这边就假设说,generator跟discriminator,它们的参数 都已经被初始化了

Step 1: Fix generator G, and update discriminator D

初始化完以后,接下来训练的第一步是,定住你的generator,只train你的discriminator

- Initialize generator and discriminator G D
- In each training iteration:

Step 1: Fix generator G, and update discriminator D



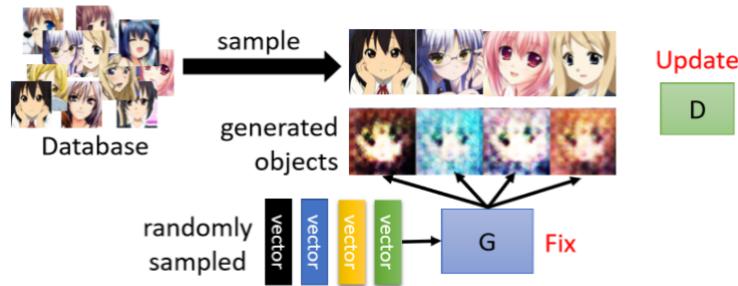
那因為一开始你的generator的参数,是随机初始化的,那如果你又固定住你的generator,那它根本就什麼事都没有做啊,它的参数都是随机的啊

所以你丢一堆向量给它,它的输出都是乱七八糟的图片,那其实如果generator参数,是初始化的话,你连这样子的结果都產生不出来,那產生出来的就很像是电视机坏掉的,那一种杂讯

那你从这个gaussian distribution裡面,去random sample一堆vector,把这些vector丢到generator裡面,它就吐出一些图片 一开始这些图片,会跟正常的二次元人物非常的不像

好那你会有一个database,这个database裡面,有很多二次元人物的头像,这个去网路上爬个图库就有了,这个不难蒐集,从这个图库裡面,去sample一些,二次元人物的头像出来

Step 1: Fix generator G, and update discriminator D

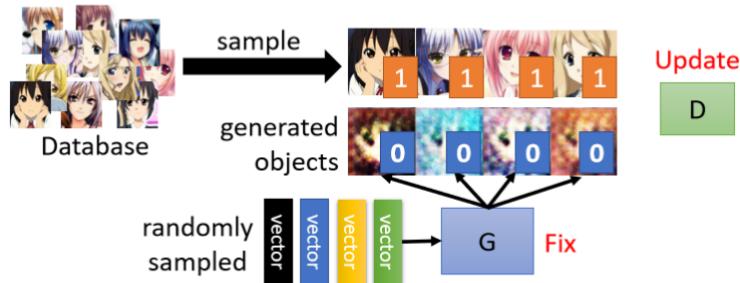


Discriminator learns to assign high scores to real objects and low scores to generated objects.

接下来你就拿真正的二次元人物头像,跟generator產生出来的结果,去训练你的discriminator,discriminator它训练的目标是要分辨,真正的二次元人物,跟generator產生出来的二次元人物,它们之间的差异

讲得更具体一点啊,你实际上的操作是这个样子,你可能会把这些真正的人物都标1,Generator產生出来的图片都标0

Step 1: Fix generator G, and update discriminator D



Discriminator learns to assign high scores to real objects and low scores to generated objects.

接下来对於discriminator来说,这就是一个分类的问题,或者是regression的问题

- 如果是分类的问题,你就把真正的人脸当作类别1,Generator產生出来的,这些图片当作类别2,然后训练一个classifier就结束了
- 或者是有人会把它当作,regression的问题,那你就教你的discriminator说,看到这些图片你就输出1,看到这些图片你就输出0,都可以 总之discriminator就学著,去分辨这个real的image,跟產生出来的image之间的差异

但是实际上怎麽做,你可以当作分类的问题来做,也可以当作regression的问题来做

Step 2: Fix discriminator D, and update generator G

我们训练完,discriminator以后,接下来定住discriminator改成训练generator,怎麽训练generator呢

拟人化的讲法是,我们就让generator想办法去骗过discriminator,因为刚才discriminator,已经学会分辨,真图跟假图的差异,真图跟生成的图片的差异,Generator如果可以骗过,discriminator它可以产生一些图片,Discriminator觉得,是真正的图片的话,那generator产生的图片,可能就可以以假乱真

Generator learns to “fool” the discriminator



它实际上的操作方法是这样子,你有一个generator,generator吃一个向量作為输入,从gaussian distribution sample,出来的向量作為输入,然后產生一个图片

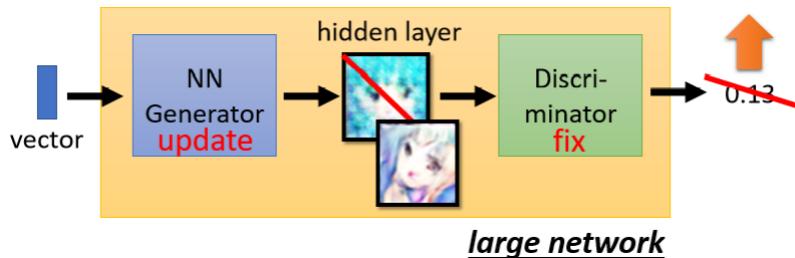
接下来我们把这个图片丢到,Discriminator裡面,Discriminator会给这个图片,一个分数,那generator它训练的目标,就Discriminator参数是固定的,我们只会调整generator的参数



Generator训练的目标,是要Discriminator的输出值,越大越好,那因為Discriminator,它本来训练的时候,它训练的目标它可以做的事情就是,看到好的图片就给它大的分数,如果generator可以调整参数之后,输出来的图片Discriminator,会给予高分,那意味著generator產生出来的图片,是比较真实的

得更具体一点,实际上你的操作是这个样子,Generator是一个network裡面有好几层,Discriminator也是一个, network裡面有好几层,我们把generator跟Discriminator直接接起来,当做一个比较大的network来看待

举例来说generator,如果是五层的network,Discriminator如果是五层的network,把它们接起来我们把它当作是一个十层的network来看待



而这个十层的network裡面,某一个hidden layer它的输出很宽,它的输出的这个dimension呢,就跟图片裡面pixel的数目,乘三是一样的,你把这个hidden layer的输出呢,做一下整理以后就会变成一张图片,所以这整个大的network裡面,其中某一层的输出就是代表一张图片

我们要做的事情是,整个巨大的network啊,它会吃一个向量作為输入,然后他会输出一个分数,那我们希望调整这个network,让输出的分数越大越好

但是要注意一下 我们不会去调,对应到Discriminator的部分,我们不会去调这个巨大, network的最后几层,為什麼不调最后几层呢

你可以想想看假设调最后几层的话,这整个游戏就被hack了,因为假设你要输出的分数越大越好,我直接调最后output layer,那个neural bias,把它设成一千万,那不是输出就很大了吗,所以Discriminator这边的参数,是不能动的,

我们只调generator的参数,好那这边呢,至於怎麼调Generator的参数呢,这个训练的方法啊,跟我们之前训练一般的network,是没有什麼不同的

我们之前说训练network的时候就是,定一个loss啊 然后你用gradient descent,让loss越小越好,那这边呢你也有一个目标,只是这个目标呢不是越小越好,而是越大越好,那当然你也可以把这个目标,Discriminator output成一个负号,就当作loss你可以把Discriminator,output成一个负号当作loss,然后generator训练的目标,就是让loss越小越好

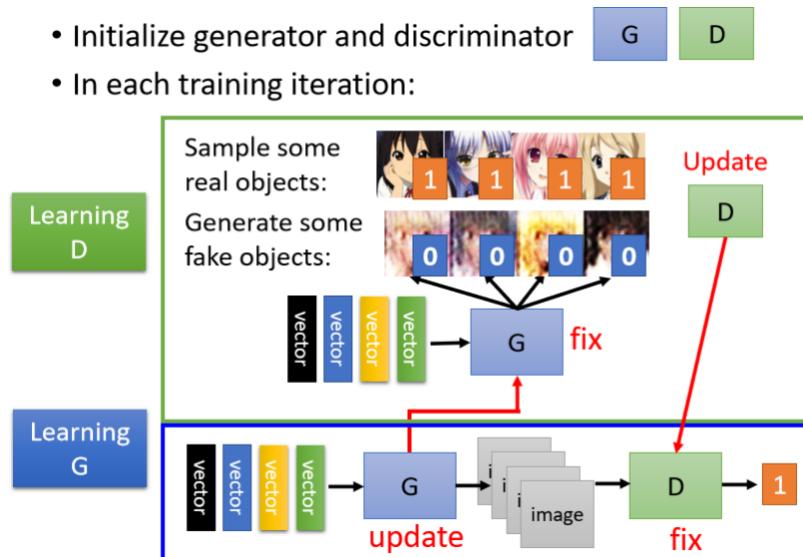
或者你也可以说,我们就是要让Discriminator output,的值越大越好,然后我们用gradient ascent,不是gradient descent,gradient descent是,让loss越小越好,gradient ascent会让你的目标函数,越大越好,我们会用gradient ascent去调generator,让Discriminator的输出越大越好

这是同一件事,这边训练generator的方法,也是用gradient descent base的方法,跟我们之前在训练一个,一般network的时候,是没有什麼差异的

所以现在讲了两个步骤

- 第一个步骤 固定generator,训练discriminator
- 第二个步骤,固定discriminator训练generator

接下来就是反覆的训练,discriminator跟generator,训练完discriminator以后,固定住discriminator,训练generator,训练完generator以后,再用generator去产生更多的,新的产生的出来的图片,再给discriminator做训练,训练完discriminator以后,再去训练generator,反覆的去执行



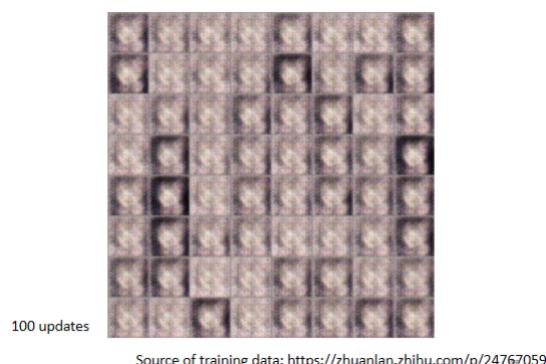
所以你是训练一阵子discriminator,训练一阵子generator,训练一阵子discriminator,再训练一阵子generator,Generator跟discriminator,它们是反覆的去进行训练,当其中一种进行训练的时候,另外一个就固定住,那你期待discriminator跟generator,都可以做得越来越好,

GAN for Images Generation——Anime Face Generation

下一个作业就是,要做动画人物的人脸生成,那你可能会问说,到底可以做到什麼样的程度呢

以下的结果是我在17年的时候做的 Source of training data: <https://zhuanlan.zhihu.com/p/24767059>,我自己试著train了一下GAN,看看GAN是不是真的可以产生,二次元的人物

好那我训练了,我把那个generator呢,Update了一百次以后,所谓generator update一百次,的意思是说,就是discriminator train一下,generator train一下,discriminator train一下,generator train一下,这样往返一百次以后得到的结果,是这样子



嗯 不知道在做些什麼,但我接下来呢就再等了一下,Train一千次的



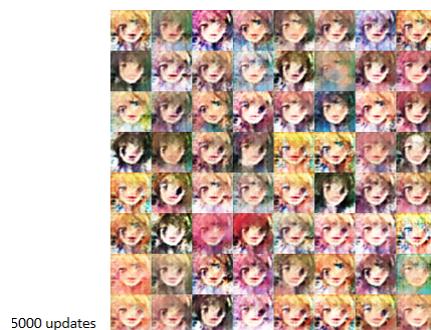
1000 updates

discriminator 跟generator,各自训练这样反覆一千次以后,机器就產生了眼睛,机器知道说 人脸就是要有两个眼睛,所以它就把眼睛标上去,训练到两千次的时候,你发现嘴巴就出来了



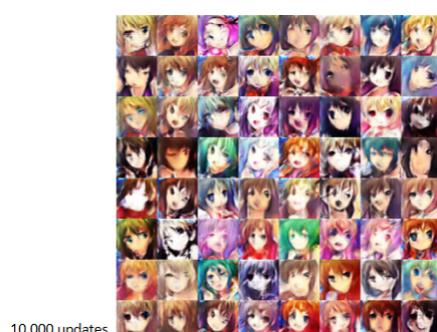
2000 updates

训练到五千次的时候,已经开始有一点人脸的样子了,而且你发现说机器学到说,动画人物啊,就是要有那个水汪汪的大眼睛,所以他每个人的眼睛呢,都涂得非常的大,涂有反白 代表说反光,是水汪汪的大眼睛



5000 updates

这个是训练一万次以后的结果,有发现形状已经有出来了,只是有点模糊,很多地方有点晕开的感觉,好像是水彩画的样子,

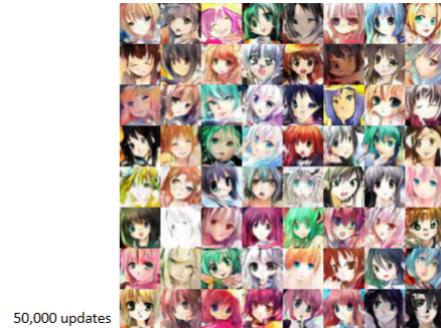


10,000 updates

接下来这个是,update两万次的结果



这个是update五万次的结果



我后来就停在五万次的地方,那其实你在作业裡面,是有机会做得比这个结果更好的,这个是助教是学生的时候做的结果啦,那如果是最好,可能可以做到这个样子



那你会发现说这些人物呢都还不错,只是有一些比较,还是会有偶尔会有一些崩坏啦,但乍看之下呢可能比一些作画画风,会崩坏的动画公司,比如说一些妹非妹做的还要好一些了,

如果你有好的资料库的话,那当然我们提供给大家的资料,是做不到这个地步的啦,如果你有真的非常好的资料的话,也许你可以做出真的很好的结果

<https://www.gwern.net/images/gan/stylegan/2019-02-11-stylegan-danbooru2017faces-interpolation.mp4>

我在网路上呢,找到了一个这样子的结果,这个是用StyleGAN做的,那用StyleGAN做起来,可以做到这个样子



我觉得非常惊人喔,很惊人喔 这些都是,用GAN產生出来的人物,这边他还產生了异色瞳,我不知道算是画错呢还是它特意呢,要產生异色瞳,对异色瞳,就一眼白眼一眼血轮眼这样子的概念,

好那除了產生动画人物以外,当然也可以產生真实的人脸,有一个技术叫做progressive GAN,它可以產生非常高清的人脸



Progressive GAN |

<https://arxiv.org/abs/1710.10196>

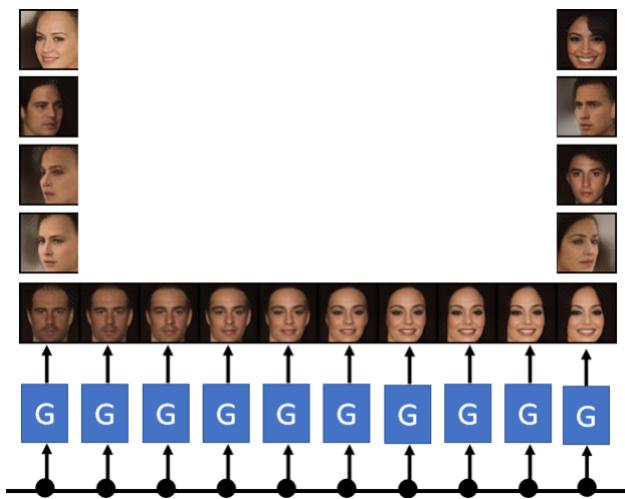
上下两排都是机器產生的,好所以这个显然progressive GAN,它有办法產生以假乱真的人脸

甚至之前啊 你不知道我有听,有一个新闻我不知道是不是真的,有一个新创公司 它裡面有很多人,但大家发现裡面那些人的头像,有点怪怪的,有人说那些头像其实,是用GAN生成的,那并不是真正的人物,那个公司没有那麼多人,用GAN呢生成一些假人的照片,当作是假员工,

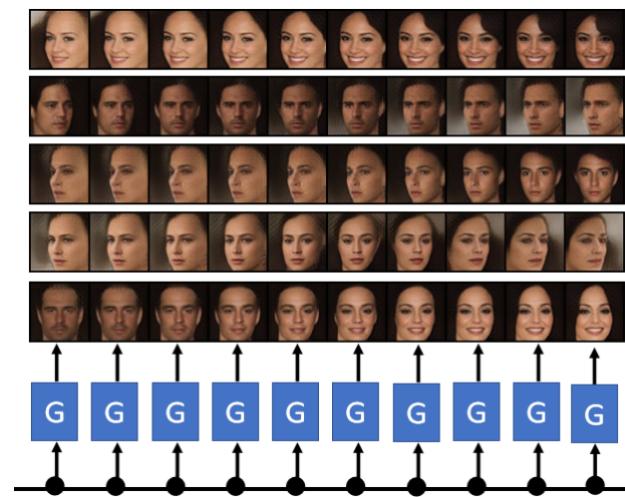
那你可能会问说要產生人脸,有什麼用呢 我去路边拍一个人,產生出来的照片不是更像真的吗

但是用GAN你可以產生,你没有看过的人脸,举例来说用GAN,你可以做到这样子的事情,我们刚才说GAN这种generator,就是输入一个向量 输出一张图片,那你不只可以输入一个向量,输出一张图片

你还可以把输入的向量,做内插 做interpolation,把输入的向量做内插以后,会发生什麼事呢,你就会看到两张图片之间连续的变化



举例来说你输入一个向量,这边產生一个看起来非常严肃的男人,你输入一个向量,这边產生一个笑口常开的女人,那你输入这两个向量中间的,interpolation它的内插,你就看到这个男人逐渐的笑了起来,或者是呢这边有更多的例子



你输入一个向量,这个输入的向量这边是假的啦,但这边產生出来的图片是真的,你输入一个向量,这边產生一个往左看的人,你输入一个向量,这边產生一个往右看的人,你把往左看的人跟往右看的人,做interpolation会发生什麼事呢

机器并不是傻傻地,把两张图片叠在一起,变成一个双面人,而是机器知道说,往左看的人脸跟往右看的人脸,介於他们中间的就是往正面看,你在训练的时候其实并没有真的告诉,机器这件事 但机器可以自己学到说,把这两张脸做内插,应该会得到一个往正面看的人脸,

刚才已经讲过说,GAN是Ian Goodfellow,在14年的时候提出来的,你可能有听过那个故事是,这不知道是不是真的啦一个传说,Ian Goodfellow去酒吧,看到两个人吵架,於是就有了GAN的灵感,然后呢回家第一次实作,就成功了这样

然后就结束,然后就投了一个paper,那但是他所谓的成功啊,其实是长这个样子的



在14年的时候,我第一看到这个结果的时候,我觉得哇靠还真的可以產生图片,太厉害了,当然如果从今天的角度来看,你会觉得说 这样你也算是有成功吗,今天比如说你用, BigGAN產生出来的图片,可以做到像这个样子



这些图片都是机器生成的,当然仔细看一下,还是可以发现一些破绽,举例来说这隻狗 它多了一个脚啊,或者是这个杯子,它左右没有很对称啊,它有点歪歪的,但这些图片都是机器生成的,那有时候机器,也会產生一些幻想中的角色,举例来说机器就產生了一个网球狗啊,



GAN_Theory behind GAN

P1是用了一堆比喻说明GAN 的操作是怎麽进行的

接下来,我们告诉你说实际上,為什麼这个 Generator 跟 Discriminator 的互动,可以让我们的 Generator 產生像是真正的人脸的图片,那这背后的互动在做的,到底是什麼样的事情

那我们先来弄清楚,我们今天的训练的目标到底是什麼

我们在训练 Network 的时候,你就是要

- 定一个 Loss Function
- 定完以后用 Gradient Descent 去调你的参数
- 去 Minimize 那个 Loss Function 就结束了

在这个 Generation 的问题裡面,我们要 Minimize,或者是 Maximize 什麼样的东西,我们要把这些事弄清楚,才能够做接下来的事情

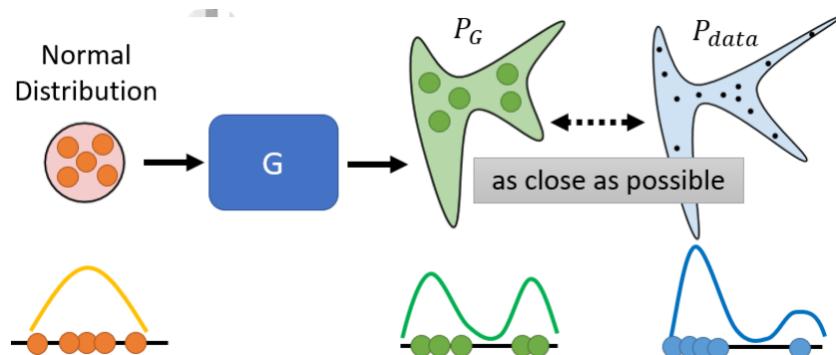
我们想要 Minimize 的东西是这个样子的, 我们有一个 Generator

- 给它一大堆的 Vector,给它从 Normal Distribution Sample 出来的东西
- 丢进这个 Generator 以后,会产生一个比较复杂的 Distribution,这个复杂 Distribution,我们叫它 PG
- 然后我们有一堆的 Data,这个是真正的 Data,真正的 Data 也形成了另外一个 Distribution,叫做 Pdata,我们期待 PG 跟 Pdata 越接近越好

Optimization

Our objective

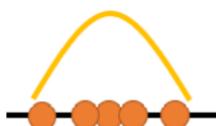
如果你一下子没有办法想像,这个 PG 、 Pdata 是怎麼一回事的话,那我们用一维的状况来跟大家说明



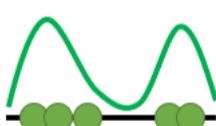
我们假设

- Generator 的 Input 是一个一维的向量
- Generator 的 Output 也是一维的向量
- 我们真正的 Data 也是一维的向量

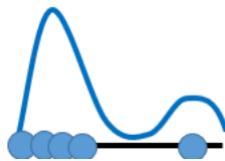
那我们的 Normal Distribution 就长这个样子



那丢到 Generator 以后,假设你输入 5 个点,那边这每一个点,它的位置会改变,那你就產生一个新的 Distribution,那可能本来大家都集中在中间,通过这个 Generator,通过一个 Network 裡面很复杂,不知道做了什麼事情以后,这些点就分成两边,所以你的 Distribution 就变成这个样子



而 P_{data} 是指真正的资料的分布,真正资料分布可能长这个样子



它分两面的状况是更极端的,左边的东西比较多,右边的东西比较少,那你期待左边这个分布跟右边这个分布,越接近越好,如果写成式子的话

你可以写成这个样子

$$G^* = \arg \min_G \underline{Div(P_G, P_{data})}$$

Divergence between distributions P_G and P_{data}

Div Of PG 跟 Pdata,它指的意思就是 PG 跟 Pdata,这两个 Distribution 之间的 Divergence(散度)

Divergence 这边指的意思就是,这两个你可以想成是,这两个 Distribution 之间的某种距离

- 如果这个 Divergence 越大,就代表这两个 Distribution 越不像
- Divergence 越小,就代表这两个 Distribution 越相近

Divergence 这就是衡量,两个的 Distribution 相似度的一个 Measure

我们现在的目标,是要去找一个 Generator,找一个 Generator 找个操作,实际上做的事情是找一个 Generator 裡面的参数,Generator 也是一个 Network,裡面有一大堆的 Weight 跟 Bias,它可以让我们產生出来的 PG 跟 Pdata,之间的 Divergence 越小越好,我要找的就是这样子的 Generator

这边把它写作 G^* ,所以我们这边要做的事情,跟一般 Train Network 其实非常地像,我们第一堂课就告诉你说,我们定义了 Loss Function,找一组参数 Minimize Loss Function

在 Generation 这个问题裡面,我们现在其实也定义了我们的 Loss Function,它就是 PG 跟 Pdata 的 Divergence,就是它们两个之间的距离,它们两个越近,那就代表这个產生出来的 PG 跟 Pdata 越像,所以 PG 跟 Pdata,我们希望它们越相像越好

但是我们这边遇到一个困难的问题

$$G^* = \arg \min_G \underline{Div(P_G, P_{data})}$$

Divergence between distributions P_G and P_{data}

How to compute the divergence?

这个 Loss,我们是可以算的,但是这个 Divergence 是要怎麼样算? 那你可能知道一些 Divergence 的式子,比如说 KL Divergence,比如说 JS Divergence,这些 Divergence 用在这种 Continues 的,Distribution 上面,你要做一个很复杂的,在实作上你几乎不知道要怎麼算的积分,那我们根本就无法把这个 Divergence 算出来

我们算不出这个 Divergence,我们又要如何去找一个 G,去 Minimize 这个 Divergence,这个就是 GAN 所遇到的问题,这就是我们在 Train 这一种,Generator 的时候会遇到的问题

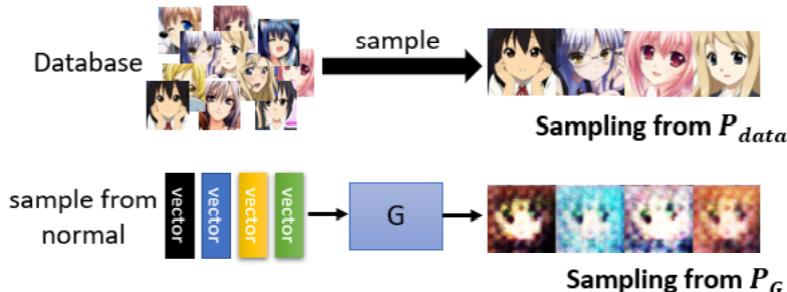
而 GAN是一个很神奇的做法,它可以突破,我们不知道怎麼计算 Divergence 的限制

Sampling is good enough

所以我现在遇到的问题就是,不知道怎麼计算 Divergence,而 GAN 告诉我们就是,你不需要知道 PG 跟 Pdata 它们实际上的 Formulation 长什麼样子,只要能从 PG 和 Pdata 这两个 Distributions Sample 东西出来,就有办法算 Divergence

$$G^* = \arg \min_G \text{Div}(P_G, P_{\text{data}})$$

Although we do not know the distributions of P_G and P_{data} , we can sample from them.



怎麼从真正的 Data 裡面,Sample 出东西来, 怎麼从 Generator 裡面,產生一些东西出来?

- 把你的图库拿出来,从图库裡面随机產生,随机 Sample 一些图片出来,你就得到 Pdata
- 那你就把你的 Generator,输入这个 Normal 的,从 Normal Distribution Sample 出来的 Vector,丢到 Generator 裡面 (我们刚才说过,你这边的拿来 Sample 那个 Distribution,要是你有办法 Sample 的,所以我们选 Normal Distribution 式子我们是知道的,是有办法 Sample 的) 我们从 Normal Distribution 裡面,Sample 一堆 Vector 出来丢给 Generator,让 Generator 產生一堆图片出来,那这些图片就是从 PG Sample 出来的结果

所以我们有办法从 PG 做 Sample,我们有办法从 Pdata 做 Sample

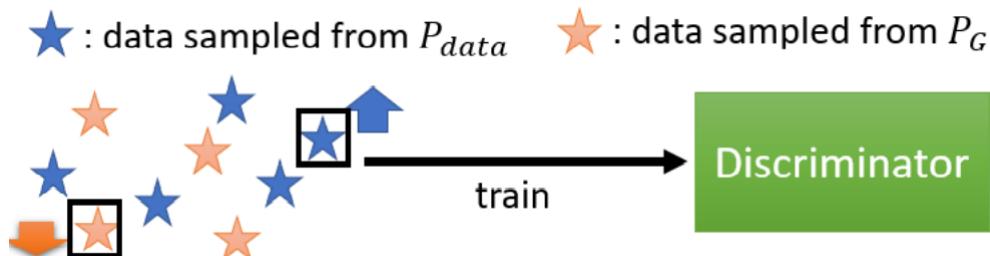
Discriminator

接下来,GAN 这一整个系列的 Work,就是要告诉你说,怎麼在只有做 Sample 的前提之下,我根本不知道 PG 跟 Pdata,实际上完整的 Formulation 长什麼样子,居然就估测出了 Divergence

那这个就是要靠 Discriminator 的力量

那我们刚才讲过说 Discriminator 是怎麽训练出来的

- 我们有一大堆的 Real Data,这个 Real Data 就是从 Pdata Sample 出来的结果
- 我们有一大堆 Generative 的 Data,Generative 的 Data,就可以看作是从 PG Sample 出来的结果



根据 Real 的 Data 跟 Generative 的 Data,我会去训练一个 Discriminator,它的训练的目标是

- 看到 Real Data,就给它比较高的分数
- 看到这个 Generative 的 Data,就给它比较低的分数

Discriminator 训练的目标,就是要分辨好的图跟不好的图,分辨真的图跟生成的图,所以看到真的图给它高分,看到生成的图给它低分

实际以上的过程,你也可以把它写成式子,把它当做是一个 Optimization 的问题,这个 Optimization 的问题是这样子的

$$\text{Training: } D^* = \arg \max_D V(D, G)$$

Objective Function for D

$$V(G, D) = E_{y \sim P_{data}}[\log D(y)] + E_{y \sim P_G}[\log(1 - D(y))]$$

这个 Discriminator 可以去 Maximize 某一个 Function,我们这边叫做 Objective Function (**我们要 Maximize 的东西,我们会叫 Objective Function,如果 Minimize 我们就叫它 Loss Function**)

我们现在要找一个 D,它可以 Maximize 这个 Objective Function,这个 Objective Function 长这个样子

- $E_{y \sim P_{data}}[\log D(y)]$ 我们有一堆 Y,它是从 Pdata 裡面 Sample 出来的,也就是它们是真正的 Image,而我们把这个真正的 Image 丢到 D 裡面,得到一个分数再取 $\log D(y)$
- $E_{y \sim P_G}[\log(1 - D(y))]$ 那另外一方面,我们有一堆 Y,它是从 PG 从 Generator 所產生出来的,把这些图片也丢到 Discriminator 裡面,得到一个分数,再取 $\log(1 - D(y))$

我们希望这个 Objective Function V 越大越好

Objective Function for D

$$V(G, D) = E_{y \sim P_{data}}[\log D(y)] + E_{y \sim P_G}[\log(1 - D(y))]$$

- 意味著我们希望这边的 D(Y) 越大越好,我们希望 Y 如果是从 Pdata Sample 出来的,它就要越大越好
- 我们希望说如果 Y 是从这个 PG Sample 出来的,它就要越小越好

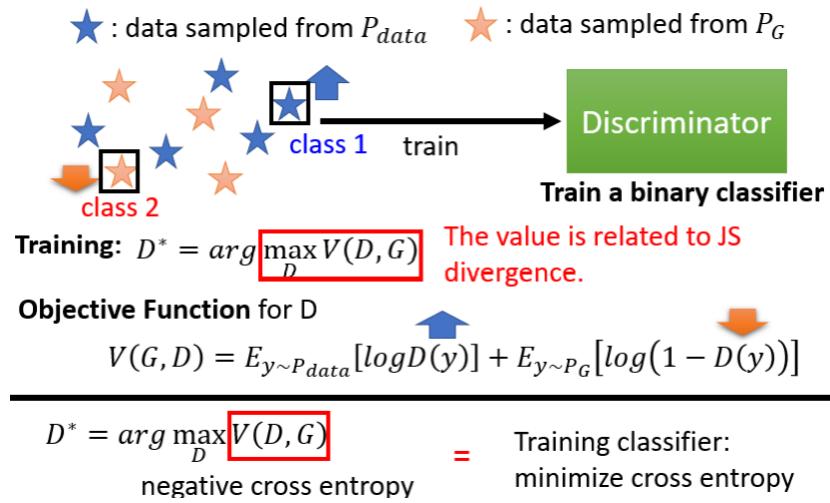
让 D(Y) 越大越好,也就是让 Discriminator Output 的值越小越好

你可能觉得没事突然写出这个式子有点奇怪,但你不一定要把这个 Objective Function,写成这个样子,它完全可以有其他的写法,那最早年之所以写成这个样子,是有一个很明确的理由,是为了要把 Discriminator 跟 Binary Classification,跟二元的分类扯上关係

事实上这个 Objective Function,它就是 Cross Entropy 乘一个负号,我们知道我们在训练一个 Classifier 的时候,我们就是要 Minimize Cross Entropy,所以当我们 Maximize Cross Entropy 乘一个负号的时候,其实等同於 Minimize Cross Entropy,也就是等同於是在训练一个 Classifier

那这个 Discriminator,其实可以当做是一个 Classifier,它做的事情就是把蓝色这些点,从 Pdata Sample 出来的真实的 Image,当作 Class 1,把从 PG Sample 出来的这些假的 Image,当作 Class 2

有两个 Class 的 Data,训练一个 Binary 的 Classifier,训练完就等同於是,解了这一个 Optimization 的问题



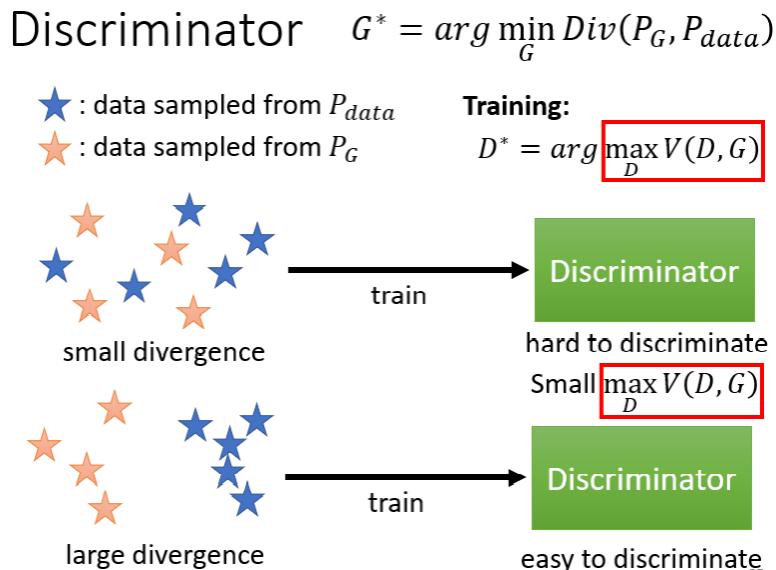
那这边最神奇的地方是这一个式子,这个红框框裡面的数值,它跟 JS Divergence 有关

那事实上有趣的事情是,我觉得最原始的 GAN 的 Paper,它的发想可能真的是从 Binary Classifier 来的,一开始是把 Discriminator,写成 Binary 的 Classifier,然后有了这样的 Objective Function,然后再经过一番推导以后,这个 Objective Function,它的 Maximum,就是你找到一个 D,可以让这个 Objective Function,它的值最大的时候,这个最大的值跟 JS Divergence 是有关的。它们没有完全一模一样,所以显然一开始,并不是针对 JS Divergence 设计的,而是经过一番推导以后,发现它们是非常有关联的,那至於实际上的推导过程,你可以参见原始 Ian J. Goodfellow 写的文章,那其实裡面的推导过程,我觉得写得算是蛮清楚的

所以真正神奇的地方就是,这一个 Objective Function 的最大值,它跟 Divergence 是有关的,所以我们刚才说,我们不知道怎麽算 Divergence 没关係,Train 你的 Discriminator,Train 完以后,看看它的 Objective Function 可以到多大,那个值就跟 Divergence 有关

这边我们并没有把证明拿出来跟大家讲了,但是我们还是可以从直观上来理解一下,為什麼这个 Objective Function 的值,会跟 Divergence 有关

你可以想想看,假设 PG 跟 Pdata,它的 Divergence 很小



- 也就 PG 跟 Pdata 很像,它们差距没有很大,它们很像 PG 跟 Pdata Sample 出来的,蓝色的星星跟红色的星星,它们是混在一起的

这个时候Discriminator 就是在 Train 一个,Binary 的 Classifier,对 Discriminator 来说,既然这两堆资料是混在一起的,那就很难分开,这个问题很难,所以你在解这个 Optimization Problem 的时候,你就没有办法让这个 Objective 的值非常地大

所以这个 Objective 这个 V 的 Maximum 的值就比较小,所以小的 Divergence,对应到小的这个 Objective Function 的 Maximum 的值

所以不是 Objective Function 的值本身,是 Objective Function 在穷举地找 Discriminator 要可以得到的最大的值,

- 如果今天,你的两组 Data 很不像,它们的 Divergence 很大,那对 Discriminator 而言,就可以轻易地把它分开

当 Discriminator 可以轻易把它分开的时候,这个 Objective Function 就可以冲得很大,那所以当你有大的 Divergence 的时候,这个 Objective Function 的 Maximum 的值,就可以很大

详细的证明请参见 GAN 原始的 Paper,裡面在做了一些假设,比如说 Discriminator 的 Capacity 是无穷大,等等的假设以后,可以做出这个 Maximum 的值,跟 JS Divergence 一些相关的这个推导

所以我们说我们本来的目标是要找一个 Generator,去 Minimize PG 跟 Pdata 的 Divergence

$$G^* = \arg \min_G \text{Div}(P_G, P_{\text{data}})$$

但我们卡在不知道怎麽计算 Divergence,那我们现在要知道,我们只要训练一个 Discriminator,训练完以后,这个 Objective Function 的最大值,就是这个 Divergence,就跟这个 Divergence 有关

$$G^* = \arg \min_G \text{Div}(P_G, P_{\text{data}})$$

$$D^* = \arg \max_D V(D, G)$$

The maximum objective value
is related to JS divergence.

那我们何不就把红框框裡面这一项,跟 Divergence 做替换,我们何不就把 Divergence,替换成红框框裡面这一项,所以我们就有了这样一个 Objective Function

$$G^* = \arg \min_G \max_D V(G, D)$$

$$D^* = \arg \max_D V(D, G)$$

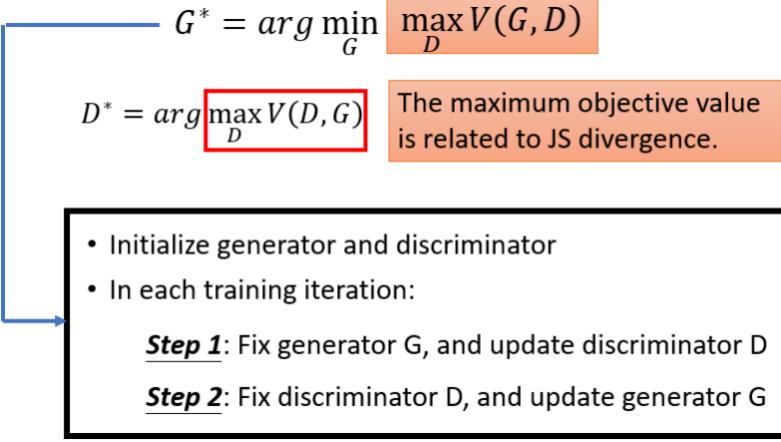
The maximum objective value
is related to JS divergence.

这个 Objective Function乍看之下有点复杂,它有一个 Minimum,又有一个 Maximum,所以你不小心就会脑筋转不过来

- 我们是要找一个 Generator,去 Minimize 红色框框裡面这件事
- 但是红框框裡面这件事,又是另外一个 Optimization Problem,它是在给定 Generator 的情况下,去找一个 Discriminator,这个 Discriminator,可以让 V 这个 Objective Function 越大越好

Conclusion

然后我们要找一个 G,让红框框裡面的值最小,这个 G 就是我们的 Generator,而刚才我们讲的这个 Generator 跟 Discriminator,互动 互相欺骗这个过程,其实就是要解这一个有 Minimize,又有 Maximize 这个 Min Max,Min Max 的问题,就是透过下面这一个,我们刚才讲的 GAN 的 Argument 来解的



那至於实际上,為什麼下面这个 Argument 可以解这个问题,你也可以参见原始 GAN 的 Paper

那讲到这边,也许你就会问说,為什麼是 JS Divergence,而且还不是真的 JS Divergence,是跟 JS Divergence 相关而已,怎麼不用真正的 JS Divergence,或不用别的,比如说 KL Divergence

你完全可以这麼做,你只要改了那个 Objective Function,你就可以量各式各样的 Divergence,那至於怎麼样设计 Objective Function,得到不同的 Divergence,那有一篇叫做 F GAN 的 Paper 裡面,有非常详细的证明

Can we use other divergence?

Name	$D_f(P\ Q)$	Generator $f(u)$
Total variation	$\frac{1}{2} \int p(x) - q(x) dx$	$\frac{1}{2} u - 1 $
Kullback-Leibler	$\int p(x) \log \frac{p(x)}{q(x)} dx$	$u \log u$
Reverse Kullback-Leibler	$\int q(x) \log \frac{q(x)}{p(x)} dx$	$-\log u$
Pearson χ^2	$\int \frac{(q(x)-p(x))^2}{p(x)} dx$	$(u-1)^2$
Neyman χ^2	$\int \frac{(p(x)-q(x))^2}{q(x)} dx$	$\frac{(1-u)^2}{u}$
Squared Hellinger	$\int \left(\sqrt{p(x)} - \sqrt{q(x)} \right)^2 dx$	$(\sqrt{u}-1)^2$
Jeffrey	$\int (p(x) - q(x)) \log \left(\frac{p(x)}{q(x)} \right) dx$	$(u-1) \log u$
Jensen-Shannon	$\frac{1}{2} \int p(x) \log \frac{2p(x)}{p(x)+q(x)} + q(x) \log \frac{2q(x)}{p(x)+q(x)} dx$	$-(u+1) \log \frac{1+u}{2} + u \log u$
Jensen-Shannon-weighted	$\int p(x) \pi \log \frac{2p(x)}{\pi p(x)+(1-\pi)q(x)} + (1-\pi)q(x) \log \frac{q(x)}{\pi p(x)+(1-\pi)q(x)} dx$	$\pi u \log u - (1-\pi+\pi u) \log(1-\pi+\pi u)$
GAN	$\int p(x) \log \frac{2p(x)}{p(x)+q(x)} + q(x) \log \frac{2q(x)}{p(x)+q(x)} dx - \log(4)$	$u \log u - (u+1) \log(u+1)$

Name	Conjugate $f^*(t)$
Total variation	t
Kullback-Leibler (KL)	$\exp(t-1)$
Reverse KL	$-1 - \log(-t)$
Pearson χ^2	$\frac{1}{4}t^2 + t$
Neyman χ^2	$2 - 2\sqrt{1-t}$
Squared Hellinger	$\frac{t}{1-t}$
Jeffrey	$W(e^{1-t}) + \frac{1}{W(e^{1-t})} + t - 2$
Jensen-Shannon	$-\log(2 - \exp(t))$
Jensen-Shannon-weighted	$(1-\pi) \log \frac{1-\pi}{1-\pi e^{t/\pi}}$
GAN	$-\log(1 - \exp(t))$

Using the divergence you like 😊

<https://arxiv.org/abs/1606.00709>

它有很多的 Table 告诉你说,不同的 Divergence,要怎麽设计它的 Objective Function,你设计什麼样的 Objective Function,去找它的 Maximum Value,就会变成什麼样的 Divergence,在这篇文章裡<https://arxiv.org/abs/1606.00709>面都有详细的记载,这一开始有人会觉得说,GAN 之所以没有很好 Train,也许是因為,就是我们没有在真的,这个 Minimize JS Divergence

但是有了这个 F GAN 这个 Paper以后,它就告诉你说,我们有办法 Minimize JS Divergence,但就算你真的可以 Minimize JS Divergence,结果也还是没有很好,GAN 还是没有很好 Train

所以俗话就说,No Pain No Gan

GAN is difficult to train



(I found this joke from 陳柏文's facebook.)

所以接下来我们就要讲一些,GAN 训练的小技巧

Tips for GAN——WGAN

其实它 GAN 训练的小技巧非常非常多,我们只挑一个最知名的来讲,这个最知名的就是很多人都听过,就是很多人都听过的 WGAN

在讲 WGAN 之前,我们先讲 JS Divergence 有什么样的问题, 在最早的 GAN 说,我们要 Minimize 的是 JS Divergence,那选择 JS Divergence 的时候,会有什麼问题

Problems of JS divergence

在想 JS Divergence 的问题之前,我们先看一下 PG 跟 Pdata 有什么样的特性

那在实作中, PG 跟 Pdata 有一个非常关键的特性是,PG 跟 Pdata 它们重叠的部分往往非常少

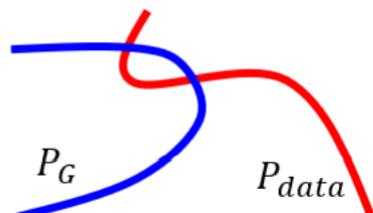
这边有两个理由

- 第一个理由是来自於 Data 本身的特性

- 1. The nature of data

Both P_{data} and P_G are low-dim manifold in high-dim space.

The overlap can be ignored.



PG 跟 Pdata,它们都是要產生图片的,那图片其实是高维空间裡面的一个低维的 Manifold(流形 (英语: Manifolds) 是可以局部欧几里得空间化的一个拓扑空间, 是欧几里得空间中的曲线、曲面等概念的推广),怎麼知道图片是高维空间,裡面低维的 Manifold

我们可以想想看,你在高维空间裡面,随便 Sample 一个点,它通常都没有办法构成一个,二次元人物的头像,所以二次元人物的头像它的分布,在高维的空间中,其实是非常狭窄的,所以二次元头像的分布,这个图片的分布,其实是高维空间中的低维的 Manifold

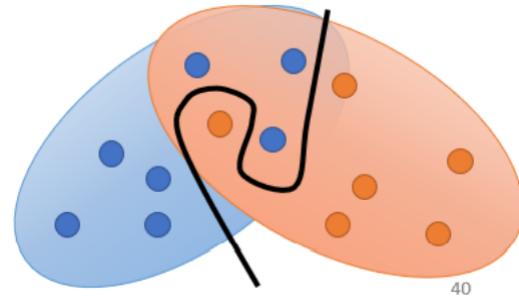
或者是以二维空间来想的话,那图片的分布,可能就是二维空间的一条线,二维空间中多数的点都不是图片,就高维空间中随便 Sample 一个点,都不是图片,只有非常小的范围,Sample 出来它会是图片,所以从这个角度来看,从资料本身的特性来看,PG 跟 Pdata,它们都是 Low Dimensional 的 Manifold,用二维空间来讲,PG 跟 Pdata 都是二维空间中的两条直线,而二维空间中的两条直线,除非它刚好重合,不然它们相交的范围,几乎是忽略的,这是第一个理由,那也许有人说,这个也许图片根本就不是什麼 Low Dimensional 的 Manifold,那会不会第一个理由就不成立了

- 第二个理由是,我们是从来都不知道 PG 跟 Pdata 长什麼样子,我们对 PG 跟 Pdata,它的分布的理解,其实来自於 Sample

• 2. Sampling

Even though P_{data} and P_G have overlap.

If you do not have enough sampling



所以也许 PG 跟 Pdata,它们是有非常大的 Overlap 的范围,但是我们实际上,在了解这个 PG 跟 Pdata,在计算它们的 Divergence 的时候,我们是从 Pdata 裡面 Sample 一些点出来,从 PG 裡面 Sample 一些点出来

如果你 Sample 的点不够多,你 Sample 的点不够密,那就算是这两个 Divergence 实际上,这两个 Distribution 实际上有重叠,但是假设你 Sample 的点不够多,对 Discriminator 来说,它也是没有重叠的

就是这个蓝色的分布跟红色的分布,明明是有重叠的,但如果你从蓝色分布 Sample 一些点,红色分布 Sample 一些点,这些点你又 Sample 得不够多,你完全就可以画一条楚河汉界,把红色的点跟蓝色的点完全地分开来,然后说红色的点,它的分布就是在这个楚河汉界的右边,蓝色的点就在左边,它们完全是没有任何冲突

所以以上给你两个理由,试图说服你说 PG 跟 Pdata 这两个分布,它们重叠的范围是非常小的

而几乎没有重叠这件事情,對於 JS Divergence 会造成什麼問題?

JS Divergence 有个特性,是两个没有重叠的分布,JS Divergence 算出来,就永远都是 Log2,不管这两个分布长什麼样,所以两个分布只要没有重叠,算出来就一定是 Log2

所以举例来说

$$P_{G_0} \quad \longleftrightarrow \quad P_{data}$$

$$JS(P_{G_0}, P_{data}) = \log 2$$

假设这是你的 Pdata,这是你的 PG,它们都假设它们都是一条直线,然后中间有很长的距离,你算它们的 JS Divergence,是 Log2

假设你的 PG 跟 Pdata 其实蛮接近的

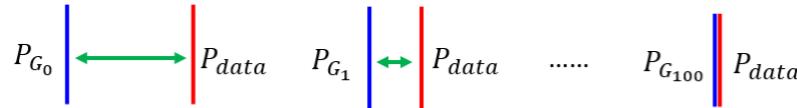
$$P_{G_{100}} \parallel P_{data}$$

$$JS(P_{G_{100}}, P_{data}) = 0$$

那中间的间隔其实是比较小的,算出来结果还是 Log2

除非你的 PG 跟 Pdata 有重合

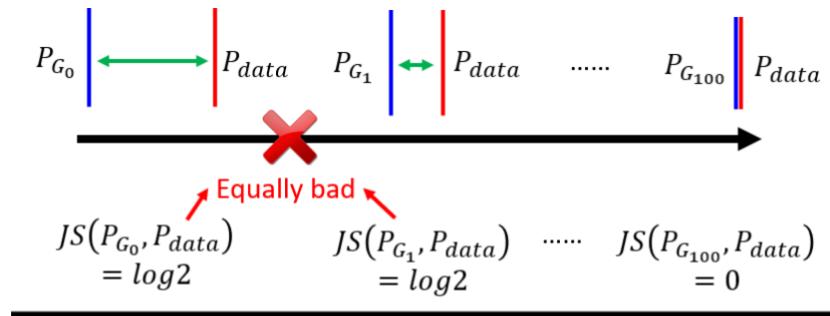
JS divergence is always log2 if two distributions do not overlap.



$$\begin{aligned} JS(P_{G_0}, P_{data}) &= \log 2 \\ JS(P_{G_1}, P_{data}) &= \log 2 \\ \dots & \\ JS(P_{G_{100}}, P_{data}) &= 0 \end{aligned}$$

Equally bad

不然这个 PG 跟 Pdata 只要它们是两条直线,它们这两条直线没有相交,那算出来就是 Log2,算出来这个 Case,算出来是 Log2,这个 Case 算出来也是 Log2



Intuition: If two distributions do not overlap, binary classifier achieves 100% accuracy.

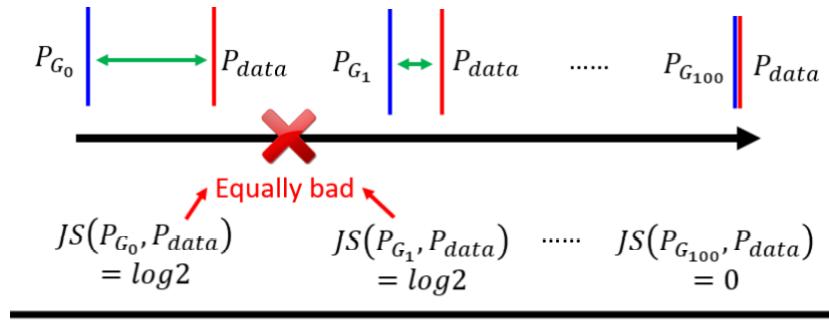
The accuracy (or loss) means nothing during GAN training.

那但是明明中间这个 Case,中间这个 Generator 就比左边这个 Generator 好,明明蓝色的线就跟红色的线比较近,但是从 JS Divergence 上面,看不出这样子的现象

那既然从 JS Divergence 上,看不出这样的现象,你在 Training 的时候,你根本就没有办法把这样子的 Generator,Update 参数变成这样子的 Generator,因为对你的 Loss 来说,对你的目标来说,这两个 Generator 是一样好,或者是一样糟

那以上是从比较理论的方向来说明,如果我们从更直观的实际操作的角度来说明,你会发现,当你是用 JS Divergence 的时候,你就假设你今天在 Train 一个,Binary 的 Classifier,去分辨 Real 的 Image,跟 Generated Image,你会发现实际上你通常 Train 完以后,正确率几乎都是 100%

因为你 Sample 的图片根本就没几张,对你的 Discriminator 来说,你 Sample 256 张 Real 的图片,256 张 Fake 的图片,它直接用硬背的,都可以把这两组图片分开,知道谁是 Real 谁是 Fake,所以实际上,如果你有自己 Train 过 GAN 的话你会发现,如果你用 Binary 的 Classifier Train 下去,你会发现,你几乎每次 Train 完你的 Classifier 以后,也就你 Train 完你的 Discriminator 以后,正确率都是 100%



Intuition: If two distributions do not overlap, binary classifier achieves 100% accuracy.

The accuracy (or loss) means nothing during GAN training.

我们本来会期待说,这个 Discriminator 的 Loss,也许代表了某些事情,这个 Binary Classifier Loss,也许代表某些事情,这个 **Loss 越来越大,代表问题越来越难**,代表我们的 Generated Data,跟 Real 的 Data 越来越接近

但实际上,你在实际操作的时候你根本观察不到这个现象,这个 Binary Classifier 训练完的 Loss 根本没有什麼意义,因為它总是可以让它的正确率变到 100%,两组 Image 都是 Sample 出来的,它硬背都可以得到 100% 的正确率,你根本就没有办法看出你的 Generator,有没有越来越好

所以过去,尤其是在还没有 WGAN 这样的技术,在我们还用 Binary Classifier,当作 Discriminator 的时候,Train GAN 真的就很像巫术 黑魔法,你根本就不知道你 Train 的时候,有没有越来越好,所以怎麼办,那时候做法就是,你每次 Update 几次 Generator 以后,你就要把你的图片 Print 出来看,然后你就要一边那个吃饭,一边看那个图片生成的结果,然后跑一跑就发现 哇 坏掉了,然后咔掉重做这样子

所以以前你根本就没有,不像我们在 Train 一般的 Network 的时候,你有个 Loss Function,然后那个 Loss 随著训练的过程,会慢慢慢慢变小,那你就会看说 Loss 慢慢变小,你就放心知道说你的 Network 有在 Train,但会不会 Overfitting 是另外一件事,至少还在它在 Training Data 上有越来越好

但是对 GAN 而言,本来我们期待 Classifier 的 Loss,可以提供一些资讯,但是当你的 Classifier 是一个,简单一般的 Binary Classifier 的时候,它训练的结果你就没有任何资讯,你每次训练出来正确率都是 100%,你根本不知道你的 Generator,有没有越来越好,变成你只能够用人眼看,用人眼守在电脑前面看,发现结果不好 咔掉,重新用一组 Hyperparameter,重新调一下 Network,加工重做,所以过去训练 GAN 是有点辛苦的

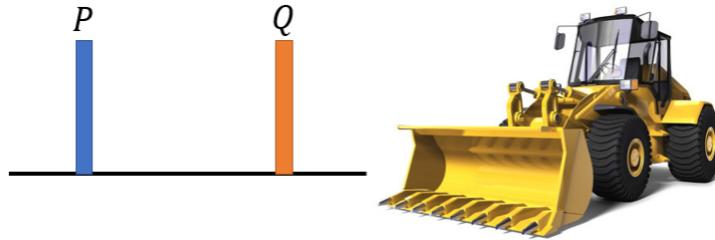
那既然是 JS Divergence 的问题,於是有人就想说,那会不会换一个,衡量两个 Distribution 的相似程度的方式,换一种 Divergence,就可以解决这个问题了,於是就有了这个 Wasserstein,或使用 Wasserstein Distance 的想法

Wasserstein Distance

Wasserstein Distance 的想法是这个样子,假设你有两个 Distribution,一个 Distribution 我们叫它 P,另外一个 Distribution 我们叫它 Q

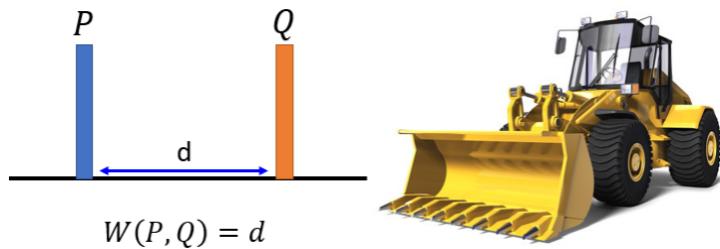
Wasserstein Distance 它计算的方法,就是想像你在开一台推土机

- Considering one distribution P as a pile of earth, and another distribution Q as the target
- The average distance the earth mover has to move the earth.



推土机的英文叫做 Earth Mover,想像你在开一台推土机,那把你 P 想成是一堆土,把 Q 想成是你要把土堆放的目的地,那这个推土机把 P 这边的土,挪到 Q 所移动的平均距离,就是 Wasserstein Distance

在这个例子裡面,我们假设 P 都集中在这个点, Q 都集中在这个点,对推土机而言,假设它要把 P 这边的土挪到 Q 这边,那它要平均走的距离,就是 D



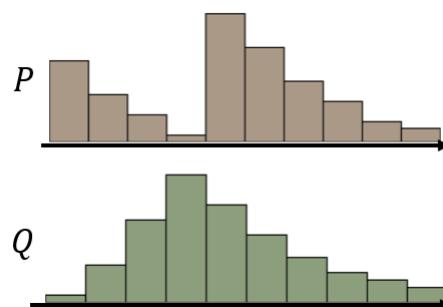
所以在这个例子裡面,假设 P 集中在一个点, Q 集中在一个点,这两个点之间的距离是 D 的话,那 P 跟 Q 的 Wasserstein Distance,就是 D

Problems of Wasserstein Distance

那因為在讲这个 Wasserstein Distance 的时候,你要想像有一个 Earth Mover,有一个推土机在推土,所以其实 Wasserstein Distance,又叫 Earth Mover Distance

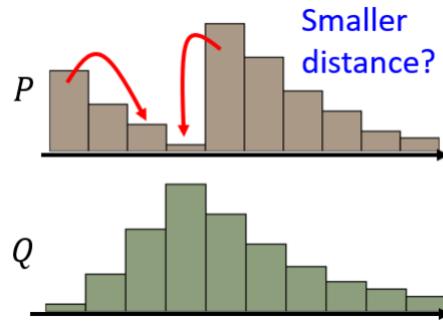
但是如果s是更复杂的 Distribution,你要算 Wasserstein Distance,就有点困难了

假设这是你的 P ,假设这是你的 Q

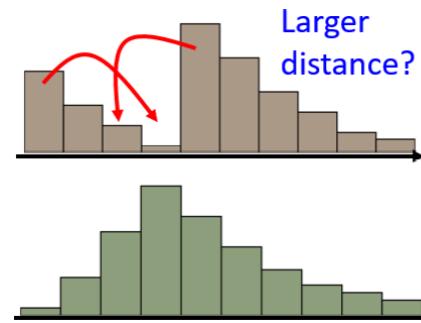


假设你开了一个推土机,想要把 P 把它重新塑造一下形状,让 P 的形状跟 Q 比较接近一点,那有什麼样的做法,你会发现说,你可能的 Moving Plans,把 P 重新塑造成 Q 的方法有无穷多种

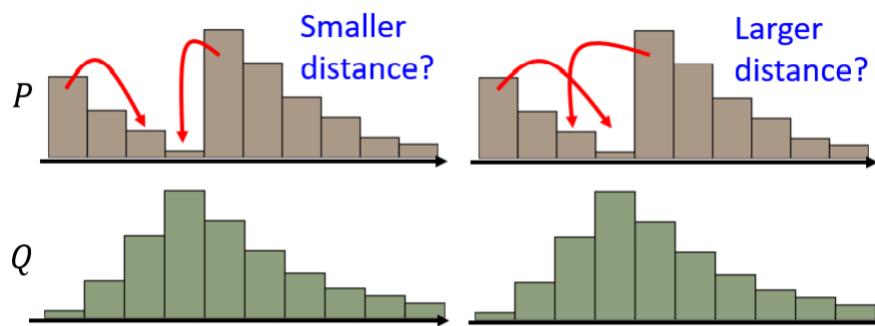
你可以说我把这边的土搬到这裡来,我把这边的土搬到这裡来,把 P 变成 Q



但你也可以捨近求远说,我把这裡的土搬到这裡来,把这裡的土搬到这裡来,捨近求远,一样还是可以把 P 变成 Q



所以当我们考虑,比较复杂的 Distribution 的时候,把 Q 把 P 变成 Q 的方法,是有非常非常多不同的方法,你有各式各样不同的 Moving Plan,用不同的 Moving Plan,你推土机平均走的距离就不一样



There are many possible “moving plans”.

Using the “moving plan” with the smallest average distance to define the Wasserstein distance.

Source of image: <https://vincentherrmann.github.io/blog/wasserstein/>

43

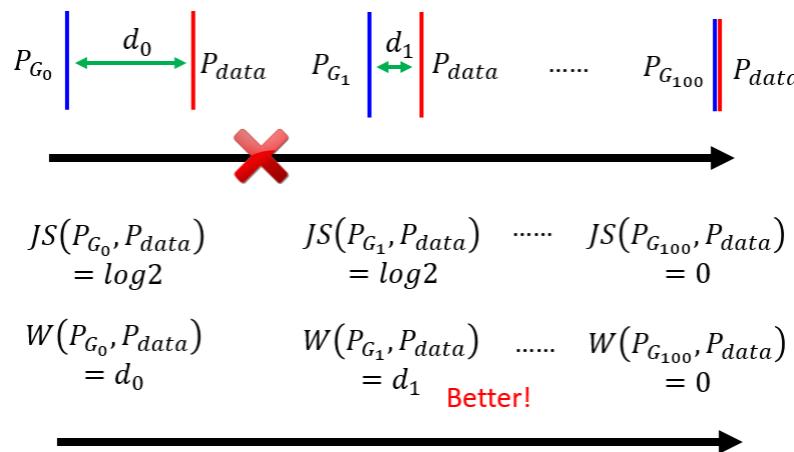
在左边这个例子裡面,推土机平均走的距离比较少,在右边这个例子裡面因為捨近求远,推土机平均走的距离比较大

為了让 Wasserstein Distance 只有一个值,所以这边 Wasserstein Distance 的定义是,穷举所有的 Moving Plans,然后看哪一个推土的方法,哪一个 Moving 的计划,可以让平均的距离最小,那个最小的值,才是 Wasserstein Distance

所以其实要计算 Wasserstein Distance,是挺麻烦的,光只是要计算一个 Distance,居然还要解一个 Optimization 的问题,解出这个 Optimization 的问题,才能算 Wasserstein Distance

Advantages of Wasserstein Distance

我们先不讲,怎麽计算 Wasserstein Distance 这件事,我们先来讲假设我们能够计算 Wasserstein Distance 的话,它可以带给我们什麼样的好处



那假设 PG 跟 Pdata 它们的距离是 d_0 ,在左边这个例子裡面,Wasserstein Distance 算出来就是 d_0

在中间这个例子裡面,PG 跟 Pdata 它们之间的距离是 d_1 ,那 Wasserstein Distance 算出来的距离,就是 d_1

假设 d_1 比较小, d_0 比比较大,那算 Wasserstein Distance 的时候,这个 Case 的 Wasserstein Distance 就比较小,这个 Case 的 Wasserstein Distance 就比较大,所以我们就可以知道说,由左向右的时候,Wasserstein Distance 是越来越小的

所以如果你观察,Wasserstein Distance 的话会知道说,从左到右我们的 Generator 越来越进步,但是如果你观察 Discriminator,你会发现你观察不到任何东西,对 Discriminator 而言,这边每一个 Case 算出来的 JS Divergence,都是一样,都是一样好或一样差,但是如果换成 Wasserstein Distance,由左向右的时候我们会知道说,我们的 Discriminator 做,我们的 Generator 做得越来越好

所以我们换一个计算 Divergence 的方式,我们就可以解决 JS Divergence,有可能带来的问题

这又让我想到一个演化的例子,这是眼睛的生成



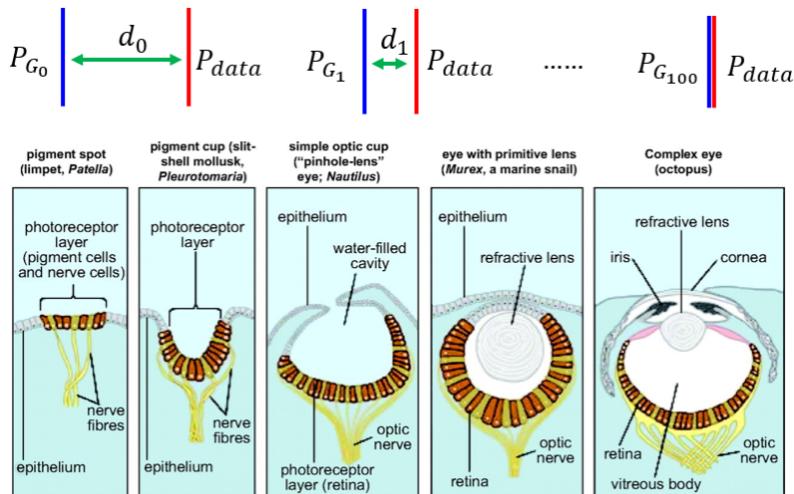
https://www.pnas.org/content/104/suppl_1/8567.figures-only

45

右边这个是人类的眼睛,人类的眼睛是非常地复杂的,那有一些生物它有非常原始的眼睛,比如说有一些细胞具备有感光的能力,这可以看做是最原始的眼睛,但是这些最原始的眼睛,怎麽变成最复杂的眼睛,这对人类来说其实觉得非常难想像

左边这个图像构造这麼简单,只是一些感光的细胞在皮肤上,经过突变產生一些感光的细胞,听起来像是一个合理的,但是天择突变,怎麽可能產生这麼复杂的器官,怎麽產生眼睛这麼精巧的器官

那如果你直接觉得说,从这个地方就可以一步跳到这个地方,那根本不可能发生,但是中间其实是有很连
续的步骤,从感光细胞到眼睛,中间其实是有连续的步骤的



https://www.pnas.org/content/104/suppl_1/8567.figures-only

45

举例来说,感光的细胞可能会出现在一个比较凹陷的地方,皮肤凹陷下去,这样感光细胞可以接受来自不同方向的光源,然后后来觉得说,乾脆把凹陷的地方盖起来,后来觉得盖起来的地方裡面,可以放一些液体,然后最后就变成了人的眼睛

而从最左边跳到最右边这一步,非常大,但是这边每一步,都可以让一个生命存活的机率变大,都可以让生命繁衍下去的机率变高

既然这边每一个步骤,都可以让生命繁衍的机率变高,那生命就可以从左边一直演化到右边,最终產生复杂的器官

而我觉得對於当你使用 WGAN,当你使用这个 Wasserstein Distance,来衡量你的 Divergence 的时候,其
实就製造了类似的效果,本来 PG0 跟 Pdata 非常地遥远,你要它一步从这裡跑到这裡,一步从这裡跑到这裡,
让这个 PG0 跟 Pdata,直接 Align 在一起,是不可能的,可能非常地困难

对 JS Divergence 而言,它需要做到直接从这一步跳到这一步,它的 JS Divergence 的 Loss 才会有差异,但是对 **W Distance** 而言,你只要每次有稍微把 PG 往 Pdata 挪近一点,W Distance 就会有变化,W Distance 有变化,你才有办法 Train 你的 Generator,去 Minimize W Distance,所以这就是為什麼,当我们从 JS Divergence,换成 Wasserstein Distance 的时候,可以带来的好处

WGAN

WGAN 实际上就是用,当你用 Wasserstein Distance,来取代 JS Divergence 的时候,这个 GAN 就叫做 WGAN

接下来你会遇到的问题就是,Wasserstein Distance 是要怎麼算,Pdata 跟 PG,它的 Wasserstein Distance 要怎麼计算,Wasserstein Distance,是一个非常复杂的东西,我光要算个 Wasserstein Distance,还要解一个 Optimization 的问题,实际上要怎麼计算

那这边就不讲过程,直接告诉结果,解下面这个 Optimilazion 的 Problem,解出来以后你得到的值,就是 Wasserstein Distance

$$\max_{D \in 1-Lipschitz} \{E_{y \sim P_{data}}[D(y)] - E_{y \sim P_G}[D(y)]\}$$

我们就观察一下这个式子,这个式子裡面有说

- y 如果是从 Pdata 来的,那我们要计算它的 $D(y)$ 的期望值
- y 如果是从 PG 来的,我们计算它的 $D(y)$ 的期望值,但是前面乘上一个负号

所以如果你要 Maximum,这个 Objective Function,你会达成这样的效果,

- 如果 y 是从 P_{data} Sample 出来的, $D(y)$,就 Discriminator 的 Output 要越大越好
- 如果 X 是从 P_G ,从 Generator Sample 出来的,那 $D(y)$,也就 Discriminator 的 Output,应该要越小越好

$$\max_{D \in 1-\text{Lipschitz}} \{E_{y \sim P_{data}}[D(y)] - E_{y \sim P_G}[D(y)]\}$$

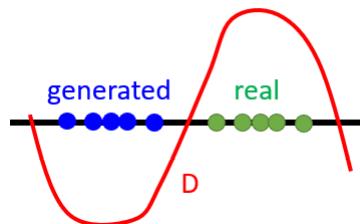
D has to be smooth enough.

但是这边还有另外一个限制,它不是光大括号裡面的值变大就好,还有一个限制是,D 不能够是一个随便的 Function,D必须是一个 1-Lipschitz 的 Function

1-Lipschitz 是什麼东西,如果你不知道是什麼的话,也没有关係

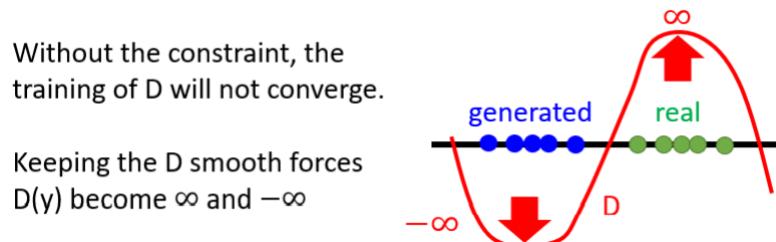
我们这边你就想成,D 必须要是一个足够平滑的 Function,它不可以是变动很剧烈的 Function,它必须要是
一个足够平滑的 Function

那為什麼足够平滑这件事情是非常重要的,我们可以从直观来理解它,假设这个是真正的资料的分布,这是
Generated 的资料的分布



如果我们没有这个限制,只看大括号裡面的值的话,大括号裡面的目标,是要这些真正的值,它的 $D(y)$ 越大越好,那要让 Generated 的值,它的 $D(y)$ 越小越好

如果你没有做任何限制,只单纯要这边的值越大越好,这边的值越小越好,在蓝色的点跟绿色的点,也就是真正的 Image,跟 Generated 的 Image,没有任何重叠的情况下,你的 Discriminator 会做什麼,它会给 Real 的 Image 无限大的正值,给 Generated 的 Image 无限大的负值



所以你这个 Training 根本就没有办法收敛,而且你会发现说,只要这两堆 Data 没有重叠,你算出来的值都是无限大,你算出来的这个 Maximum 值都是无限大,这显然不是我们要的,这不就跟 JS Divergence 的问题一模一样吗

其实你要加上这个限制以后,你这个 Maximum 的值,才会是 Wasserstein Distance,才是做 WGAN

為什麼加上这个限制,就可以解决刚才的问题?

因為这个限制是要求 Discriminator,不可以太变化剧烈,它必须要够听话,那今天如果你,要求你的
Discriminator 够平滑的时候,假设 Real Data 跟 Generated Data,它们距离比较近,那你就没有办法让
Real 的 Data 值非常大,然后 Generated 的值非常小,因為如果你让 Real 的值非常大,Generated 的值非常小,它们中间的差距很大,那这一个 Discriminator 变化就很剧烈,它就不平滑了,它就不是 1-Lipschitz

那為了要是 1-Lipschitz,这边的值没办法很大,这边的值没办法很小,让 Discrimination 必须足够,只觉得发现说,如果 Real 跟 generated 的 Data 差距离很远,那它们的值就可以差很多,这边算出来的值就会比较大,你的 Wasserstein Distance 就比较大

如果 Real 跟 Generated 很近,就算它们没有重合,因為有了这一个限制,有了这一个写在 Max 下面的,1-Lipschitz Function 的限制,那你就发现说,它们其实不会真的都跑到无限大,它们的距离,因為有 1-Lipschitz Function 的限制,所以 Real Data 的值跟 Generated Data 的值,就没有办法差很多,那你算出来的 Wasserstein Distance,就会比较小

接下来的问题就是,怎麼做到这个限制

怎麼确保 Discriminator,一定符合 1-Lipschitz Function 的限制,最早刚提出 WGAN 的时候,其实没有什麼好想法,只知道写出了这个式子,那要怎麼真的解这个式子,有点困难,所以最早的一篇 WGAN 的 Paper,最早使用 Wasserstein 的那一篇 Paper,它说了它做了一个比较 Rough,比较粗糙的处理的方法

$$\max_{D \in 1-\text{Lipschitz}} \{E_{y \sim P_{\text{data}}} [D(y)] - E_{y \sim P_G} [D(y)]\}$$

- Original WGAN → Weight

Force the parameters w between c and -c

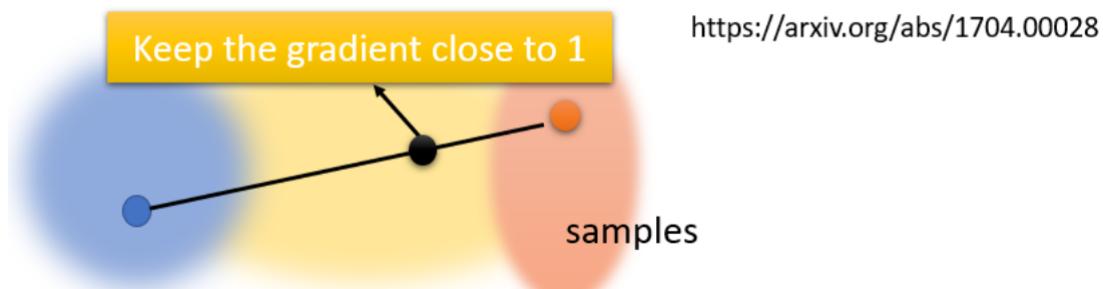
After parameter update, if w > c, w = c; if w < -c, w = -c

它是说我就 Train Network,那 Train Network 的时候,如果我 Training 的那个参数,我就要求它放得在 C 跟 -C 之间,如果超过 C,用 Gradient Descent Update 以后超过 C,就设為 C,Gradient Descent Update 以后小於 -C,就直接设為 -C

其实这个方法,并不一定真的能够让 Discriminator,变成 1-Lipschitz Function,那你可以想像说这个方法,也许真的可以让我们的 Discriminator,比较平滑,但它并没有真的去解这个,Optimization 的 Problem,它并没有真的让 Discriminator,符合这个限制

所以接下来就有其它的想法,有一个想法叫做 Gradient Penalty

- Improved WGAN → Gradient Penalty



Gradient Penalty 是出自,Improved WGAN 这篇 Paper<https://arxiv.org/abs/1704.00028>,那 Improve WGAN 这边paper 是说

假设这个是你的 Real Data 的分布,这个是你的这个 Fake Data 的分布,那我在 Real Data 这边取一个 Sample,Fake Data 这边取一个 Sample,两点连线中间再取一个 Sample,我要求这个点它的 Gradient 要接近,那你可能觉得说这个,不知道 不知道在说些什麼这样,这件事情的关係,跟这个限制的关係,到底是什麼,那就详见 Improved WGAN 的 Paper

那其实我觉得你现在,也不一定要真的非常较真说,这件事情 Gradient Penalty,跟这个限制之间的关係,因为其实有更多其它的方法,真的可以去解这个问题

那其实后来 Improved WGAN 之后,还有什麼 Improved 的 Improved WGAN,真的有这篇 Paper,它就是把这个限制再稍微改一改

有另外一篇 Paper 就叫 Improved The Improved WGAN,那今天其实你有一个方法,真的把 D 限制,让它 是 1-Lipschitz Function,这个叫做 Spectral Normalization

那我就把它的论文<https://arxiv.org/abs/1802.05957>放在这边给大家参考,那如果你要 Train 真的非常好的 GAN,你可能会需要用到 Spectral Normalization

- Spectral Normalization → Keep gradient norm smaller than 1 everywhere

<https://arxiv.org/abs/1802.05957>

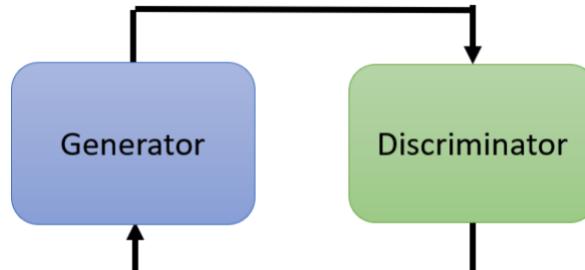
Other tips

虽然说已经有 WGAN,但其实并不代表说,GAN 就一定特别好 Train,GAN 仍然是以,很难把它 Train 起来而闻名的,那为什麼 GAN 很难被 Train 起来?

它有一个本质上困难的地方

- Discriminator 做的事情,是要分辨真的图片跟产生的出来的,也就是假的图片的差异
- 而 Generator 在做的事情,它是要去产生假的图片,骗过 Discriminator

Generate fake images to fool discriminator

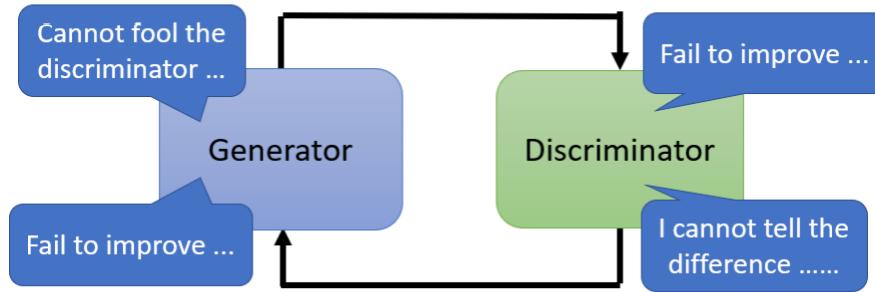


Tell the difference between real and fake

而事实上这两个 Network,这个 Generator 跟 Discriminator,它们是互相砥砺,才能互相成长的,只要其中一者,发生什麼问题停止训练,另外一者就会跟著停下训练,就会跟著变差

- Generator and Discriminator needs to match each other (棋逢敌手)

Generate fake images to fool discriminator



Tell the difference between real and fake

- 假设你在 Train Discriminator 的时候,一下子没有 Train 好
- 你的 Discriminator 没有办法分辨,真的跟产生的出来的图片的差异
- 那 Generator,它就失去了可以进步的目标,Generator 就没有办法再进步了
- 如果 Generator 没有办法再进步,它没有办法再产生更真实的图片,那 Discriminator 就没有办法再跟著进步了

但是到目前为止,大家已经 Train 过了很多次的 Network,我们没有办法保证 Train 下去,它的 Loss 就一定会下降,你要让 Network Train 起来,往往你需要调一下 Hyperparameter,才有可能把它 Train 起来

那今天这个 Discriminator 跟 Generator,它们互动的过程是自动的,因为我们不会在中间,每一次 Train Discriminator 的时候,你都换 Hyperparameter

所以只能祈祷每次 Train Discriminator 的时候,它的 Loss 都是有下降的,那如果有一次没有下降,那整个 Training,很有可能就会变就会惨掉,整个 Discriminator 跟 Generator,彼此砥礪的这个过程,就可能会停下来

所以今天 Generator 跟 Discriminator,在 Train 的时候,它们必须要棋逢敌手,就任何一个人放弃了这一场比赛,另外一个人也就玩不下去了

因此 GAN 本质上它的 Training,仍然不是一件容易的事情,当然它是一个非常重要的技术,所以虽然它是一个前瞻性的技术

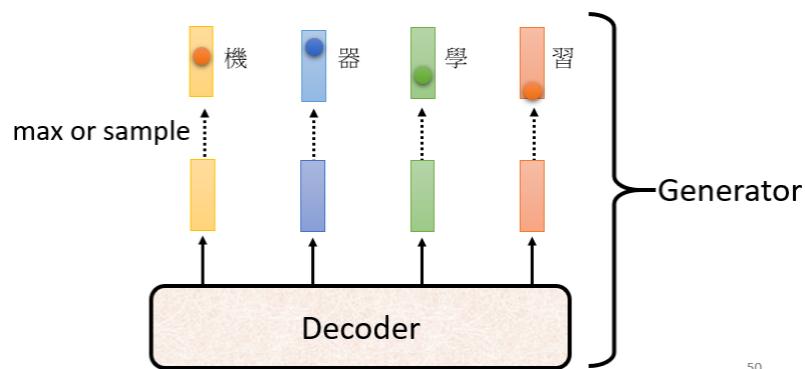
一些相关的,跟 Train GAN 的诀窍有关的文献,还有链接列在这边,其实就给大家自己参考

- [Tips from Soumith](#)
- [Tips in DCGAN: Guideline for network architecture design for image generation](#)
- [Improved techniques for training GANs](#)
- [Tips from BigGAN](#)

GAN for Sequence Generation

Train GAN 最难的其实是要拿 GAN 来生成文字

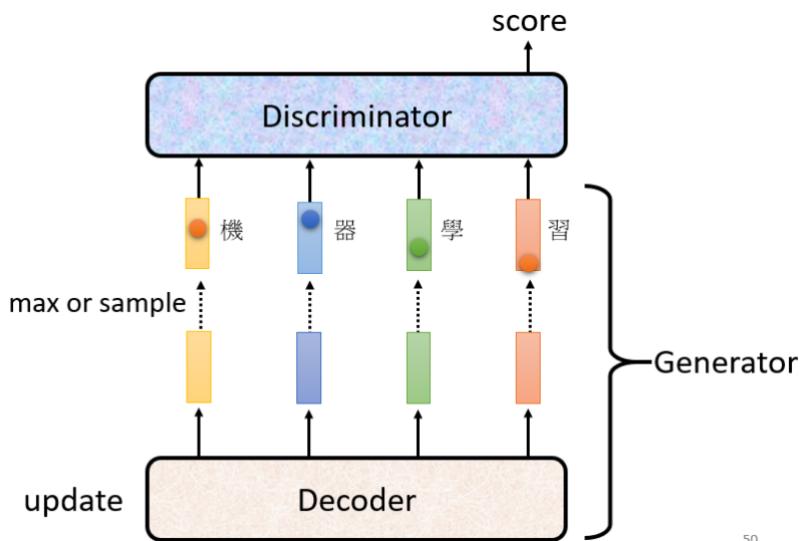
如果你要生成一段文字,那你可能会有一个,Sequence To Sequence 的 Model,你有一个 Decoder



50

那这个 Decoder 会产生一段文字,那我们现在这个,Sequence To Sequence 的 Model就是我们的 Generator,这个在过去,在讲 Transformer 的时候,这是一个 Decoder,那它现在,在 GAN 裡面,它就扮演了 Generator 的角色,负责产生我们要它产生的东西,比如说一段文字

那你说这个会跟原来的 GAN,在影像上的 GAN 有什么不同? 就最 High Level 来看,就演算法来看,可能没有太大的不同,因为接下来,你就是训练一个 Discriminator



50

Discriminator 把这段文字读进去,去判断说这段文字是真正的文字,还是机器產生出来的文字,而 Decoder 就是想办法去骗过 Discriminator,Generator 就是想办法去骗过 Discriminator

你去调整你的这个 Generator 的参数,想办法让 Discriminator 觉得,Generator 產生出来的东西是真正的但是真正的的难点在於,你如果要用 Gradient Descent,去 Train 你的 Decoder,去让 Discriminator Output 分数越大越好,你会发现你做不到

大家知道,在用Gradient Descent方法中计算微分的时候,所谓的 Gradient,所谓的微分,其实就是某一个参数,它有变化的时候,对你的目标造成了多大的影响

我们现在来想想看,假设我们改变了 Decoder 的参数,也就是 Generator 的参数,有一点小小的变化的时候,到底对 Discriminator 的输出,有什麼样的影响

如果Decoder 的参数有一点小小的变化,那它现在输出的这个 Distribution,也会有小小的变化,那因為这个变化很小,所以它不会影响最大的那一个 Token

Token,可能会觉得有点抽象,那如果你要想得更具体一点,Token 就是你现在在处理这个问题,处理產生这个 Sequence 的单位,那 Token ,是人定的了

- 假设我们今天,在產生一个中文的句子的时候,我们是每次產生一个 Character,一个方块字,那方块字就是我们的 Token
- 那假设你在处理英文的时候,你每次產生一个英文的字母,那字母就是你的 Token
- 假设你一次,是產生一个英文的词,英文的词和词之间,是以空白分开的,那就是词就是你的 Token

所以 Token 的定义,是你自己决定的,看你要拿什麼样的东西

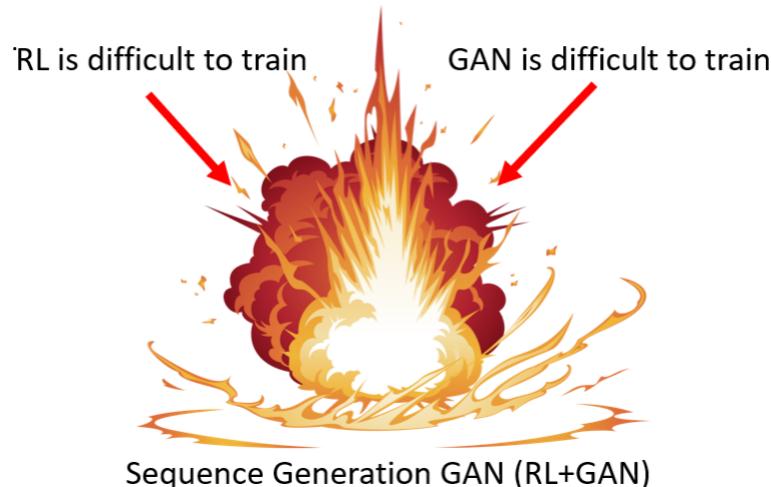
那今天,这个 Distribution 只有小小的变化,在取 Max 的时候,在找分数最大那个 Token 的时候,你会发现,分数最大的那个 Token 是没有改变的,你 Distribution 只有小小的变化,所以分数最大的那个 Token 是同一个,那对 Discriminator 来说,它输出的分数是一模一样的,这样输出的分数就没有改变

所以你根本就没有办法算微分,你根本就没有办法做 Gradient Descent

有同学可能会说,欸 这边不是 Max,是因為 Max 造成不能做 Gradient Descent 吗,那那个 CNN 裡面不是有那个 Max Pooling 吗,那怎麼还可以做 Gradient Descent? 这个问题就留给你自己深思一下,為什麼在这个地方,有 Max 不能做 Gradient Descent,而在CNN有 Max Pooling,却可以做 Gradient Descent

但是就算是不能做 Gradient Descent,你也不用害怕,记不得我们上週有讲说,遇到不能用 Gradient Descent Train 的问题,就当做 Reinforcement Learning 的问题,硬做一下就结束了,所以你确实可以用 Reinforcement Learning,来 Train 你的 Generator,在你要產生一个 Sequence 的时候,你可以用 Reinforcement Learning,来 Train 你的 Generator

Reinforcement learning (RL) is involved

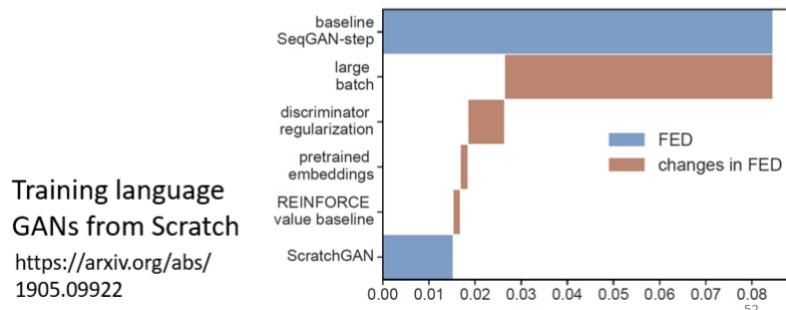


但 Reinforcement Learning 是以难 Train 而闻名, GAN 也是以难 Train 而闻名, 这样的东西加在一起, 就大炸裂这样 Train 不起来, 非常非常地难训练

所以要用 GAN 产生一段文字, 过去一直被认为是一个非常大的难题, 所以有很长一段时间, 没有人可以成功地把 Generator 训练起来产生文字, 通常你需要先做 Pretrain, 那 Pretrain 这件事情, 其实我们等一下马上就会提到, 如果你现在还不知道 Pretrain 是什么的话, 也没有关系, 总之过去你没有办法用正常的方法, 让 GAN 产生一段文字

直到有一篇 Paper 叫做 ScrachGAN

- Usually, the generator are fine-tuned from a model learned by other approaches.
- However, with enough hyperparameter-tuning and tips, ScrachGAN can train from scratch.



它的 Title 就开宗明义跟你炫耀说, 它可以 Train Language GANs From Scrach, **From Scrach 就是不用 Pretrain 的意思**

它可以直接从随机的初始化参数开始 Train 它的 Generator, 然后让 Generator 可以产生文字, 它最关键的就是强调 Hyperparameter, 跟一大堆的 Tips

比如说, 这个横轴是它们的 Major, 这个叫做 FED, 那这个是用在文字上的, 我们今天就不讲, 这不重要, 总之这个值越低越好

- 一开始要有一个叫做 SeqGAN-Step 的技术, 没这个完全 Train 不起来
- 然后接下来有一个很大的 Batch Size, 通常就是上千, 没有那个, 你自己在家没办法这么做的
- Discriminator 加 Regularization, Embedding 要 Pretrain
- 改一下 Reinforcement Learning 的 Argument
- 最后就有 ScrachGAN, 就可以从真的把 GAN Train 起来, 然后让它来产生 Sequence

那今天有关 GAN 的部分, 我们只是讲了一个大概, 那如果你想要学最完整的内容, 我在这边留下一个连结给大家参考,

GAN_Full Version & Other Generative Models

- This lecture: Generative Adversarial Network (GAN)



Full version

https://www.youtube.com/playlist?list=PLJV_el3uVTsMq6JEFPW35BCiOQTsoqwNw

那其实有关 Generative 的 Model,不是只有 GAN 而已,还有其他的,比如说 VAE,比如说 FLOW-Based Model,那我在这边也列了两个影片的连结,给大家参考

Variational
Autoencoder (VAE)



<https://youtu.be/8zomhgKrsmQ>

FLOW-based
Model



<https://youtu.be/uXY18nzdSsM>

强调一下就是,这边的影片连结并不是一定要看过这些影片连结,才能够学习接下来的内容,因为机器学习可以讲的东西实在太多了,所以如果,假设你没有太多的时间,那你唯一真正需要听的,上课讲的内容是 Self Content,它本身是 Consistent 的,你只要每一堂课都有听,你接下来的内容,你应该都可以依序听下去,应该都可以听懂

然后在上课中,会放一些影片的连结,这个就等於是额外分出去的分支,如果你真的很有兴趣的话,可以进行深入的研究

那為什麼我们不再讲更多东西,因为在上课的设计,这个课程的内容,是以真正能够对你有帮助,以实务为导向的,就假设你想要 Train 一个 Generator,你想让机器可以产生东西,你有很多方法,你可以用 GAN,你可以用 VAE,可以用 FLOW-Bases Model

我们这边就选择告诉你 GAN,所以以后,你如果有人叫你,Train 一个 Generative Model,你有办法去 Train 它,那你如果想要深入研究,你可以在研究 VAE 跟 FLOW-Bases Model,那有人可能会问说,為什麼选择 GAN,為什麼不是选择其他的 Model 一个最直接的理由是,GAN 的 Performance 是比较好的

如果你要产生非常好的图片的话,你还是今天要用 GAN,通常 VAE 或 FLOW-Bases Model,它们产生的结果,都是跟 GAN 有非常大的一段差距,它们通常都是 Plan 说,我经过了一番努力,暴调了一堆参,爆弄了一堆 Tips,最后可以跟 GAN 差不多而已,所以 GAN 通常,它产生的结果还是比较好的

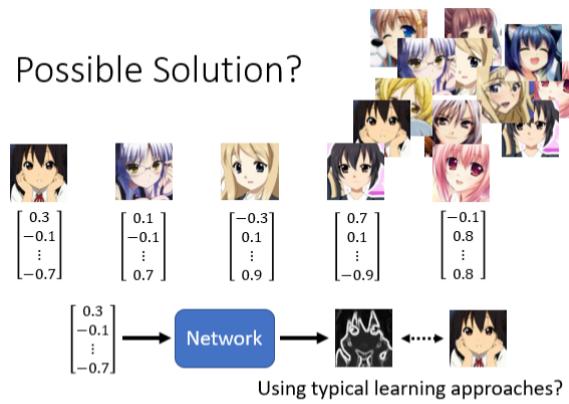
那你可能会说,GAN 比较难 Train,这个比较 Train 吧,VAE 或 FLOW 会不会比较好 Train?

如果你真的有实做 VAE 或 FLOW 的话,它们没有比较好 Train 老实说,你可能会觉得说,从它的式子上看起来,GAN 很神祕,有一个 Discriminator Generator,它们要互动,然后像 FLOW-Bases Model VAE,它们都比较像是,直接 Train 一个一般的模型,它们有一个很明确的 Objective

但你实际上 Train 起来发现说,它们也没有那麼容易成功地被训练起来,它们的 Objective 裡面有很多项,它们的 Loss 裡面有很多项,然后把每一项都平衡,才能够有好的结果,但要达成平衡也非常地困难,跟 GAN,我觉得 Train 的难度是不遑多让,所以我们这边就选择 GAN ,作為我们课堂上介绍的,生成式的 Generative 的 Model,那至於其他 Model,你可以再多多,如果你有兴趣,你可以再自己涉猎

也许有同学会想说,為什麼我们要特別用一些,提出一些新的做法,来做 Generative 这件事?

如果我们今天的目标就是,输入一个 Gaussian 的 Random 的 Variable,输入一个 Gaussian,从 Gaussian 的这个 Random Variable,Sample 出来的 Vector,把它变成一张图片,那我们能不能够用,Supervised Learning的方法来做?



Generative Latent Optimization (GLO), <https://arxiv.org/abs/1707.05776>
Gradient Origin Networks, <https://arxiv.org/abs/2007.02798>

也就说我有一堆图片,我把这些图片拿出来,每一个图片都去配一个 Vector,都去配一个,从 Gaussian Distribution,Sample 出来的 Vector, 接下来就当做 Supervised Learning 的方法,硬做就结束了,

这张图片就是对到这个 Vector,Train 一个 Network,输入一个 Vector,输出就是它对应的图片,把对应的图片当做你训练的目标,训练下去

真的有这样子的生成式的模型,那难的点是说,如果这边,纯粹放随机的向量,Train 起来结果会很差,你可能根本连 Train 都 Train 不起来,所以怎麽办,你需要有一些特殊的方法,我在这边一样放两篇论文的连结

Generative Latent Optimization (GLO), <https://arxiv.org/abs/1707.05776>

Gradient Origin Networks, <https://arxiv.org/abs/2007.02798>

Evaluation of Generation

我们现在產生出来的 Generator,它好或者是不好,那要评估一个 Generator 的好坏,并没有那麼容易,那最直觉的做法,也许是找人来看,你要知道,今天这个 Generator 產生出来的图片,到底像不像动画的人物,那就找人直接来看,也许就结束了

其实很长一段时间,尤其是人们刚开始研究,Generative 这样的技术的时候,很长一段时间没有好的 measure,那时候要评估 Generator 的好坏,都是人眼看,然后直接用吹的这样

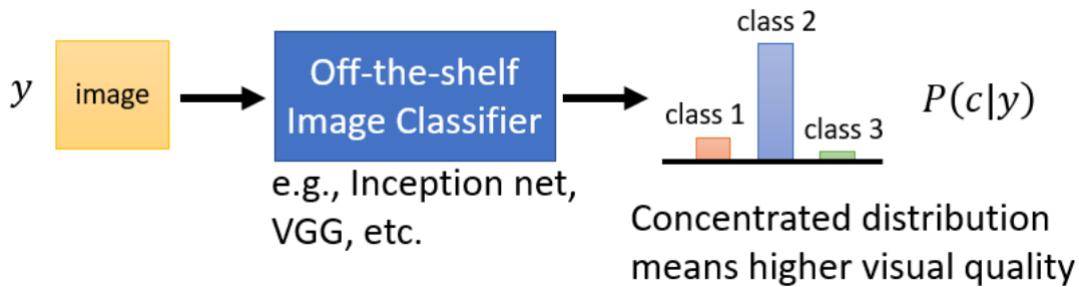
就说在 Paper 最后就放几张图说,你看这个,我觉得应该是比文献上,目前结果都还要好,太棒了,这应该是 state of the art,然后就结束了这样子,所以发现比较早年的 GAN 的 Paper,它没有数字,整篇 Paper 裡面没有 Accuracy,它就是放几张图片告诉你说,这个应该是比过去的文章都好,然后就结束了

完全用人来看显然有很多的问题,比如说不客观,不稳定等等诸多的问题,所以有没有比较客观,而且自动的方法,来想办法量一个 Generator 的好坏,如果针对特定的一些任务,是有办法设计一些方法的,

Other Models to Evaluate

如果是一般的 Case ,如果我们不侷限在我们的作业,跟一般的 Case,我随便训练了一个 Generator,它不一定是產生动画人物的,因為它產生别的,它专门產生猫,专门產生狗,专门產生斑马等等

那有一个方法,是一样跑一个影像的分类系统,把你的 GAN 产生的图片,丢到一个的影像的分类系统裡面,看它产生什麼样的结果



影像分类系统输入是一张图片,我们这边叫做 y ,输出,是一个机率分布,我们这边叫它 $P(c|y)$, $P(c|y)$ 是一个机率的分布

然后接下来我们就看说,这个机率的分布如果越集中,就代表说现在产生的图片可能越好,虽然我们不知道这边产生的图片,裡面有什麼东西,不知道它是猫还是狗还是斑马,我们不知道它是什么,但是如果丢到了一个影像分类系统以后,它输出来的结果,它输出来的这个分布非常集中,代表影像分类系统,它非常肯定,它现在看到什麼样的东西

它非常肯定它看到了狗,它非常肯定它看到了斑马,然后代表说,你产生的图片,也许是比較接近真实的图片,所以影像辨识系统才辨识得出来

如果你产生的图片是一个四不像,根本看不出是什麼动物,那影像辨识系统就会非常地困惑,它产生的这个机率分布,就会非常地平坦,非常地平均分布,那如果是平均分布的话,那就代表说你的 GAN,产生的图片,可能是比較奇怪的,所以影像辨识系统才会辨识不出来

所以这个是靠影像辨识系统,来判断你产生的图片好坏,这是一个可能的做法

Diversity - Mode Collapse

但是光用这个评估的方法会被一个,叫做 Mode Collapse 的问题骗过去,Mode Collapse 是说,你在 Train GAN 的时候,你有时候 Train 著 Train 著,就会遇到一个状况是



- 假设这些蓝色的星星,是真正的资料的分布
- 红色的星星是你的 GAN,你的 Generative 的 Model,它的分布

你会发现说 Generative Model,它输出来的图片来来去去,就是那几张,可能单一张拿出来,你觉得好像还做得不错,但让它多產生几张就露出马脚

那以下是一个 Mode Collapse 的例子啦,就是我们在这个上週有看到说,我就 Train 了一个 Generator,让它產生二次元的人物,那 Train 著 Train 著 Train 到最后,我就发现变成这样的一个状况,这一张脸越来越多



越来越多,而且它还有不同的髮色,这个髮色比较偏红,这个髮色比较偏黄,越来越多,最后就通通都是这张脸,那这就是一种 Mode Collapse 的现象

那為什麼会有 Mode Collapse,这种现象发生,就直觉上你还是比较容易理解,你可以想成说,这个地方就是 Discriminator 的一个盲点,当 Generator 学会產生这种图片以后,它 Discr,它就永远都可以骗过 Discriminator,Discriminator 没办法看出说,这样子的图片是假的,那这是一个 Discriminator 的盲点,Generator 抓到这个盲点就硬打一发,就发生 Mode Collapse 的状况

那可是到底要怎麼避免Mode Collapse 的状况,我认为今天其实还没有一个非常好的解答,举例来说,我们在上週给大家看到了,BGAN 的结果,就是会產生网球狗那个结果,那是 Google 做的,它也爆收了参数,但就算是它爆收了参数,它发现最终,它仍然没有办法真的避免,Mode Collapse 的状况,就 BGAN Train 到最后,还是 Mode Collapse

BGAN 那边 Paper 怎麼解决这个问题,其实很简单,Model 在 Generator 在训练的时候,一路上都会把 checkpoint 存下来,在 Mode Collapse 之前,把 Training 停下来,然后就把之前的 Model 拿出来用,就结束了这样,所以就算是强如 Google 爆收参数,现在还是没有办法彻底解决,Mode Collapse 的问题

Diversity - Mode Dropping

但是有另外一种更难被侦测到的问题,叫做 Mode Dropping,Mode Dropping 的意思是说,你的真实的资料分布可能是这个样子,但是你的產生出来的资料,只有真实资料的一部分,单纯看產生出来的资料,你可能会觉得还不错,而且分布,它的这个多样性也够



但你不知道说真实的资料,它的多样性的分布,其实是更大的,我这边举一个例子,好 那这边,是一个真实的例子,就有个同学,他 Train 了这个人脸生成的 GAN,那它在某一个 Iteration 的时候,它的 Generator 產生出这些人脸



Generator
at iteration t

你会觉得说,没有问题,而且人脸的多样性也够,有男有女,有向左看,有向右看,各式各样的人脸都有,好 这是第 T 个 Iteration 的时候 Generator,你也不觉得,它的多样性有问题,但如果你再看下一个 Iteration,Generator 產生出来的图片是这样子的



Generator
at iteration t+1

(BEGAN on CelebA)

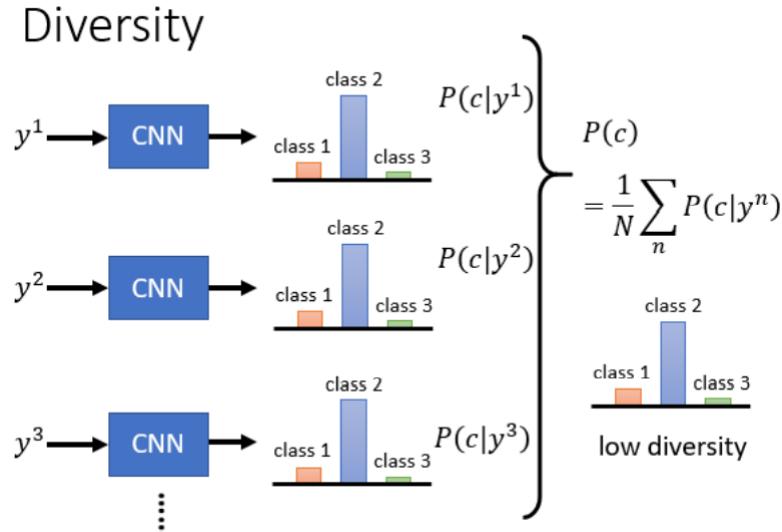
它的肤色有问题,所以它之前,你看有男有女没有问题,但是它肤色偏白,这边肤色偏黄,你没弄好人家都觉得,你的 Generator 有种族歧视

所以在这种 Mode Dropping 的问题是,不太容易被侦测出来的,事实上今天到底,今天这些非常好的 GAN,BGAN,Progress GAN,BGAN,Progress GAN,可以产生非常真实人脸这些 GAN,到底有没有 Mode Dropping 的问题,可能还是有的

如果你看多了,GAN 产生出来的人脸,你会发现说,虽然非常真实,但好像来来去去,就是那麽几张脸而已,它有一个非常独特的特徵是,你看多了以后就觉得,这个脸好像是被生成出来的,所以今天也许 Mode Dropping 的问题,都还没有获得本质上的解决

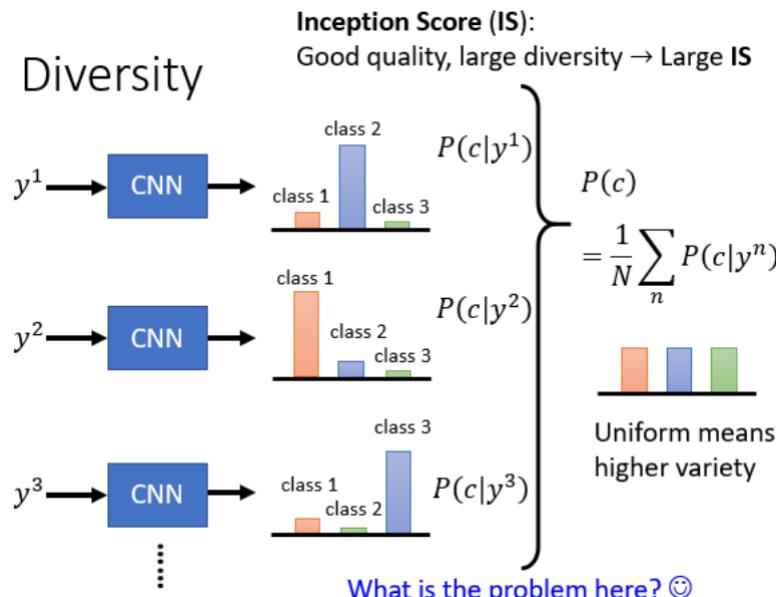
但是我们会需要去量说,现在我们的 Generator,它产生出来的图片,到底多样性够不够

过去有一个做法,一样是藉助我们的 Image Classifier,你就把一堆图片,就很像你的 Generator 产生 1000 张图片,把这 1000 张图片裡,都丢到 Image Classify 裡面,看它被判断成哪一个 Class



每张图片,都会给我们一个 Distribution,你把所有的 Distribution 平均起来,接下来看看平均的 Distribution 长什麼样子

如果平均的 Distribution 非常集中,就代表在多样性不够,如果什麼图片丢进去,你的影像分类系统都说,是看到 Class 2,看到裡面有 Class 2 这样的东西,那代表说,每一张图片也许都蛮像的,你的多样性是不够的
那如果另外一个 Case,不同张图片丢进去,不同张,你的 Generator 产生的图片,丢到 Image Classifier 的时候,它产生的输出的分布,都非常地不同



你平均完以后发现,平均完后的结果是非常平坦的,那这个时候代表什麼,这个时候代表说,也许你的多样性是足够的,那你会发现在评估的标準上

当我们用这个 Image 的 Classifier, 来做评估的时候, Diversity 跟 Quality 好像是有点互斥的, 因为我们刚才在讲 Quality 的时候, 我们说越集中代表 Quality 越高, 但是 Diversity 分布越平均, 代表 Diversity 越大

Inception Score

强调一下这个 Quality 跟 Diversity, 它们评估的范围不一样

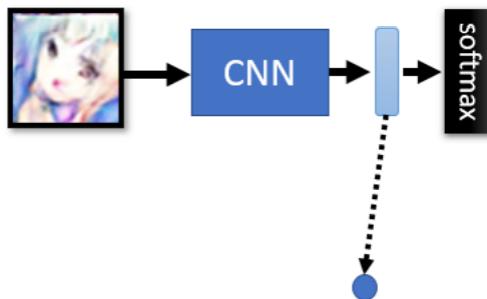
- Quality 是只看一张图片, 一张图片丢到 Classifier 的时候, 分布有没有非常地集中
- 而 Diversity 看的是一堆图片, 它分布的平均, 一堆图片你的 Image Classifier, 如果输出的平均越平均的话, 就代表说现在的 Diversity 越大

那过去有一个常常被使用的分数, 叫做 Inception Score, 那它的缩写是 IS, 所谓 Inception Score, 顾名思义就是这边用的这个基于 CNN 的 Inception 模型来做的, 所以叫 Inception Score

用 Inception Network 量一下 Quality, 如果 Quality 高, 那个 Diversity 又大, 那 Inception Score 就会比较大

Fréchet Inception Distance (FID)

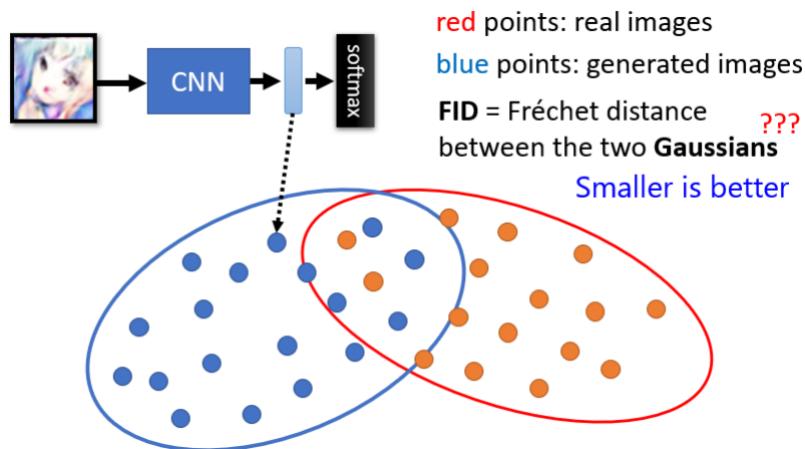
在我们的作业中, 会採取另外一个 Evaluation 的 Measure, 叫 Fréchet Inception Distance, 它的缩写叫做 FID, 这个东西是什麼, 你先把你產生出来的二次元的人物, 丢到 Inception Net 裡面



把这个二次元人物一路丢到最后, 让那个 Inception Network 输出它的类别, 那你得到的可能就是人脸, 那每一张二次元的人物看起来都是人脸, 那我们不要拿那个类别

我们拿进入 Softmax 之前的 Hidden Layer 的输出, 进入 Softmax 之前, 你的 Network 会产生一个向量, 那可能是长度是上千维的一个向量, 把那个向量拿出来, 代表这张图片

那如果我们拿出来的是一个向量, 而不是最后的类别, 那虽然最后分类的类别可能是一样的, 但是在决定最后的类别之前, 这个向量就算都是人脸, 可能还是不一样的, 可能会随著肤色 髮型, 这个向量还是会有所改变的, 所以我们就不取最后的类别, 只取这个 Inception Network 中间的, 其实是最后一层的这个 Hidden Layer 的输出, 来代表一张图片



所有红色的点, 代表你把真正的图片, 丢到 Inception Network 以后, 拿出来的向量, 那这个向量其实非常高维度, 是上千维的, 我们就把它假设, 我们可以把它画在二维的平面上,

蓝色的点是你自己的 GAN,你自己的 Generator,產生出来的图片,它丢到 Inception Network 以后,进入 Softmax 之前的向量,把它画出来,假设是长这个样子

接下来,假设真实的图片跟產生出来的图片它们都是 Gaussians 的 Distribution,然后去计算这两个 Gaussians Distribution 之间的Fréchet Distance,就结束了

那至於 Fréchet 的 Distance 是什麼,你有兴趣再自己看一下文献,反正在作业裡面,我们的 Judge System 会帮大家算好

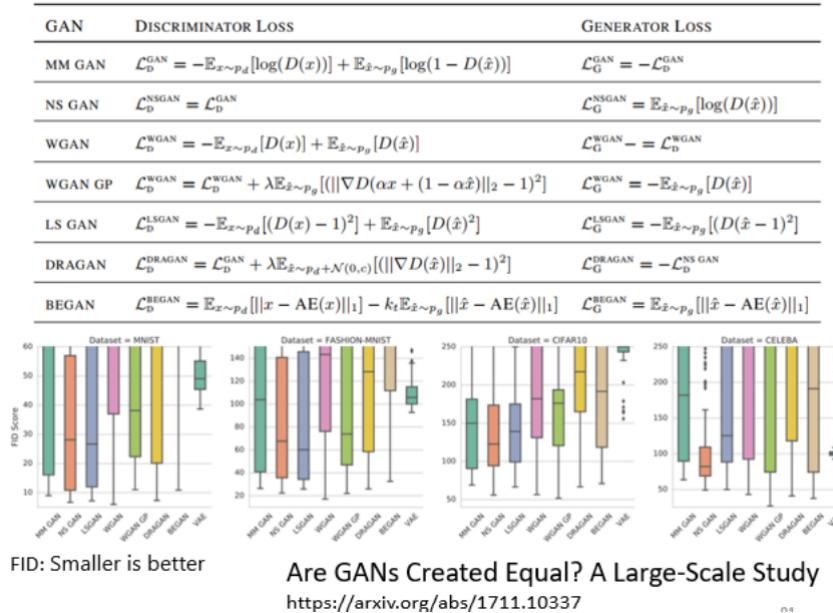
因為它是一个 Distance,所以这个值就是越小越好,距离越小,代表这两组图片越接近,那当然就是產生出来的品质越高

但这边你一定心裡还是有很多问号

- 第一个问号就是,当做 Gaussians Distribution 没问题吗,这个应该不是 Gaussians Distribution 吧?
会有问题!
- 然后另外一个问题是,如果你要準确的得到你的 Network 它的分布,那你可能需要產生大量的 Sample 才能做到,那这需要一点运算量,那这个也是要做 FID 不可避免的问题

所以其实我们在作业裡面,我们不会只看,我们也不会只看 FID,只看 FID,其实结果会怪怪的,怪怪的 因為你,你假设你的这个输出的分布一定是 Gaussians ,那它实际上不是 Gaussians,硬假设它是 Gaussians,没有怪怪的吗,会怪怪的,所以我们是同时看 FID,跟动画人物人脸的这个,侦测出来的人脸的数目,这两个指标,我会同时看这两个指标,那这样可以得到比较合理而精确的结果

FID 算是今天比较常用的一种 Measure,那有一篇 Paper 叫做,Are GANs Created Equal,A Large Scale Study



那你可以想见说这个也是 Google 做的啦,那就是爆做了各式各样不同的 GAN,有,那个时候它就列举了好多不同的,各式各样的 GAN,那每一个 GAN,当然它的这个训练的这个 Objective,训练的那个 Loss 有点不太一样,我这边就不细讲,各式各样的 GAN,每一种 GAN,它都用不同的 Random Seed,去跑过很多次以后,看看结果怎麽样

上面这个图,就是在四个不同的资料库上面得到的结果

横轴这边代表的是不同的 GAN,那这边的值FID,是越小越好

你会发现说,这边每一个方法,它都不是只得到一个数值,它都得到一个分布,為什麼它得到是一个分布,因為你要用那个不同的 Random Seed 去跑,每次跑出来的结果都不太一样,那这边混了一个不是 GAN 的做法,混了一个 VAE 在这裡

那你会发现说,如果比较这些 GAN 的方法跟 VAE 的方法,VAE 的方法显然是比较稳定的,不同的 Random Seed,看起来差距还是比较小的,那 GAN 的方法,不同 Random Seed 差距是很大的,那你又可以很明显地看出,VAE 跟 GAN,它的这个好的程度,不在同一个量级上,GAN 可以产生远比 VAE 更好的结果

不过你会发现说不同的 GAN 好像结果差不多,所以这边就,那抬头就是 Are GANs Created Equal,然后看起来所有的 GAN 都差不多,

如果你仔细看那篇文章的话,在做实验的时候,所有这些不同的 GAN 用的 Network 架构都是同一个,它只是爆收了那个,Random Seed 跟 Learning Rate 而已,所以 Network 架构还是同一个

所以我们不知道是不是有某些 Network 架构,特别 Favor 某些种类的 GAN,或者是某些种类的 GAN,会不会在不同的 Network 架构上,表现得比较稳定,比如说如果你看 WGAN 的话,WGAN 最原始的 Paper,它标榜的其实是它 Network 架构胡乱设计,它胡乱兜个什麼 100 层的 Generator,就很没有必要弄一个 100 层的 Generator,它也 Train 得起来,所以也许 WGAN 是在不同的 Generator,不同的 Network 架构的时候比较稳定,那你试不同的 Random Seed,可能没有特别稳定等等之类的,不知道,这篇 Paper 并没有给我们这方面的答案

We don't want memory GAN.

那其实刚才那些 Measure 也完全,也并没有完全解决 GAN 的 Evaluation 的问题

你想想看以下的状况,假设这是你的真实资料



你不知道怎麼回事,训练了一个 Generator,它产生的 Data,跟你的真实资料一模一样

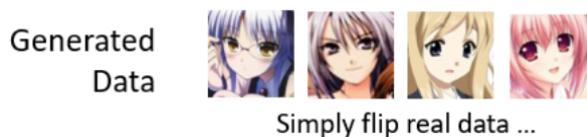
所以如果你不知道真实资料长什麼样子,你光看这个 Generator 的输出,你会觉得太棒了,它做得很棒,那 FID 算出来,一定是非常小的

但问题是这个是你要的吗,如果它产生的图片都跟资料库裡面的,训练资料的一模一样,训练资料就在你手上,直接从训练资料裡面,Sample 一些 Image 出来不是更好,干嘛要 Train Generator

我们 Train Generator 其实是希望它产生新的图片,训练资料裡面没有的人脸,如果训练资料裡面有一模一样的人脸,直接用训练资料裡面的人脸就好了,何必用 GAN,所以有时候你的 GAN 产生的结果很好,也许你在作业裡面,FID 算出来也很低,然后人脸辨识系统也给你很高的分数,但是它不一定是一个好的 GAN

你可能会说,那我们就把,我们 Generator 产生的图片,跟真实资料比个相似度吧,看看是不是一样嘛,如果很多张都一样就代表说,Generator 只是把那个训练资料背起来而已,它没有很厉害

但是那如果我问另外一个问题,假设你的 Generator 学到的是,把所有训练资料裡面的图片都左右反转,那它也是什麼事都没有做



Simply flip real data ...

假设它学到就是,把训练资料裡面所有的图片都左右翻转,那你会觉得,嗯 它看起来很棒,它实际上也是什麼事都没有做,但问题是比相似度的时候,又比不出来,所以 GAN 的 Evaluation 是非常困难的,还甚至光要如何评估,一个 Generator 做得好不好这件事情,都是一个可以研究的题目

如果你真的很有兴趣的话,这边放了一篇相关的文章啦<https://arxiv.org/abs/1802.03446>,裡面就列举了二十几种,GAN Generator 的评估的方式

Measure	Description
1. Average Log-likelihood [18, 22]	• Log likelihood of explaining realworld held out/test data using a density estimated from the generated data
2. Coverage Metric [34]	• The probability mass of the true data ‘covered’ by the model distribution $C := P_{\text{Model}}(\ d\ _{\text{Model}} > t)$ with t such that $P_{\text{Model}}(\ d\ _{\text{Model}} > t) = 0.95$
3. Inception Score (IS) [3]	• KLD between conditional and marginal label distributions over generated data. $\exp\left[\mathbb{E}_{\mathbf{x}}\left[\mathbb{E}_{\mathbf{y} \mathbf{x}}\left[\text{KL}\left(p(\mathbf{y} \mathbf{x}) \parallel p(\mathbf{y})\right)\right]\right]\right]$
4. Modified Inception Score (m-IS) [34]	• Similar to IS but also takes into account the prior distribution of the labels over real data. $\exp\left[\mathbb{E}_{\mathbf{x}}\left[\mathbb{E}_{\mathbf{y} \mathbf{x}}\left[\text{KL}\left(p(\mathbf{y} \mathbf{x}) \parallel p(\mathbf{y}) \parallel p(\mathbf{y}^{\text{prior}})\right)\right]\right]\right] - \text{KL}\left(p(\mathbf{y}) \parallel p(\mathbf{y}^{\text{prior}})\right)$
5. Mode Score (MS) [34]	• Similar to IS but also takes into account the prior distribution of the labels over real data. $\exp\left[\mathbb{E}_{\mathbf{x}}\left[\mathbb{E}_{\mathbf{y} \mathbf{x}}\left[\text{KL}\left(p(\mathbf{y} \mathbf{x}) \parallel p(\mathbf{y})\right)\right]\right]\right] - \text{KL}\left(p(\mathbf{y} \mathbf{x}) \parallel p(\mathbf{y})\right)$
6. AM Score [36]	• Wasserstein-1 distance between the empirical distribution of the labels over real labels vs. predicted labels, as well as the entropy of prediction. $\text{KL}(p(\mathbf{y}^{(true)}) \parallel p(\mathbf{y})) + H(p(\mathbf{y}))$
7. Fréchet Inception Distance (FID) [37]	• Wasserstein-2 distance between multi-variate Gaussians fitted to data embedded into a feature space
8. Maximum Mean Discrepancy (MMD) [18]	• Wasserstein-2 distance between two samples drawn independently from each distribution. $M_d(P_1, P_2) = \ p_1 - p_2\ ^2 + Tr(\Sigma_1 + \Sigma_2 - 2\Sigma_1\Sigma_2^{-1}\Sigma_2)$
9. The Wasserstein Critic [38]	• The critic (<i>e.g.</i> an NN) is trained to produce high values at real samples and low values at generated samples $W_{\text{GAN}}(\mathbf{x}_1, \mathbf{x}_2) = \frac{1}{N} \sum_{i=1}^N \mathbb{E}_{\mathbf{x} \sim P_{\text{real}}} f_i(\mathbf{x}_1) - \frac{1}{N} \sum_{i=1}^N \mathbb{E}_{\mathbf{x} \sim P_{\text{gen}}} f_i(\mathbf{x}_2)$
10. Birthday Paradox Test [27]	• Answers whether two samples are drawn from the same distribution by counting the duplicates (many duplicates)
11. Classifier Two Sample Test (C2ST) [4]	• An indirect technique for evaluating the quality of unsupervised representations (<i>e.g.</i> , feature extraction, FID score). See also the GAN Quality Index (GQI) [41]
12. Classification Performance [1, 15]	• An indirect technique for evaluating the quality of unsupervised representations (<i>e.g.</i> , feature extraction, FID score). See also the GAN Quality Index (GQI) [41]
13. Boundary Distortion [42]	• Number of Statistically-Different Bits (NDDB) [43]
14. Image Retrieval Performance [44]	• Compares the generated images to the underlying data manifold in terms of the number of nearest neighbors (<i>i.e.</i> , density).
15. Generative Adversarial Metric (GAM) [1]	• Compares the generated images against each other in a battle-like setting using discriminators generators. $p(\mathbf{x} y=1; M_1)p(\mathbf{y} y=1; M_2) = (p_{\mathbf{y}}=1 x)D_1(p(x G_1)) / (p_{\mathbf{y}}=1 x)D_2(p(x G_2))$
17. Tournament Win Rate and Skill Rating [45]	• Implements a tournament in which a player is either a discriminator that attempts to distinguish between real and generated data or a generator that attempts to fool the discriminator.
18. Normalised Relative Discriminative Score (NRDS) [3]	• Compares the generated images to the underlying data manifold between real and generated data.
19. Adversarial Accuracy and Divergence [46]	• Evaluates the quality of generated images using measures such as SDR, PSNR, and sharpness difference.
20. Geometry Score [47]	• Compares the generated images to the underlying data manifold using the Euclidean distance between the generated image to optimizing for z (<i>i.e.</i> , $\min_{\mathbf{z}} \ G(\mathbf{z}) - \mathbf{x}^{\text{target}}\ ^2$)
21. Reconstruction Error [48]	• These measures are used to quantify the degree of overfitting in GANs, often over toy datasets.
22. Image Quality Measures [49, 50, 51]	• To detect overfitting, generated samples are shown next to their nearest neighbors in the training set
23. Low-level Image Statistics [52, 53]	• In these experiments, participants are asked to distinguish generated samples from real images
24. Precision, Recall and F1 score [32]	• Participants are asked to rank models in terms of the fidelity of their generated images (<i>e.g.</i> , pairs, triplets)
5. Network Internals [1, 60, 61, 62, 63, 64]	• Over datasets with known modes (<i>e.g.</i> a GMM or a labeled dataset), modes are computed as by measuring the mean of the generated samples in the neighborhood of each mode.
Quantitative	• Results exploring and illustrating the internal representation and dynamics of models (<i>e.g.</i> , space continuity) as well as visualizing learned features

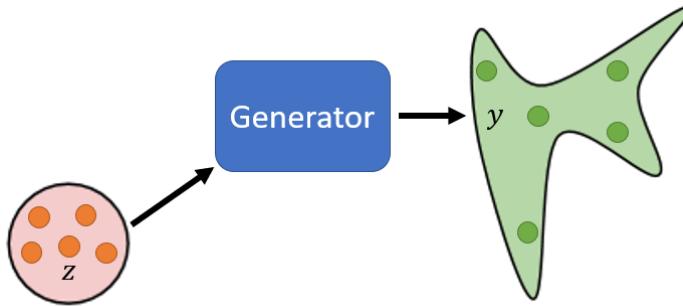
Pros and cons of GAN evaluation measures

<https://arxiv.org/abs/1802.03446>

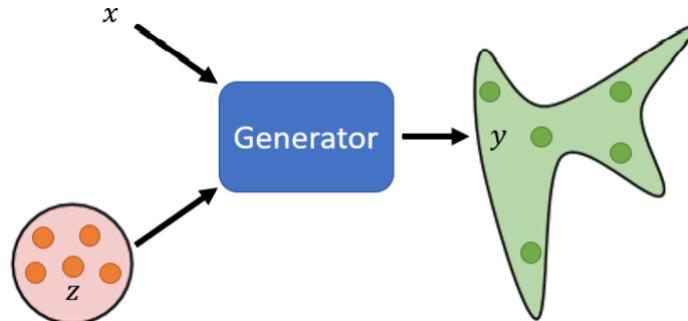
Conditional Generation

什麼是 Conditional 的 Generation?

刚才我们讲的那个 Generator,到目前为止我们讲的 Generator,它输入都是一个随机的分布而已,那这个不见得非常有用

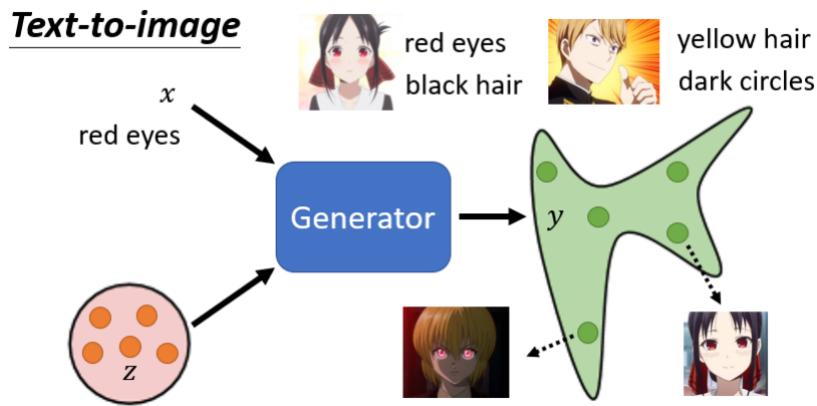


我们现在想要更进一步的是,我们可以操控 Generator 的输出,我们给它一个 Condition x,让它根据 x 跟 z 来產生 y,那这样的 Conditional Generation



有什麼样的应用,比如说你可以做文字对图片的生成

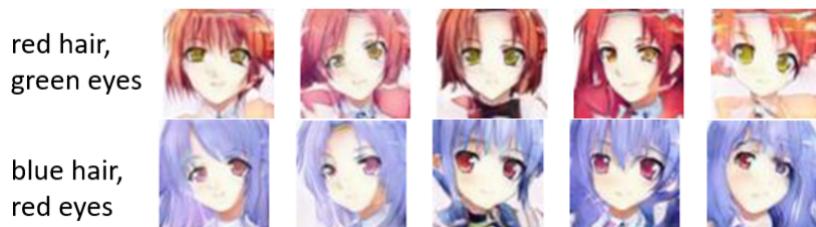
那如果你要做文字对图片的生成,它其实是一个 Supervised Learning 的问题,你需要一些 Label 的 Data,你需要去蒐集一些图片,蒐集一些人脸,然后这些人都要有文字的描述,告诉我们说,这个是红眼睛,这个是黑头髮,这个是黄头髮,这个是有黑眼圈等等,告诉我们这样子,我们要这样的 Label 的资料,才能够训练这种 Conditional 的 Generation



所以在 Text To Image 这样的任务裡面,我们的 x 就是一段文字,那你可能问说,一段文字怎麽输入给 Generator ,那就要问你自己了,你要怎麽做都可以

以前会用 RNN 把它读过去,然后得到一个向量,再丢到 Generator,今天也许你可以把它丢到一个 Transformer 的 Encoder 裡面去,把 Encoder Output 这些向量通通平均起来,丢到 Generator 裡面去,怎麽样都可以 反正,你用什麼方法都可以,只要能够让 Generator 读一段文字就行

那你期待说你输入 Red Eyes,然后,机器就可以画一个红眼睛的角色,但每次画出来的角色都不一样,那这个画出来什麼样的角色,取决於什麼,取决於你 Sample 到什麼样的 z ,Sample 到不一样的 z ,画出来的角色就不同,但是通通都是红眼睛的,这个就是 Text To Image 想要做的事情

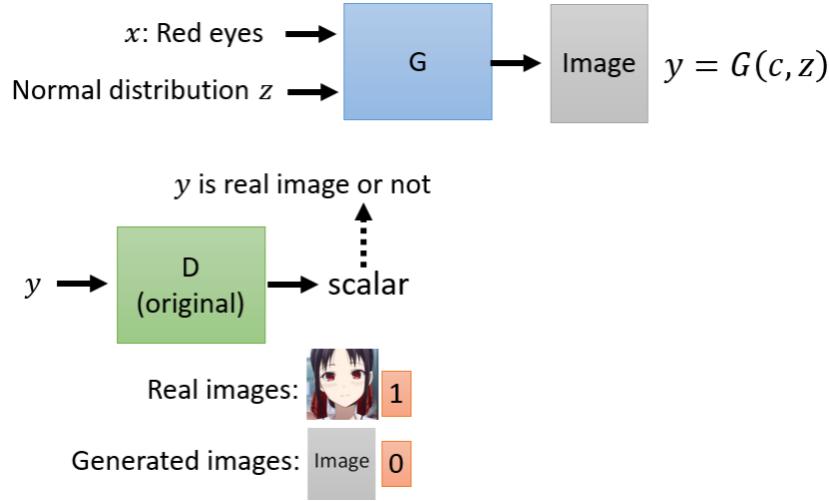


这学期虽然没有,但过去有这个作业,就是输入红头髮,这个是之前助教做的结果,输入红头髮,输入绿眼睛,那產生的结果就是这个样子,產生各式各样红头髮 绿眼睛的角色,输入蓝头髮 红眼睛,就產生各式各样蓝头髮 红眼睛的角色,你发现,那个有时候机器也是会犯错的啦,比如说这边有一个异色瞳,虽然说要画红眼睛,但它觉得画一隻红色的眼睛就可以矇混过去,另外一隻眼睛仍然是蓝色的

我们现在的 Generator 有两个输入,一个是从 Normal Distribution,Sample 出来的 z ,另外一个是 x ,也就是一段文字



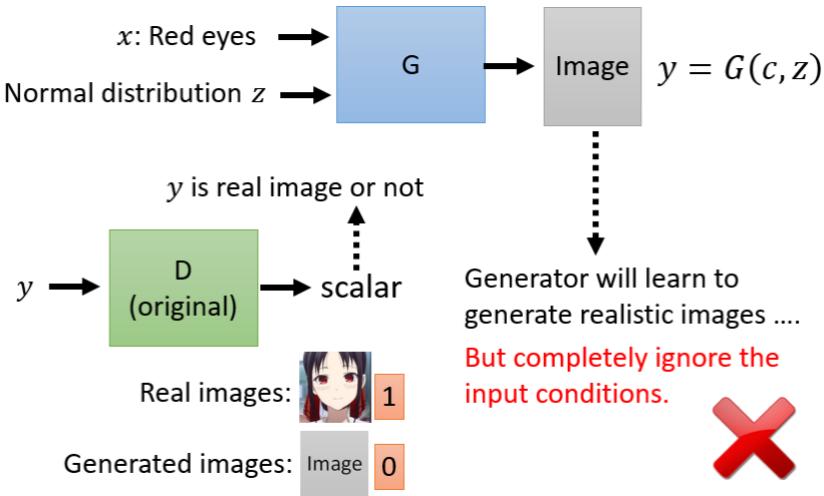
那我们的 Generator 会产生一张图片 y ,那我们需要一个 Discriminator,那如果按照我们过去所学过的东西,Discriminator,它就是吃一张图片 y 当作输入,输出一个数值,这个数值代表输入的图片,多像真实的图片



是真实的,还是生成的,那怎麽训练这个 Discriminator ,你就说如果看到真实的图片,你就输出 1,如果看到生成的图片,就输出 0,你就可以训练 Discriminator,然后 Discriminator 跟 Generator 反覆训练

也许你就可以去把 Generator 训练出来,但这样的方法,没办法真的解 Conditional GAN 的问题,為什麼,因為如果我们只有 Train 这个 Discriminator,这个 Discriminator 只会看 y 当做输入的话,那 Generator 会学到的是,它会产生可以骗过 Discriminator 的,非常清晰的图片

它会产生清晰的图片,但是跟输入完全没有任何关係,因為对 Generator 来说,它只要产生清晰的图片,就可以骗过 Discriminator 了,它何必要去管 Input 文字叙述是什麼



你的 Discriminator 又不看文字的叙述,所以它根本就不需要管文字的叙述,你不管输入什麼文字,就无视这个 x ,反正就是產生一个图片,可以骗过 Discriminator 就结束了,但这显然不是我们要的

所以在 Conditional GAN 裡面,你要做有点不一样的设计,你的 Discriminator 不是只吃图片 y ,它还要吃 Condition x

所以你的 Discriminator,它有 y 作为输入,有 x 作为输入,然后產生一个数值,那这个数值不只是看 y 好不好,光图片好,没有用,光图片好,Discriminator 还是不会给高分

Discriminator 给高分的时候,一方面图片要好,另外一方面,这个图片跟文字的叙述必须要是相配的,Discriminator 才会给高分

那怎麽样训练这样的 Discriminator ?

那你需要文字跟影像成对的资料,所以 **Conditional GAN,一般的训练,是需要这个 Pair 的 Data 的,是需要有标註的资料的,是需要成对资料的**



True text-image pairs: (red eyes,) 1

(red eyes,) 0

有这些成对资料,那你就告诉你的 Discriminator 说,看到这些真正的成对的资料,就给它一分,看到 Red Eyes,但是搭配,可能 Red Eyes 跟机器产生的出来的图片,那就是给 0 分,然后训练下去,就可以产生,就可以做到 Conditional GAN,

那其实在实作上,光是这样子,拿这样子的 Positive Sample,还有 Negative Sample,来训练这样的 Discriminator,其实你得到的结果往往不够好,光是告诉 Discriminator 说,这样子的状况是好的,这样子的状况是不好的,这样是不够的

你还需要加上一种不好的状况是,已经产生的好的图片,但是文字叙述配不上的状况

True text-image pairs: (red eyes,) 1

(red eyes,) 0 (red eyes,) 0

所以你通常会把你的训练资料拿出来,然后故意把文字跟图片乱配,故意配一些错的,然后告诉你的 Discriminator 说,看到这种状况,你也要说是不好的,用这样子的资料,你才有办法把 Discriminator 训练好,然后 Generator 跟 Discriminator,反覆的训练,你最后才会得到好的结果,这个就是 Conditional GAN

在目前的例子裡面都是,看一段文字产生图片,那 Conditional GAN 的应用,不只看一段文字产生图片啦,也可以看一张图片,产生图片

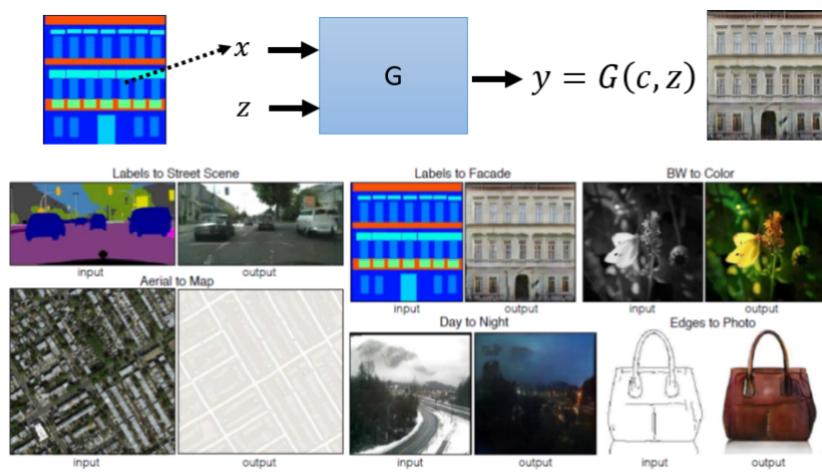


Image translation, or pix2pix

那看一张图片产生图片,也有很多的应用,比如说

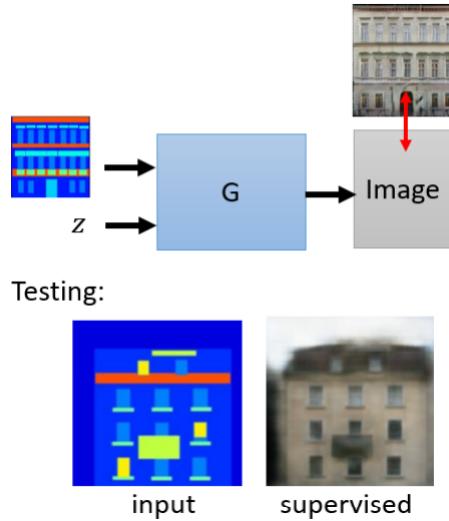
- 给它房屋的设计图,然后让你的 Generator 直接把房屋产生出来
- 给它黑白的图片,然后让它把颜色著上

- 给它这个素描的图,让它变成实景 实物
- 那给它这个白天的图片,让它变成晚上的图片
- 有时候你会给它,比如说起雾的图片,让它变成没有雾的图片,把雾去掉

所以 Conditional GAN,除了输入文字 產生影像以外,也可以输入影像 產生影像,那像这样子的应用,叫做 **Image Translation**,那有人又叫做 Pix2pix,这个 Pix 就是 Pixel,就是像素的缩写啦,所以叫做 Pix2pix

实现以上效果跟刚才讲的从文字產生影像,没有什麼不同,现在只是从影像產生影像,把文字的部分用影像取代掉而已,那当然同样的做法,同样要產生这样的 Generator,產生一张图片,输入一张图片 產生一张图片,你当然可以用 Supervised Learning 的方法

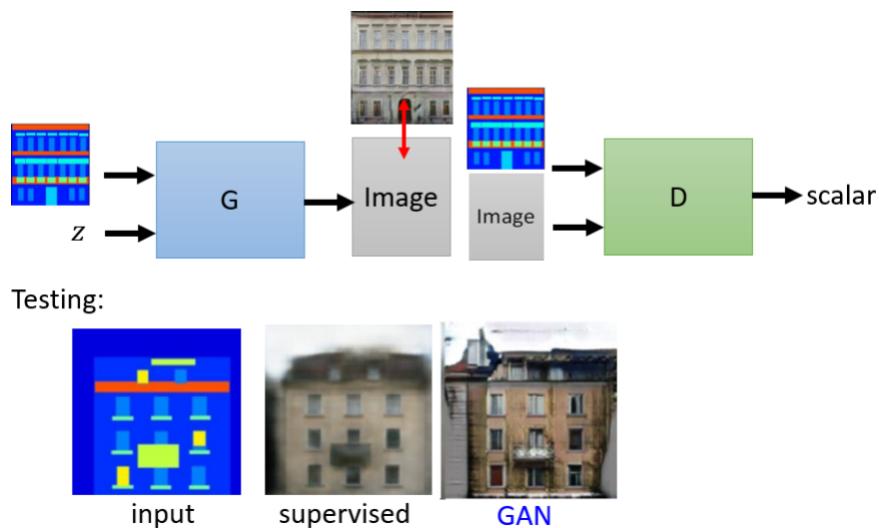
在文献上你会发现说,如果你用 Supervised Learning 的方法,你得不到非常好的结果,通常你用 Supervised Learning 的方法,训练一个图片生图片的 Generator,你產生出来的结果可能是这个样子



就是这是你的 Generator 的输入,那这个是你 Generator 的输出,那你会发现在说它非常地模糊,為什麼它非常地模糊,你可以直觉想成说,因為同样的输入,可能对应到不一样的输出,就好像我们在讲 GAN 刚开始的,开场的时候讲的那个例子,今天在同一个转角,那个小精灵可能左转,也可能右转,最后学到的,就是同时左转跟右转

那對於 Image To Image 的 Case,也是一样的,输入一张图片,输出有不同的可能,机器学到的,Generator 学到的,就是把不同的可能平均起来,结果变成一个模糊的结果

所以这个时候我们需要用 GAN 来 Train,你需要加一个 Discriminator,Discriminator 它是输入一张图片,还有输入 Condition,然后它会同时看这个图片跟这个 Condition,有没有匹配,来决定它的输出,那这个是文献上用 GAN 的输出,从右上角这篇 Paper 截取出来的



那你会发现说,如果单纯用 GAN 的话,它有一个小问题,所以它產生出来的图片,比较真实,但是它的问题是它的创造力,想像力过度丰富,它会产生一些输入没有的东西,没有叫它输入的东西,举例来说,这是一个房子,左上角明明没有其他东西,这边它却在屋顶上,加了一个不知道是烟囱还是窗户的东西

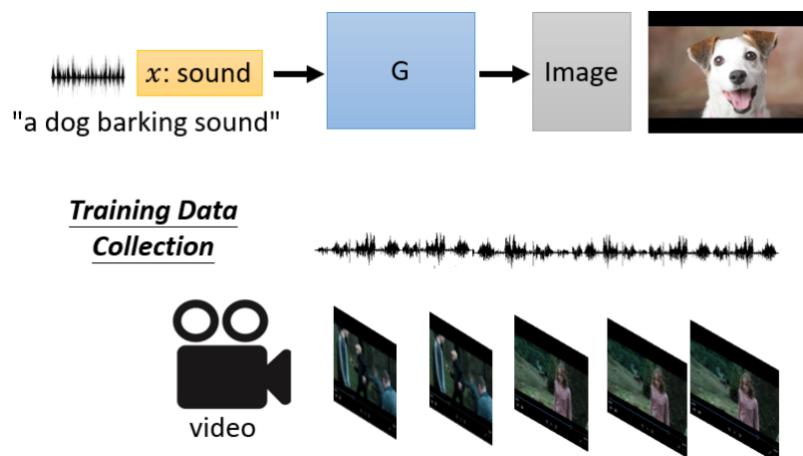
那文献上如果你要做到最好,往往就是 GAN 跟 Supervised Learning,同时使用



GAN + supervised

那同时使用,往往可以给你最好的结果,那所谓同时使用的意思就是,Generator 在训练的时候,一方面它要去骗过 Discriminator,这是它的一个目标,但同时它又想要产生一张图片,跟标准答案越像越好,它同时去做这两件事,那往往产生的结果是最好的

Conditional GAN 还有很多应用啦,这边给大家看一个莫名其妙的应用,就是给 GAN ,听一段声音,然后它产生一个对应的图片啦



比如说给它听一段狗叫声,看它能不能够画出一隻狗啦,好 那我刚才讲说 Conditional GAN 需要这个,Label 的资料,需要成对的资料

那这个声音跟影像成对的资料,其实并没有那麼难蒐集,因为你可以爬到大量的影片,那影片裡面有影像 有画面,也有声音讯号,那你就知道说,这一个画面 这一帧,这一帧的图片,这一帧的画面,对应到这一小段声音,这一帧的画面对应到这一小段声音,把这些资料蒐集起来,你就可以 Train 一个 Conditional GAN

那这个是我们实验室有个同学做的,这个是一个那个真正的 Demo

Conditional GAN

The images are generated by Chia-Hung Wan and Shun-Po Chuang.
https://wjohn1483.github.io/audio_to_scene/index.html

- Sound-to-image

Louder



那机器听这样的声音,好 这听起来有点像是这个电视机坏掉的声音,那机器觉得它听到什麼,刚才那一段声音机器觉得,它听到一个小溪,听到一个小瀑布

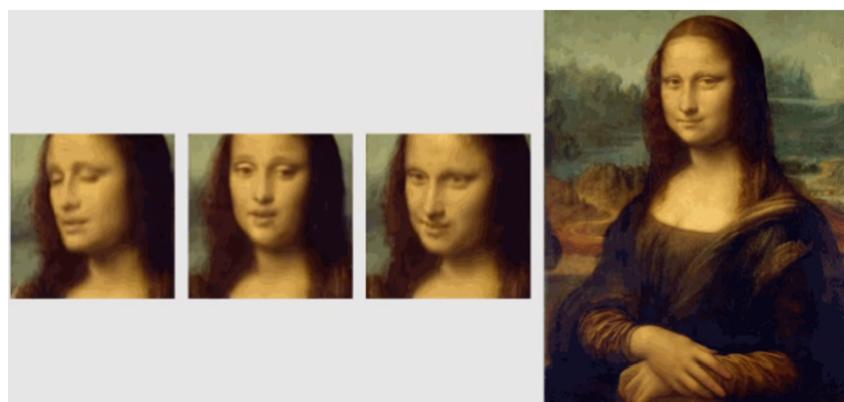
或者是我们再听另外一段声音,机器觉得它听到一艘快艇在海上奔驰

当然我有点担心说,欸 这个会不会机器并没有真的学到,声音跟图片之间的关係,会不会它只是把,它在训练资料裡面有看过的图片存起来而已,所以我决定把声音调大,你看看结果会怎样,所以我们把声音调大,接下来真的很大声哦,好 然后声音越来越大,你就发现说,这个溪流裡面的水花就越来越多,从一条小溪,变成尼加拉瓜瀑布

然后刚才的这个快艇的例子也是一样,就把快艇的声音变大,你看看会怎样,当声音越来越大的时候,你发现快艇旁边的水花就越来越多,好像快艇开得越来越快

不过我要承认,这个其实是稍微 Cherry Pick 的结果,就稍微挑过的结果,很多时候觉得 Generator 產生出来的东西,就是这个样子啦,不知所云这样,这就给它一个钢琴声,然后它好像想画一个钢琴,但又没有很清楚,这个是给它听狗叫声啦,好像想画一个动物,但又不知道要画些什麼,这个是声音到影像的產生,好 那我看到最近最惊人的,Conditional GAN 的应用,是有人用 Conditional GAN 產生会动的图片

Talking Head Generation



<https://arxiv.org/abs/1905.08233>

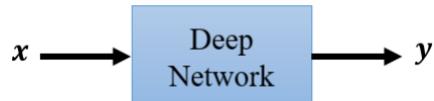
我们知道在哈利波特裡面,那些人物的画像是会动的,是会说话的,那 Samsung ,就做了一个类似的应用,用 GAN 做的,给它一张图片,比如说蒙娜丽莎的画像,然后就可以让蒙娜丽莎开始讲话,这个是 Conditional GAN 的其中一个应用,我把论文放在这边给大家参考

GAN from Unpaired Data for Unsupervised Learning

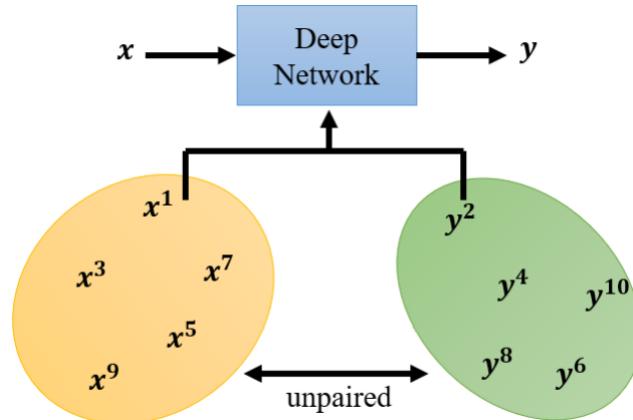
Problems

有关GAN的最后一段,是一个GAN的神奇应用,它把GAN用在unsupervised Learning上,到目前为止,我们讲的几乎都是Supervised Learning

我们要训练一个Network, Network的输入叫做X输出叫做Y,我们需要成对的资料,才有办法训练这样子的Network,



但是你可能会遇到一个状况是,我们有一堆X我们有一堆Y,但X跟Y是不成对的,在这种状况下,我们有没有办法拿这样的资料,来训练Network呢,像这一种没有成对的资料,我们就叫做unlabeled的资料



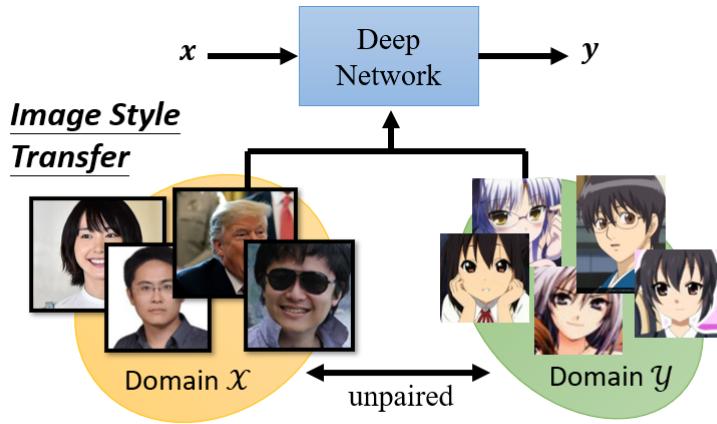
其实在作业三跟作业五裡面,都提供给你两个例子,我们把这个怎麽用,没有标註的资料,怎麽做Semi-supervised Learning,这件事情放在作业裡面,如果你有兴趣的话就可以来,体验一下semi-supervised Learning,到底可以带多大的帮助

但是不管是作业三的pseudo labeling,还是作业五的back translation,这些方法或多或少,都还是需要一些成对的资料

在作业三裡面,你得先训练出一个模型,这个模型可以帮你提供pseudo label,如果你一开始,根本就没有太多有标註的资料,你的模型很差,你根本就没有办法产生,比较好的pseudo label,或是back translation,你也得有一个,back translation 的model,你才办法做back translation,所以不管是作业三,还是作业五的方法,还是都需要一些成对的资料

但是假设我们遇到一个更艰鉅的状况,是我们一点成对的资料都没有,那要什麼怎麽办呢?

我们这边举一个例子,影像风格转换,假设今天我要训练,一个Deep Network,它要做的事情是把X domain的图,X domain的图,我们假设是真人的照片,Y domain的图是二次元人物的头像

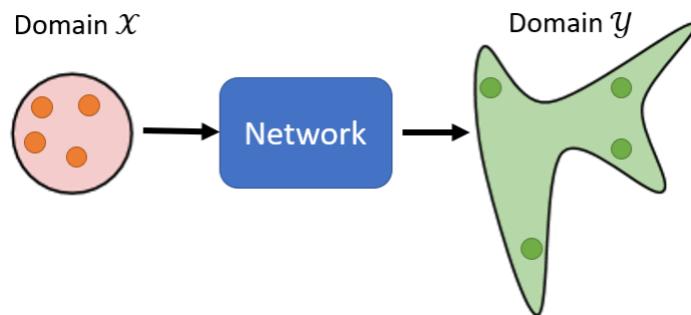


Can we learn the mapping without any paired data?
Unsupervised Conditional Generation

在这个例子裡面我们可能就没有任何的成对的资料，在这种状况下，还有没有办法训练一个Network，输入一个X產生一个Y呢，这个就是GAN可以帮我们做的事情

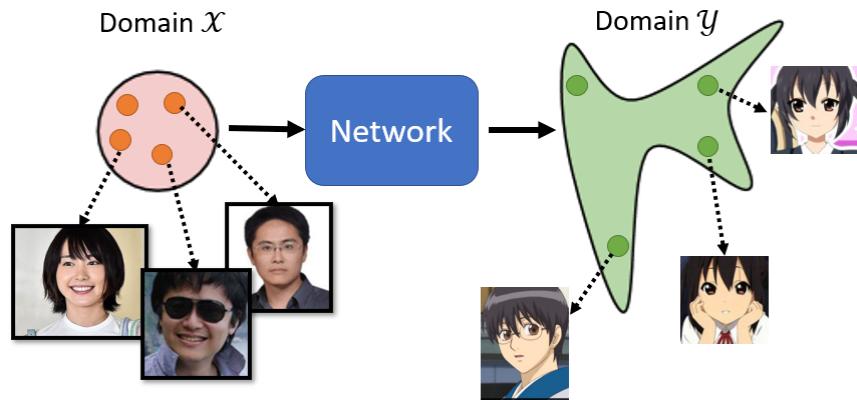
那接下来我们就是看看怎麼用GAN，在这种完全没有成对资料的情况下，进行学习

这个是我们之前在讲，unconditional的generation的时候，你看到的generator的架构



输入是一个Gaussian的分佈，输出可能是一个复杂的分佈

现在我们在稍微转换一下我们的想法，输入我们不说它是Gaussian的分佈，我们说它是X domain的图片的分佈，那输出我们说，是Y domain图片的分佈



乍听之下好像没有很难，你完全可以套用原来的GAN的想法，在原来的GAN裡面我们说，我们从Gaussian sample一个向量，丢到Generator裡面

那我们一开始也说,其实不一定只要Gaussian sample这一个distribution,只要是有办法被sample的就行了,我们选Gaussian只是因為Gaussian的formulation我们知道

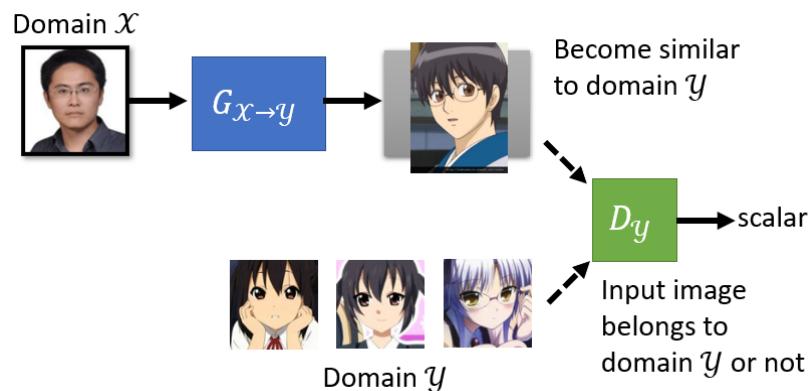
那我们现在如果,输入是X domain的distribution,我们只要改成可以,从X domain sample就结束了,那你有没有办法,从X domain sample呢



可以 你就从人脸的照片裡面,真实的人脸裡面随便挑一张出来,这是一个死臭酸宅(老师本人)然后就结束了,你就可以从X domain,sample照片出来,你把这个照片丢到generator裡面,让它產生另外一张图片,產生另外一个distribution裡面的图片

那怎麼让它变成,是Y domain的distribution呢?

那就要两三个discriminator,那这个discriminator给它,看过很多Y domain的图,所以它能够分辨Y domain的图,跟不是Y domain的图的差异

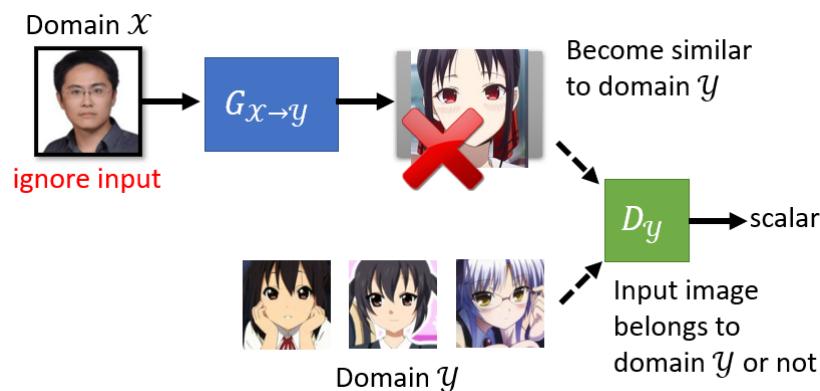


看到Y domain的图就给它高分,看到不是Y domain的图,不是二次元人物就给它低分,那就这样结束了

但是光是套用原来的GAN训练,generator跟discriminator,好像是不够的,因為我们现在的discriminator,它要做的事情是要让这个generator,输出一张Y domain的图

那generator它可能真的,可以学到输出Y domain的图,但是它输出的Y domain的图,一定要跟输入有关係吗,你没有任何的限制要求你的generator做这件事

你的generator也许就把这张图片,当作一个Gaussian的noise,然后反正它就是看到,不管你输入什麼它都无视它,反正它就输出一个,像是二次元人物的图片,discriminator觉得它做得很好,其实就结束了



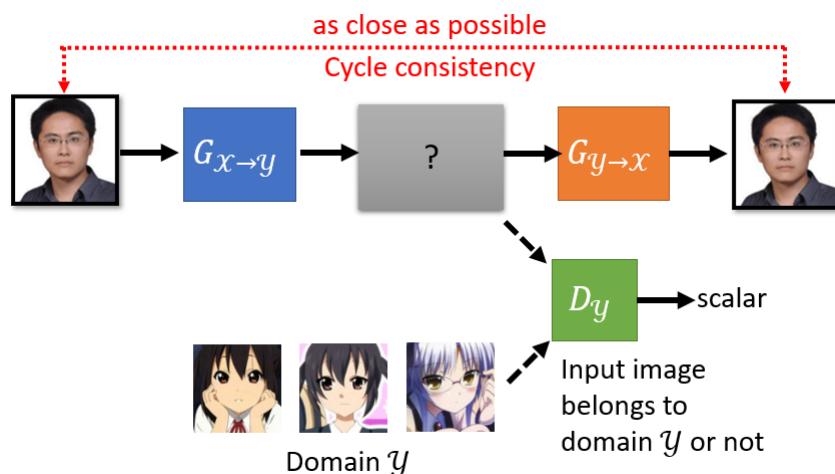
所以如果我们完全只套用,这个一般的GAN的做法,只训练一个generator,这个generator input的distribution,从Gaussian变成X domain的image,然后训练一个discriminator,显然是不够的,因為你训练出来的generator,它可以產生二次元人物的头像,但是跟输入的真实的照片,没有什麼特别的关係,那这个不是我们要的,

我们在conditional GAN的时候,是不是也看过一模一样的问题呢,在讲conditional GAN的时候,我有特别提到说,假设你的discriminator只看Y,那它可能会无视generator的输入,那產生出来的结果不是我们要的,但是这边啊,如果我们要从unpaired的数据学习,我们也没有办法,直接套用conditional GAN的想法,因為在刚才讲的,conditional GAN裡面,我们是有成对的资料,来训练的discriminator

GAN for Image Transform——Cycle GAN

这边这个想法叫做Cycle GAN,在Cycle GAN裡面,我们会train两个generator

- 第一个generator它的任务是,把X domain的图变成Y domain的图
- 第二个generator它的任务是,看到一张Y domain的图,把它还原回X domain的图



在训练的时候,我们今天增加了一个额外的目标,就是我们希望输入一张图片,从X domain转成Y domain以后,要从Y domain转回原来,一模一样的X domain的图,经过两次转换以后,输入跟输出要越接近越好

你说怎麽让两张图片越接近越好呢?

两张图片就是两个向量,这两个向量之间的距离,你就是让这两个向量,它们之间的距离越接近越好,就是要两张图片越像越好

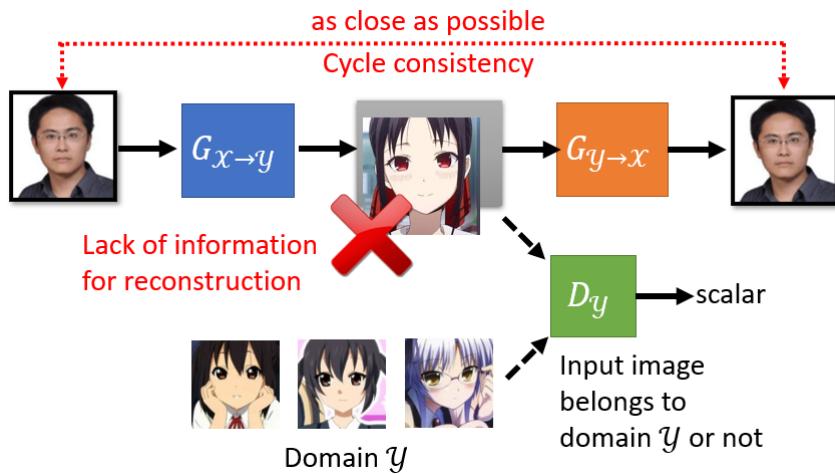
因为这边有一个循环,从X到Y 在从Y回到X,它是一个cycle,所以叫做Cycle GAN,这个要让输入经过两次转换以后,变成输出 输入跟输出越接近越好,这个叫做Cycle的consistency

所以现在这边我们有三个Network

1. 第一个generator,它的任务是把X转成Y
2. 第二个generator,它的任务是要把Y还原回原来的X
3. 那这个discriminator,它的任务仍然是要看,蓝色的这个generator它的输出,像不像是Y domain的图

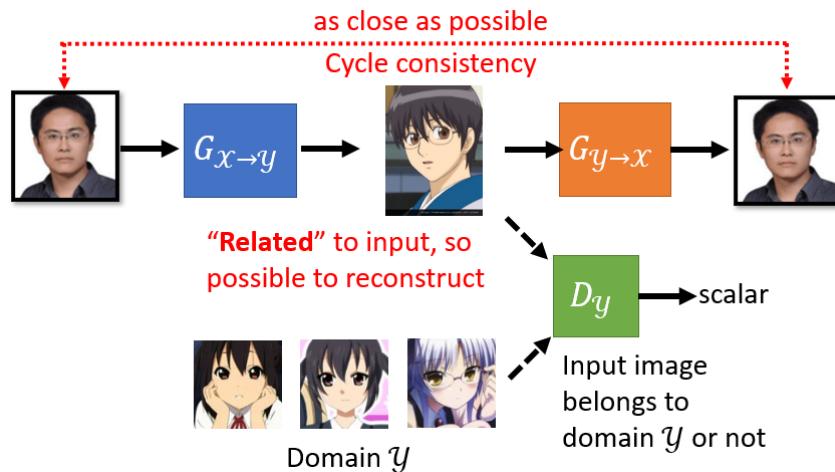
那加入了这个橙色的从Y到X的generator以后,对于前面这个蓝色的generator来说,它就再也不能够随便乱做了,它就不能够随便产生乱七八糟,跟输入没有关系的人脸了

这边假设输入一个死臭酸宅,这边假设输出的是辉夜



另外一个这个不知道这是谁的,然后对第二个generator来说,它就是视这张辉夜作为输入,它根本无法想像说,要把辉夜还原回死臭酸宅,它根本不知道说,原来输入的图片长什麼样子

所以对第一个generator来说,為了要让第二个generator能够成功的还原原来的图片,它產生出来的图片,就不能跟输入差太多,所以这边是一个死臭酸宅,这边输出至少也得是一个戴眼镜的男生的角色才行



所以这边是一个戴眼镜男生的角色,然后第二个generator才能够把这个角色还原回原来的输入,所以如果加Cycle GAN,你至少可以强迫你的generator,它输出的Y domain的图片,至少跟输入的X domain的图片,有一些关係

这时你可能会有的一个问题就是,你这边只保证有一些关係啊,你怎麼知道这个关係是我要的呢?

- 机器有没有可能学到很奇怪的转换,输入一个戴眼镜的人,然后这个generator学到的是,看到眼镜就把眼镜抹掉,然后把它变成一颗痣,然后第二个generator学到的,就是看到痣就还原回眼镜,这样还是可以满足cycle consistency,还是可以把输入的图片,变成输出的图片
- 一个更极端的例子,假设第一个generator学到的就是,把图片反转 左右翻转,第二个generator它也要学到,把图片左右翻转,你就可以还原了啊

所以今天如果我们做Cycle GAN,用cycle consistency,似乎没有办法保证,我们输入跟输出的人脸,看起来真的很像,因為也许机器会学到很奇怪的转换,反正只要第二个generator,可以转得回来就好了

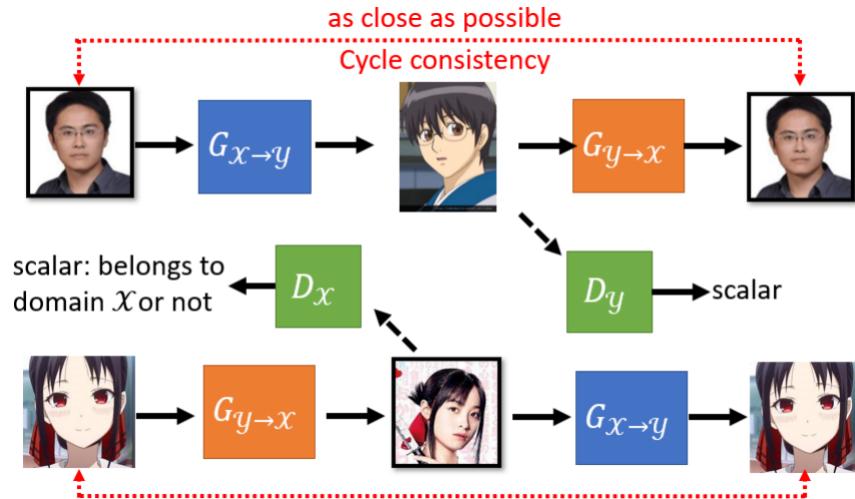
确实有可能有这样的问题发生,而且目前没有什麼特别好的解法

但我可以告诉你说,实际上你要使用Cycle GAN的时候,这样子的状况没有那麼容易出现,如果你实际上使用Cycle GAN,你会发现输入跟输出往往,真的就会看起来非常像,而且甚至在实作上,在实作的经验上,你就算没有第二个generator,你不用cycle GAN,拿一般的GAN来做,这种图片风格转换的任务,你往往也做得起来

因為在實作上你会发现, Network其实非常懶惰, 它输入一个图片, 它往往就想输出, by default就是想输出很像的东西, 它不太想把输入的图片, 做太复杂的转换, 像是什麼眼镜变成一颗痣这种状况, 它不爱這麼麻烦的东西, 有眼镜就输出眼镜, 可能对它来说是比较容易的抉择, 所以在真的实作上, 这个问题没有很大, 输入跟输出会是像, 但是理论上好像没有什麼保证说, 输入跟输出的图片一定要很像, 就算你加了cycle consistency

所以这个是实作与理论上, 你可能会遇到的差异, 总之虽然Cycle GAN没有保证说, 输入跟输出一定很像, 但实际上你会发现输入跟输出, 往往非常像, 你只是改变了风格而已

那这个Cycle GAN可以是双向的, 我们刚才有一个generator, 输入Y domain的图片, 输出X domain的图片, 我们是先把X domain的图片转成Y, 在把Y转回X

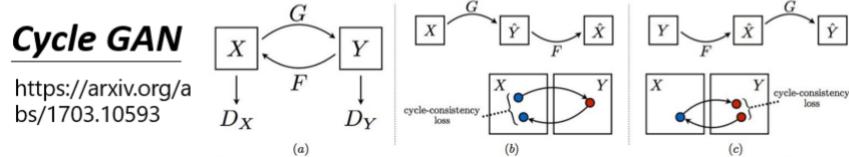
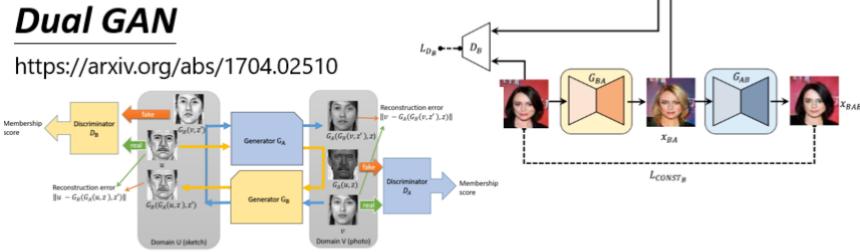
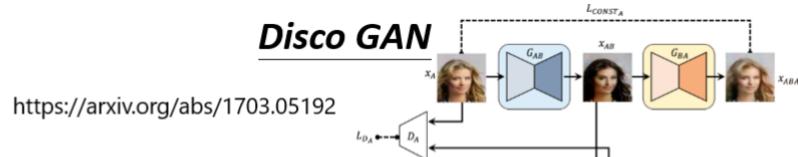


在训练cycle GAN的时候, 你可以同时做另外一个方向的训练, 也就是

- 把这个橙色的generator拿来, 给它Y domain的图片, 让它產生X domain的图片
- 然后在把蓝色的generator拿来, 把X domain的图片, 还原回原来Y domain的图片

那你依然要让, 输入跟输出越接近越好, 那你一样要训练一个discriminator, 这个discriminator是, X domain的discriminator, 它是要看一张图片, 像不像是真实人脸的discriminator, 这个discriminator要去说, 这一个橙色的generator的输出, 像不像是真实的人脸, 这个橙色的generator它要去骗过, 这个Dx这个绿色的左边, 这一个discriminator, 这个合起来就是Cycle GAN

那除了Cycle GAN以外, 你可能也听过很多其他的, 可以做风格转换的GAN



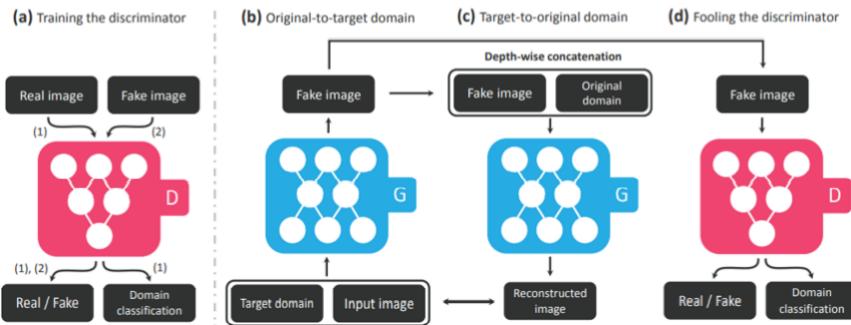
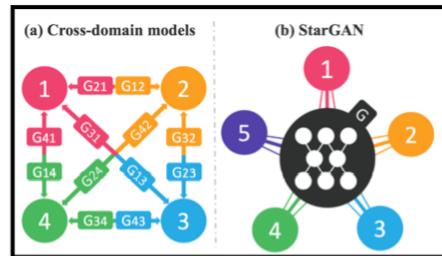
比如说Disco GAN 比如说Dual GAN,他们跟Cycle GAN有什么不同呢,就是没有半毛钱的不同这样子

你可以发现Disco GAN,Dual GAN跟Cycle GAN,其实是一样的东西,他们是一样的想法,神奇的事情是完全不同的团队,在几乎一样的时间,提出了几乎一模一样的想法,你发现这三篇文章,放到arxiv上的时间,都是17年的3月,17年的4月跟17年的3月

除了Cycle GAN以外,还有另外一个更进阶的,可以做影像风格转换的版本,叫做StarGAN

StarGAN

<https://arxiv.org/abs/1711.09020>



Cycle GAN只能在两种风格间做转换,那StarGAN 它厉害的地方是,它可以在多种风格间做转换,不过这个就不是我们接下来,想要细讲的重点

这个真实的人脸转二次元的任务,实际上能不能做呢,实际上可以做了



右上角这边放了一个连结,这个应该是一个韩国团队,他们做了一个网站,你可以上传一张图片,它可以帮你变成二次元的人物,他们实际上用的不是Cycle GAN啦,他们用的也是GAN的技术,但是是一个进阶版的东西,那我们这边就不细讲,我就把论文的连结,放在这边给大家参考

我就实际测试了一下,这个不知道大家认不认得,这是新垣结衣,这个是你老婆这样,你总该认得吧,把这个图片转成,把你老婆变成二次元的人物,长成是这个样子,你老婆二次元长这个样子知道吗

你会发现说机器确实有学到一些二次元人物的特徵,比如说 把眼睛变大,本来眼睛其实没有很大,变成二次元人物之后,眼睛变这麼大,但有时候也是会失败



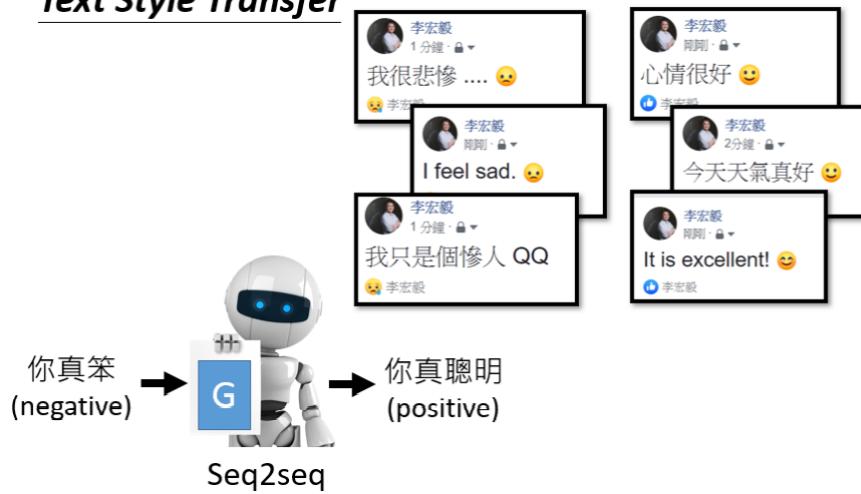
比如说 这个是美国前总统,转完以后变成这个样子,两隻眼睛一眼大一眼小就是了,它不是总是会成功的

GAN for Text Style Transfer

那同样的技术不是只能用在影像上,也可以用在文字上,你也可以做文字风格的转换

比如说,把一句负面的句子转成正面的句子,当然如果你要做一个模型,输入一个句子输出的句子,这个模型就是要能够,吃一个sequence 输出一个sequence,所以它等於是一个,sequence to sequence的model

Text Style Transfer



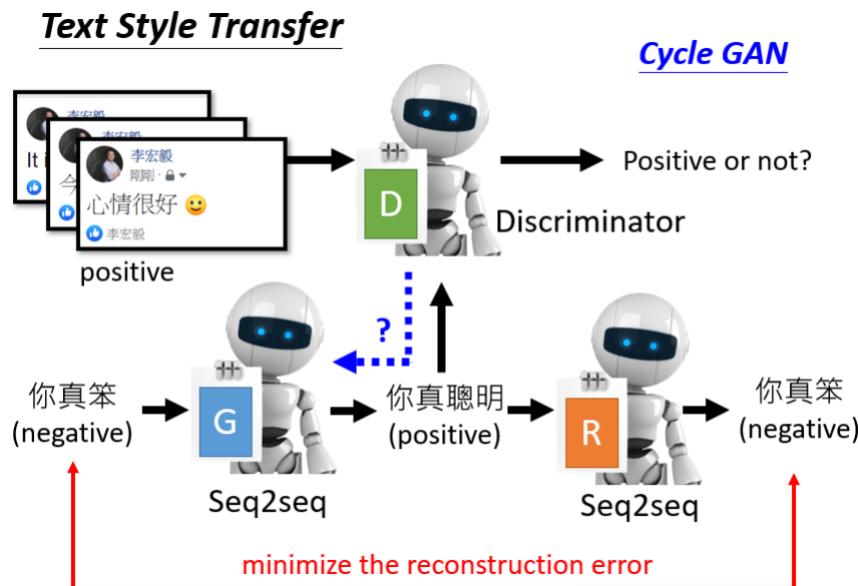
你可能就会用到,我们在作业五裡面的,Transformer的架构,来做这个文字风格转换的问题,我们在作业五做的是翻译嘛,输入一个语言输出另外一个语言嘛

现在如果要做文字风格转换,就是输入一个句子,输出另外一个风格的句子

怎麼做文字的风格转换呢,跟Cycle GAN是一模一样的,首先你要有训练资料,收集一大堆负面的句子,收集一大堆正面的句子

这个其实没有那麼难收集,你可以就是网路上爬一爬,像我们就是去PTT上爬,然后只要是推文就当作是正面的,嘘文就当作是负面的,就有一大堆正面的句子,跟负面的句子,只是成对的资料没有而已,你不知道这句推文,要怎麼转成这句嘘文,这些嘘文要怎麼转成这句推文,你没有这种资料,但是一堆推文一堆嘘文的资料,你总是可以找得到的

那接下来呢,完全套用Cycle GAN的方法,完全没有任何不同



这边就不需要再细讲 很快讲过

有一个discriminator,discriminator要看说,假设我们是要负面的句子,转正面的句子,discriminator要看说,现在generator的输出,像不像是真正的正面的句子

然后我们还要有另外一个generator,要有两个generator,这个generator要学会,把正面的句子转回原来负面的句子,你要用Cycle consistency,负面的句子转成正面的以后,还可以转回原来负面的句子

你可能会问说,这两个句子 它们两个是句子啊,怎麼算它们的相似度啊?

图片还比较好理解,图片就是个向量啊,两个向量的距离就是它们的相似度,那两个句子要怎麼做呢,这个如果你有兴趣,在留给你慢慢研究,那这边还有另外一个问题就是,这个sequence to sequence model,输出是文字,可是刚才不是有讲说,如果输出是文字,接到discriminator会有问题吗,对 会有问题 这边你就要用RL硬做

那做出来的结果怎麽样呢,这个是真正的demo, 就是真的拿PTT的推文,当正面的句子,嘘文当负面的句子,那你就可以给它一个负面的句子,它就帮你转成正面的句子,做起来像是这个样子

Text Style Transfer



- From **negative** sentence to **positive** one

胃疼, 沒睡醒, 各種不舒服 → 生日快樂, 睡醒, 超級舒服

我都想去上班了, 真夠賤的! → 我都想去睡了, 真帥的!

暈死了, 吃燒烤、竟然遇到個變態狂
→ 哈哈好~, 吃燒烤~竟然遇到帥狂

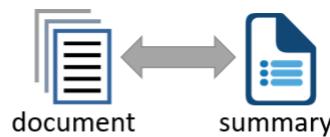
我肚子痛的厲害 → 我生日快樂厲害

,你可能问说这个系统有什麼用,就是没有任何用处 没半点用处,但是如果你觉得,你的老闆说话特別坏的话,就可以把这个系统,装在你的耳机裡面,把所有的负面的句子,转成正面的句子,你的人生可能就会,过得特别快乐一点

那其实像这一种文字风格转换,还有很多其他的应用,不是只有正面句子转负面句子

Unsupervised Abstractive Summarization

<https://arxiv.org/abs/1810.02851>



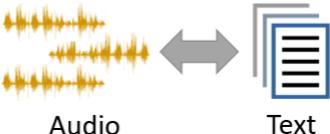
Unsupervised Translation

<https://arxiv.org/abs/1710.04087>
<https://arxiv.org/abs/1710.11041>



Unsupervised ASR

<https://arxiv.org/abs/1804.00316>
<https://arxiv.org/abs/1812.09323>
<https://arxiv.org/abs/1904.04100>



举例来说 假设我有很多长的文章,我有另外一堆摘要,这些摘要不是这些长的文章的摘要,是不同的来源,一堆长的文章 一堆摘要,让机器学习文字风格的转换,你可以让机器学会把长的文章,变成简短的摘要,让它学会怎麼精简的写作,让它学会把长的文章变成短的句子

甚至还有更狂的,同样的想法可以做,unsupervised的翻译,什麼叫做unsupervised的翻译呢,收集一堆英文的句子,收集一堆中文的句子,没有任何成对的资料,这就跟你作业五不一样,作业五你有成对的资料嘛,你知道说这句英文对到这句中文,但是unsupervised翻译就是,完全不用任何成对的资料,网路上爬一堆中文,网路上爬一堆英文,用刚才那个Cycle GAN的做法硬做,机器就可以学会把中文翻成英文了,你可以自己看一下文献,看看说机器做得怎麽样

到目前為止,我们说的两种风格都还是文字,可不可以两种风格,甚至是不同类型的资料呢,有可能做,这是我们实验室是最早做的,我们试图去做非督导式的语音辨识,,语音辨识就是,你需要收集成对的资料啊,你需要收集一大堆的声音讯号,然后找工读生,帮你把这些声音讯号标註,机器才能够学会,某个语言的语音辨识,但是要标註资料所费不貲,所以我们想要挑战,非督导式的语音辨识,也就是机器只听了一堆声音,这些声音没有对应的文字,机器上网爬一堆文字,这些文字没有对应的声音,然后用Cycle GAN硬做,看看机器有没有办法,把声音转成文字,看看它的正确率,可以做到什麼样的地步,至於正确率可以做到,什麼样的地步呢,那我把文献留在这边给大家参考,那以上就是有关GAN的部分

Concluding Remarks

Introduction of Generative Models

Generative Adversarial Network (GAN)

Theory behind GAN

Tips for GAN

Conditional Generation

Evaluation of Generative Models

Learning from unpaired data