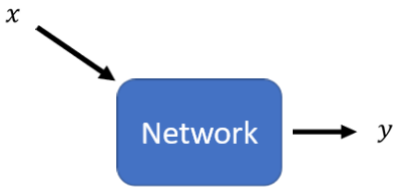
GAN_P1

Generation

Network as 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

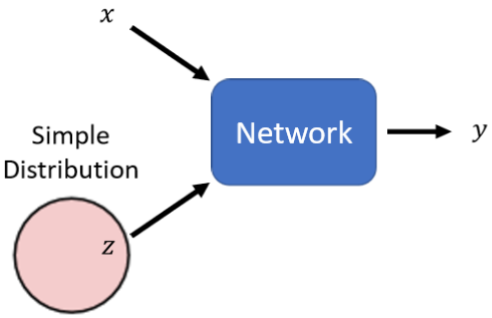
- 如果是一张图片的时候怎麼办
- 如果是一个sequence的时候怎麼办

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**



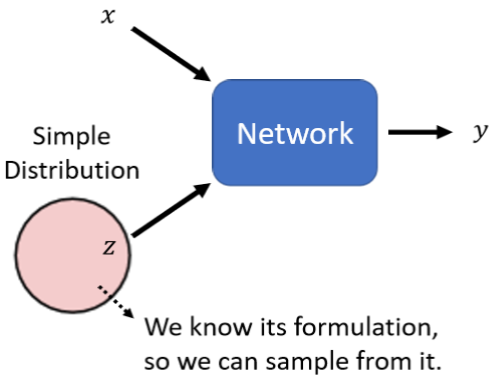
这个Z,是从某一个,distribution sample出来的,所以现在network它**不是只看一个固定的X**得到输出,它是**同时看X跟Z**得到输出

network怎麼同时看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

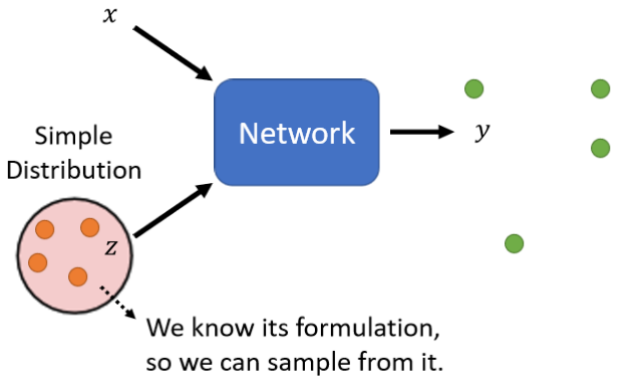
Z特别的地方是 **它是不固定的**,每一次我们用这个network的时候,都会随机生成一个Z,所以Z每次都不一样,它是从一个distribution裡面,sample出来的

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

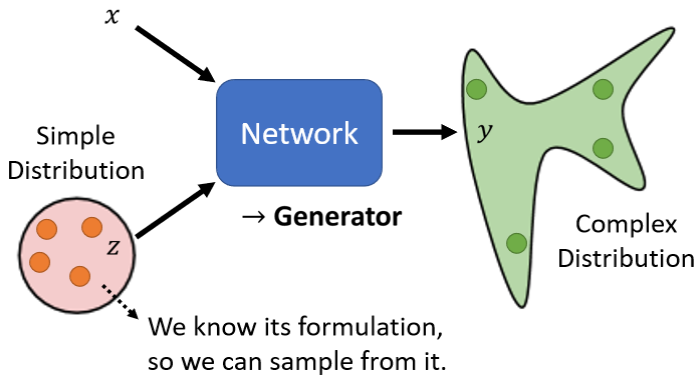


举例来说这个distribution,可以是一个function distribution,你知道function distribution的式子,你知道怎麼从,gaussian distribution做sample

它也可以是uniform distribution,那uniform distribution,的式子你一定知道,你也知道怎麼从,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,不再是单一一个固定的东西,而变成了一个复杂的distribution**,同样的X作為输入,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



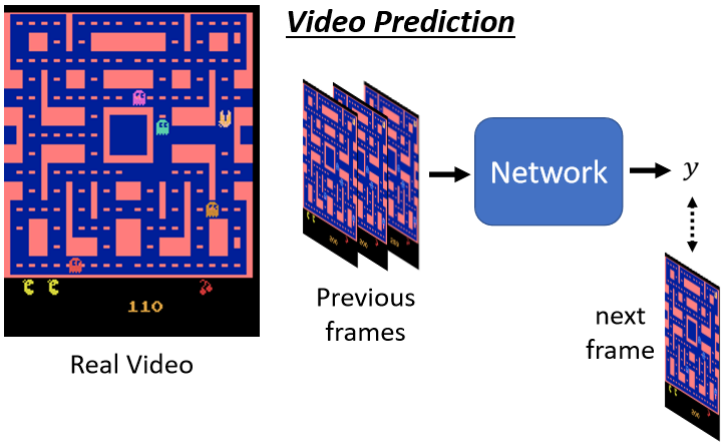
那这种**可以输出,一个distribution的network,我们就叫它generator**

Why distribution

在讲怎麼训练出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

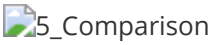
video prediction,那你就**给你的network过去的游戏画面,然后它的输出就是新的游戏画面,下一秒的下一个时间点的游戏画面**

有人可能会问说怎麼输出一张图片?

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)


5_Comparison

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**

因為今天对这个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**

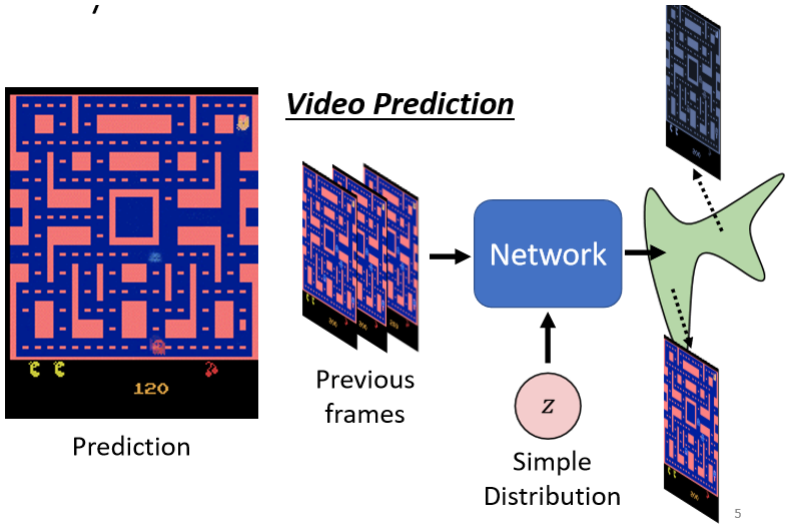
你在训练的时候,今天你的network,得到的训练的指示是,给定这一段输入,那今天得到这一笔训练资料,那它就要学到给定这段输入,输出应该要往右转,给定这一些训练资料,有时候你会看到的是向左转,那机器就会学到给定这一段输入,它要向左转

所以你的network,学到的就是**两面讨好**,

因為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,那怎麼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

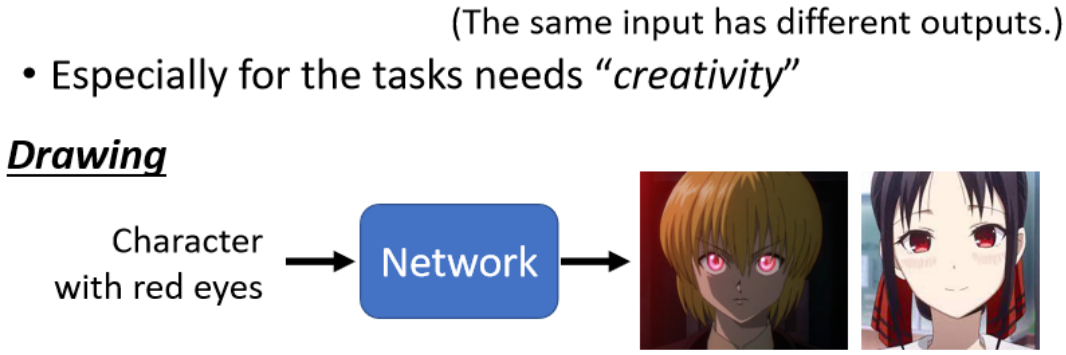
我们希望训练一个network,它可以知道说它的输出的这个分佈,包含了向左转的可能,也包含向右转的可能

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

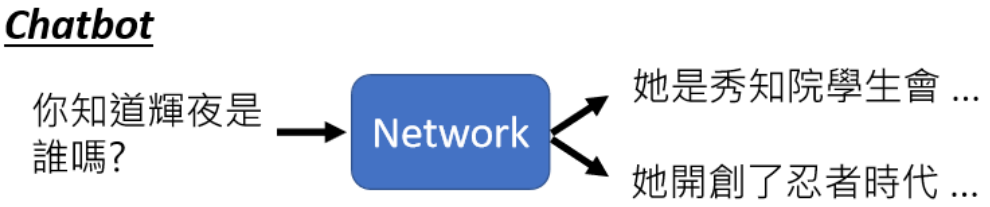


举例来说假设叫一个人,画一个红眼睛的角色,那每个人可能画出来,或者心裡想的动画人物可能都不一样,有哪些角色是红眼睛的

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,那因為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



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

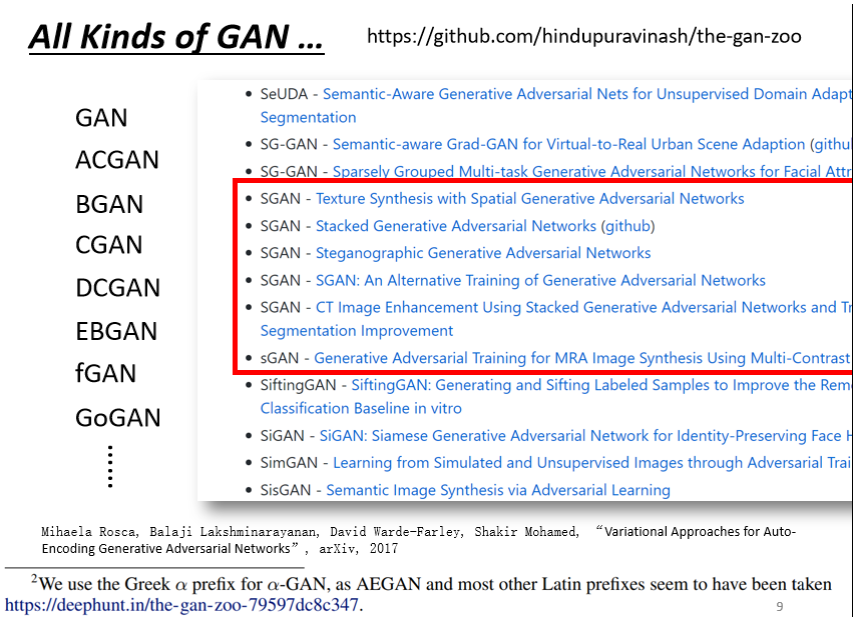
辉夜她是秀知院的学生会部会长,最近跟会长做了一些不可描述的事情,我没有说是什麼事情,所以也不算是爆雷,但是我们知道说辉夜,其实还有另外一个成就,其实虽然说辉夜姬,这个动画还没有完结,但是他的后传其实已经演完了,后来白银英年早逝所以辉夜,就把自己的头髮染白,她為了想要离开伤心的地球,就坐著太空船到另外一个星球,另外一个星球也有一些原始的生命,那些生命跟人类长得也是挺像的,他们还过著农耕的生活,然后辉夜看到其中一个小国的国王,长得跟白银有点像,所以她就跟那个小国国王在一起,她们生下了一个小孩就是六道仙人,於是就开创了忍者时代,真是可喜可贺 可喜可贺。所以我们今天学到什麼事,我们今天就是学到说,这个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

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

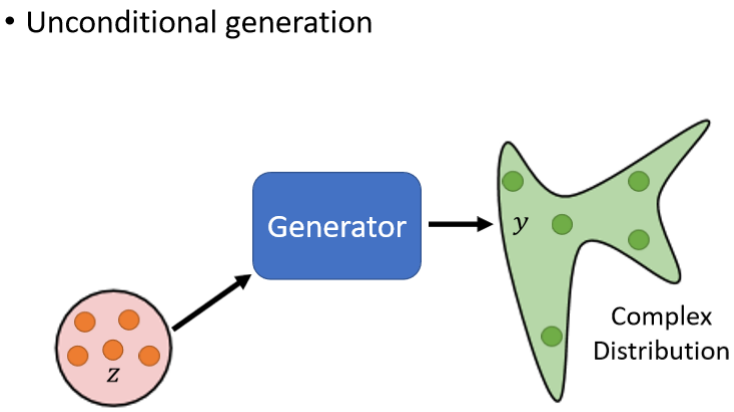


那个GAN的动物园裡面,收集了超过五百种以上的GAN,每次有人发明了,一个新的GAN的时候,他就会在前面加一个英文的字母,但是英文的字母是有限的,很快的英文的字母就被用尽了

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

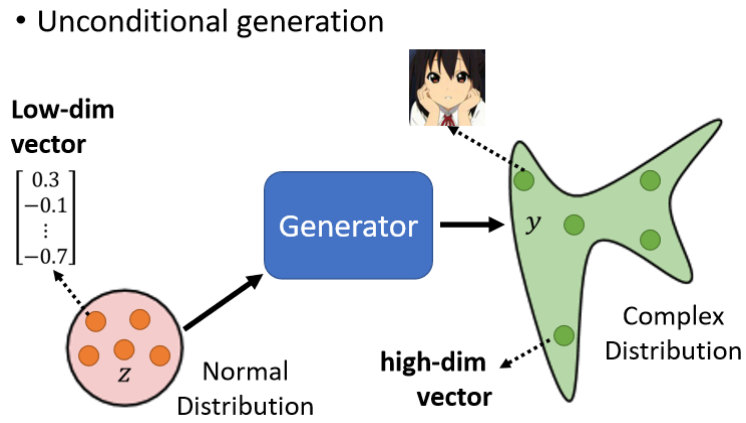
Anime Face Generation

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



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



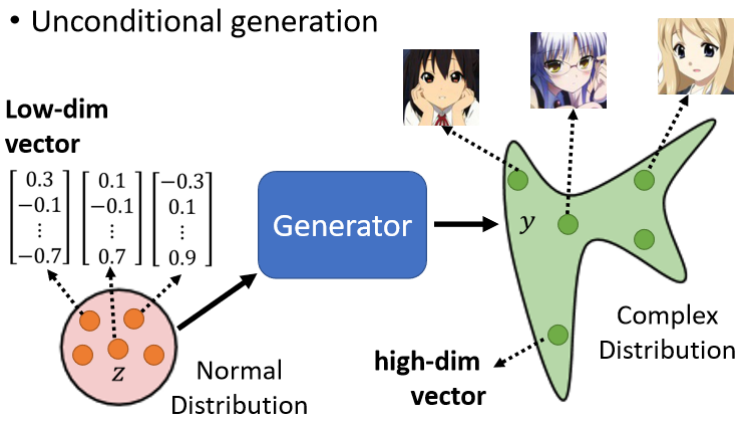
我们都假设**Z是从一个normal distribution里sample出来的向量**,那这个向量通常会是一个,**low-dimensional**的向量,它的维度其实是你自订的,你自己决定的,那通常你就订个50100,的大小,它是你自己决定的

好你从这边Z,你从这个normal distribution,裡面sample一个向量以后,丟到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,怎麼样的东西,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,这边为什麼是,normal distribution,不能是别的吗?

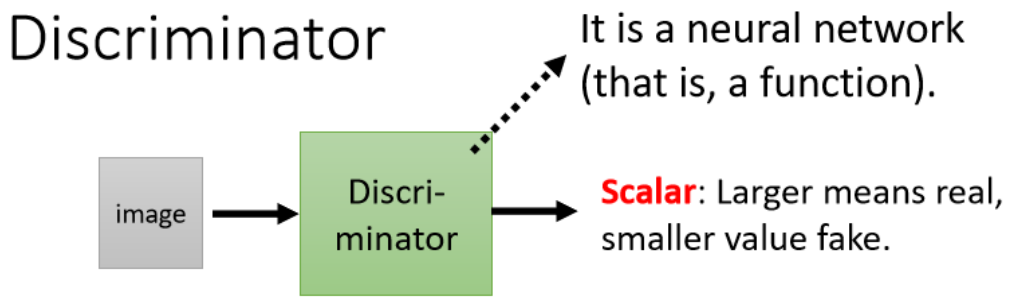
可以是别的,这边选别的你其实也会问同样的问题,就是了,那我(李宏毅本人)的经验是**不同的distribution之间的差异,可能并没有真的非常大**,不过你还是可以找到一些文献,试著去探讨不同的distribution之间,有没有差异

但是这边其实你只要选一个,够简单的distribution就行,因为你的**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

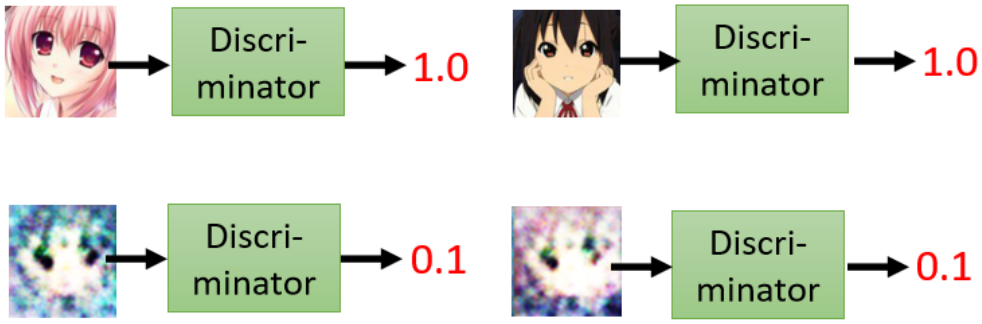
在GAN裡面,一个特别的地方就是,除了generator以外,我们要多训练一个东西,叫做discriminator

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

至於discriminator的,neural network的架构啊,这也完全是你自己设计的,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**

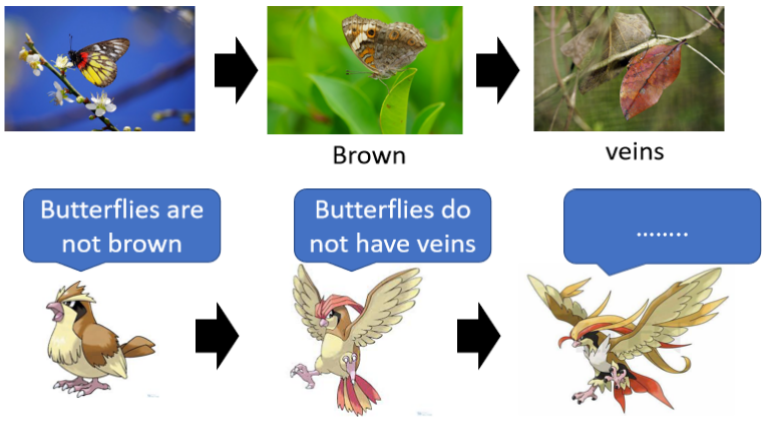
Basic Idea of GAN

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斕,但為什麼他们变得长得像枯叶一样,是因為有天择的压力

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

因為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,顏色就变成是棕色的

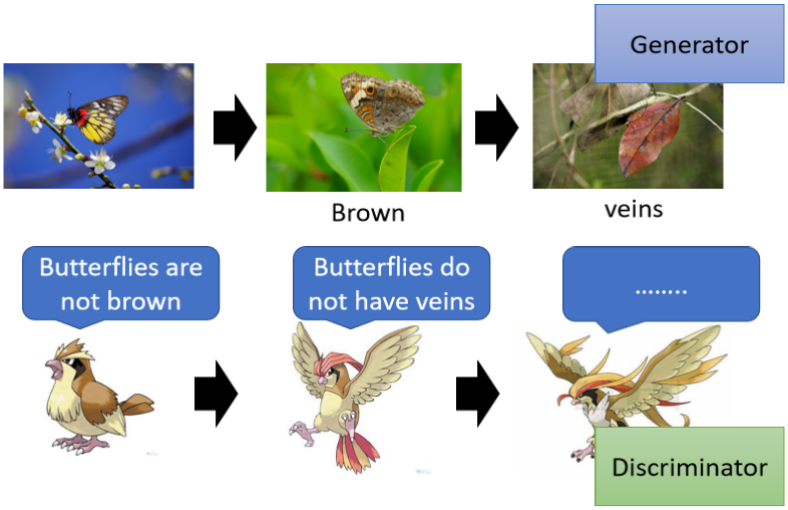


但是波波也是会演化的,所以波波為了要吃到这些枯叶蝶,你有偽装成枯叶的枯叶蝶,所以它也进化了,波波进化以后就是比比鸟这样

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,它不会只看颜色 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

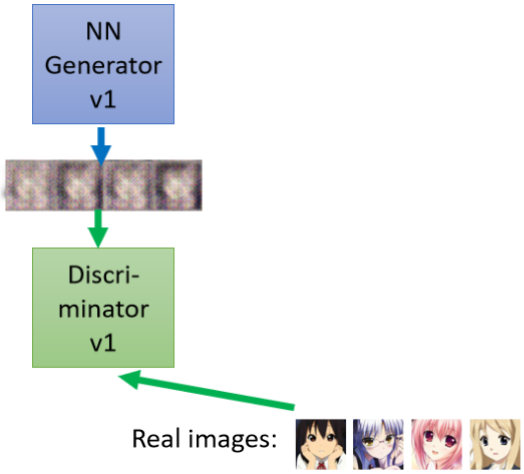
在天择的压力之下,枯叶蝶就產生了拟态 產生了叶脉,想要骗过比比鸟,但是比比鸟它也有可能会再进化,比比鸟进化是什麼,比比鸟进化就是大比鸟

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,

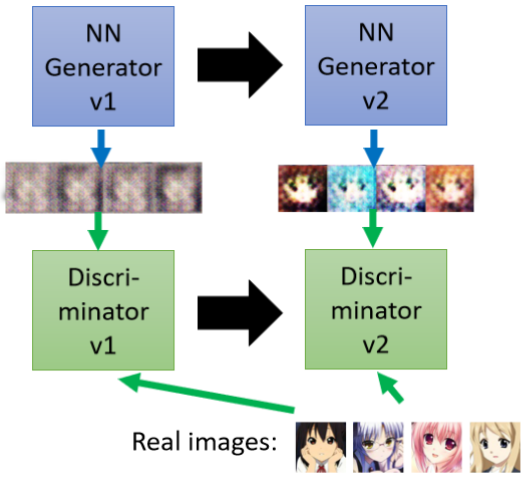
那现在我们generator要做的事情,是画出二次元的人物,那generator怎麼学习画出二次元的人物,它学习的过程是这样子



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

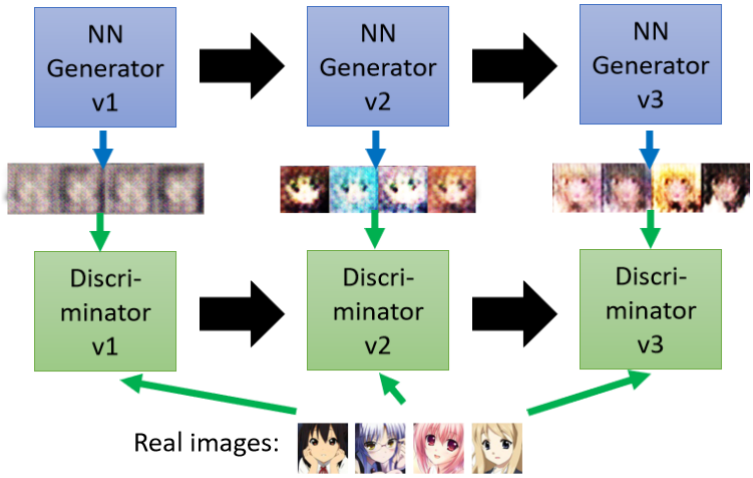
接下来**generator就调整它的裡面的参数**,Generator就**进化**了,它调整它裡面的参数 它调整的目标,是为了要骗过discriminator,假设discriminator,判断一张图片是不是真实的依据,看的是有没有眼睛,那generator就產生眼睛出来,给discriminator看



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,那因為这边有一个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arvix,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

因為觉得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

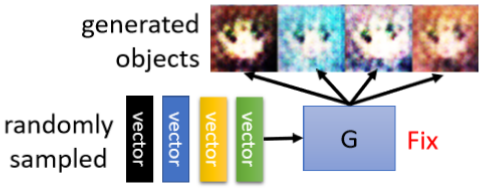
network在训练前,你要先初始化它的参数,所以我们这边就假设说,**generator跟discriminator,它们的参数 都已经被初始化了**

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



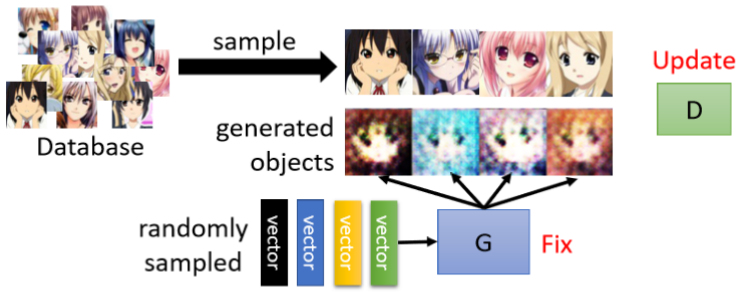
那因為一开始你的**generator的参数,是随机初始化的**,那如果你又**固定住你的generator**,那它根本就什麼事都没有做啊,它的参数都是随机的啊

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

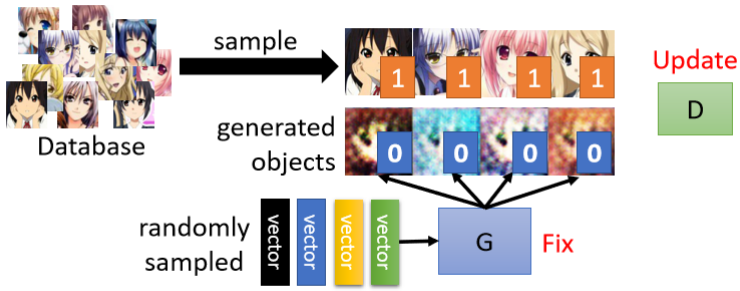


Discriminator learns to assign high scores to real objects and low scores to generated objects.

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

但是实际上怎麼做,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**,怎麼训练generator呢

拟人化的讲法是,我们就**让generator想办法去骗过discriminator**,因為刚才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它可以產生一些图片,**Discriminator觉得,是真正的图片的话,那generator產生出来的图片,可能就可以以假乱真**

Generator learns to “fool” the discriminator

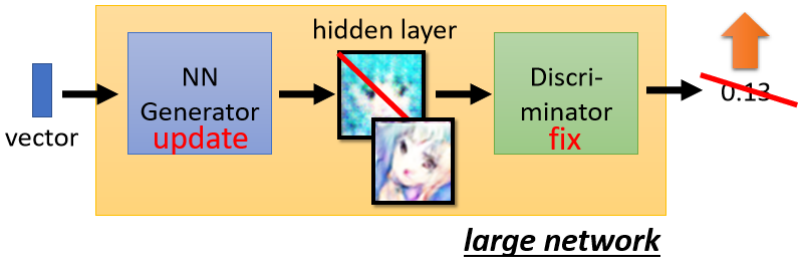
它实际上的操作方法是这样子,你有一个generator,generator吃一个向量作为输入,从gaussian distribution sample,出来的向量作为输入,然后產生一个图片

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,把它们接起来我们就把它当作是一个,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**向量作为输入**,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,因為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

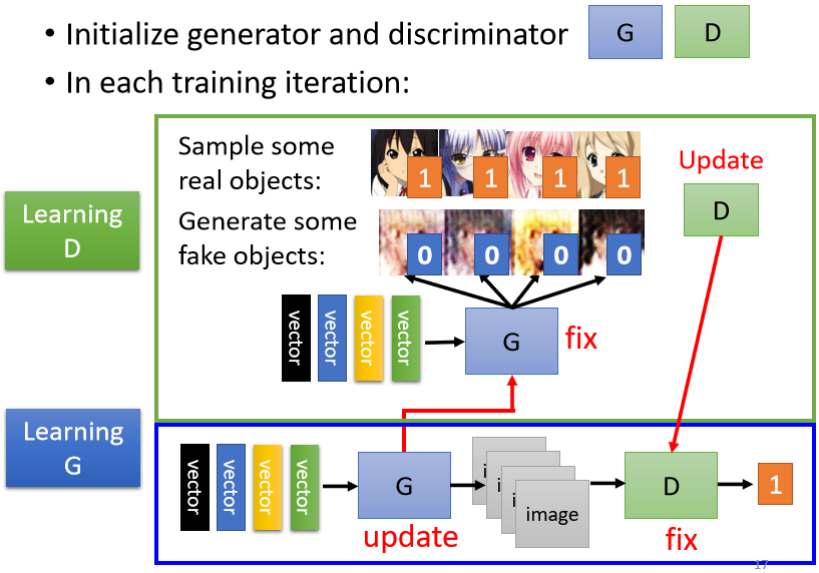
或者你也可以说,我们就是要让Discriminator output,的值越大越好,然后我们用gradient ascent,不是gradient descent,gradient descent是,让loss越小越好,gradient ascent会让你的目标函数,越大越好,我们会用gradient ascent去调generator,让Discriminator的输出越大越好

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去產生更多的,新的產生出来的图片,再给discriminator做训练,训练完discriminator以后,再去训练generator,反覆的去执行



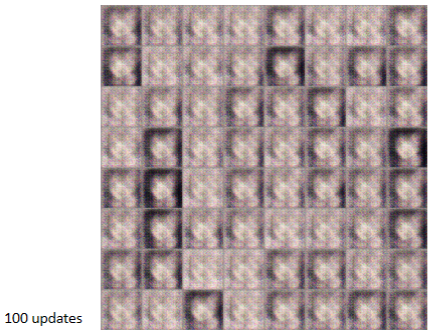
所以你是训练一阵子discriminator,训练一阵子generator,训练一阵子discriminator,再训练一阵子generator,Generator跟discriminator,它们是反覆的去进行训练,当其中一种进行训练的时候,另外一个就固定住,那你期待discriminator跟generator,都可以做得越来越好,

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是不是真的可以產生,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



Source of training data: https://zhuanlan.zhihu.com/p/24767059

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 

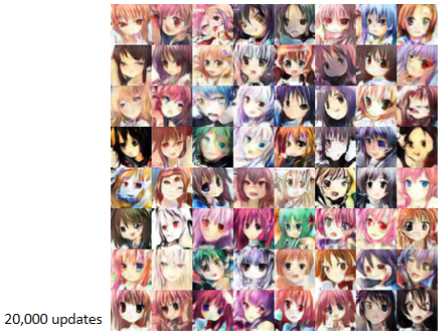
训练到五千次的时候,已经开始有一点人脸的样子了,而且你发现说机器学到说,动画人物啊,就是要有那个水汪汪的大眼睛,所以他每个人的眼睛呢,都涂得非常的大,涂有反白 代表说反光,是水汪汪的大眼睛

5000 updates 

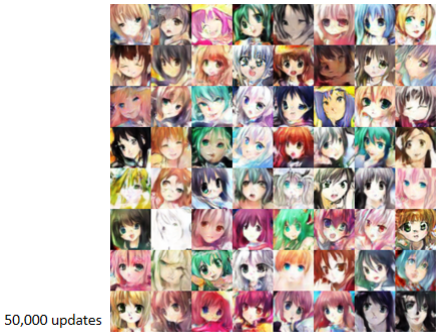
这个是训练一万次以后的结果,有发现形状已经有出来了,只是有点模糊,很多地方有点晕开的感觉,好像是水彩画的样子,

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

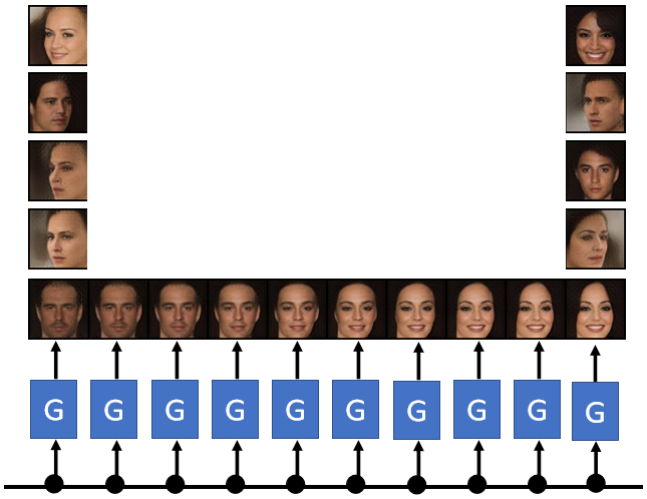
上下两排都是机器產生的,好所以这个显然progressive GAN,它有办法產生以假乱真的人脸

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,

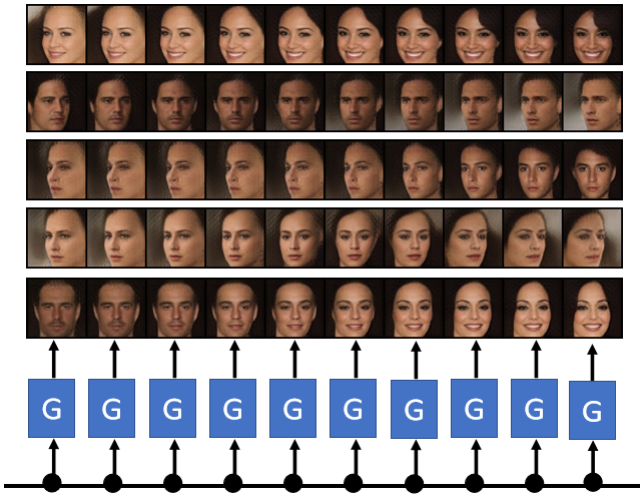
那你可能会问说要產生人脸,有什麼用呢 我去路边拍一个人,產生出来的照片不是更像真的吗

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,