7月24日

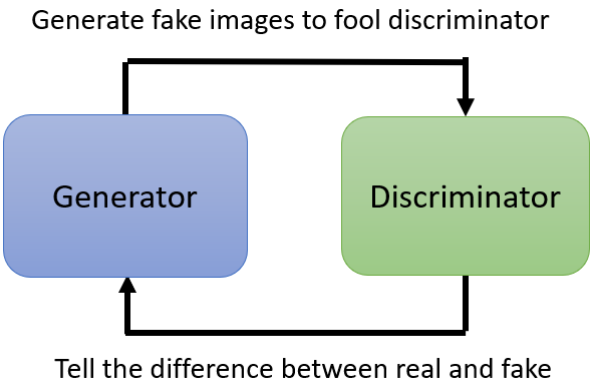
一、李宏毅2021春机器学习课程第6.2节：生成式对抗网络GAN（二）

1 No pain，no GAN

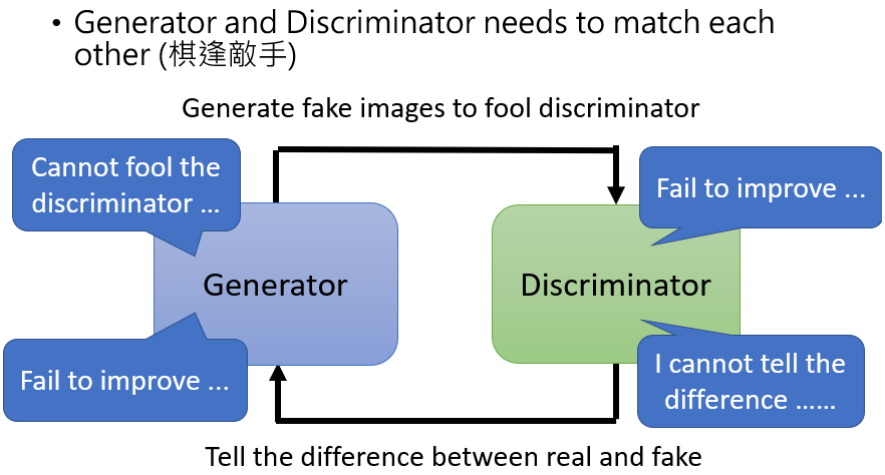
GAN是以很难 Train 起来而闻名的，那为什么 GAN 很难被 Train 起来?

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出来的图片之间的差异。
- Generator 做的事情是要产生假的图片，骗过 Discriminator。



而事实上这两个 Network，Generator 和 Discriminator，它们是**互相砥砺，才能互相成长的**，只要**其中一者发生什么问题而停止训练，另外一个就会跟着停下训练。**



到目前为止，大家已经 Train 过很多次的 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 的时候都更换 Hyperparameter，所以只能祈祷每次 Train Discriminator 的时候，它的 Loss 都是有下降的。

所以今天 Generator 跟 Discriminator 在 Train 的时候，**它们必须要棋逢敌手，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。**

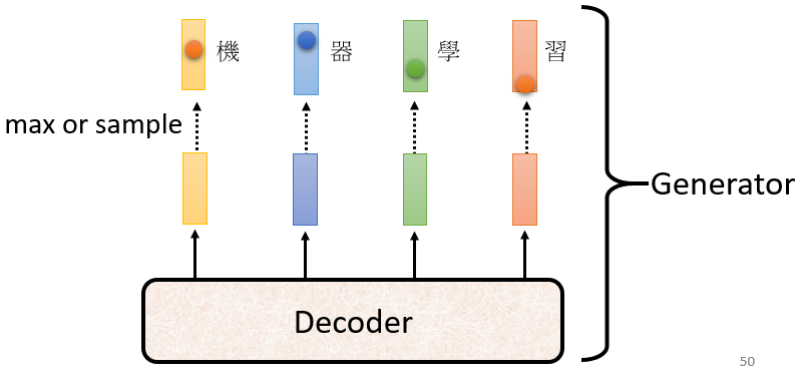
因此 GAN 本质上它的 Training 仍然不是一件容易的事情，当然**它是一个非常重要的技术**，**也是一个前瞻的技术**，这里提供一些跟 Train GAN 的诀窍有关的文献，给大家自己参考：

- Tips from Soumith
- Tips in DCGAN: Guideline for network architecture design for image generation
- Improved techniques for training GANs
- Tips from BigGAN

---

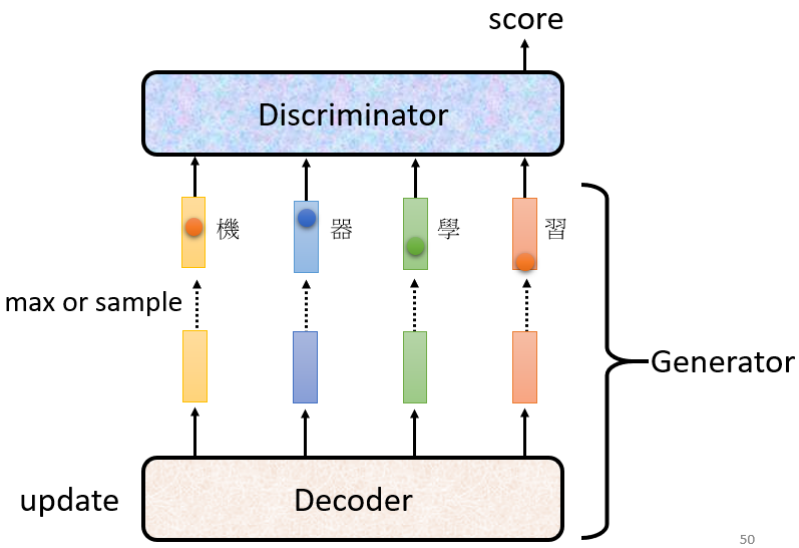
2 用 GAN 来生成一个 Sequence

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：



那这个 Decoder 会产生一段文字，所以**现在这个 Decoder 就是我们的 Generator**，这个在过去讲 Transformer 的时候，这是一个 Decoder，那它现在在 GAN 里面，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。

那我们会问这个会跟原来的在影像上的 GAN 有什么不同？**从最上层的角度来看，可能没有太大的不同**，因为接下来要做的也就是训练一个 Discriminator：



Discriminator 把 Decoder 产生的这段文字读进去，判断这段文字是真正的文字还是机器产生出来的文字，而 Decoder 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产生出来的东西是真实的。

但是真正的的难点在于，如果要用 **Gradient Descent 去 Train 你的 Decoder**，你会发现你**根本就做不到。**

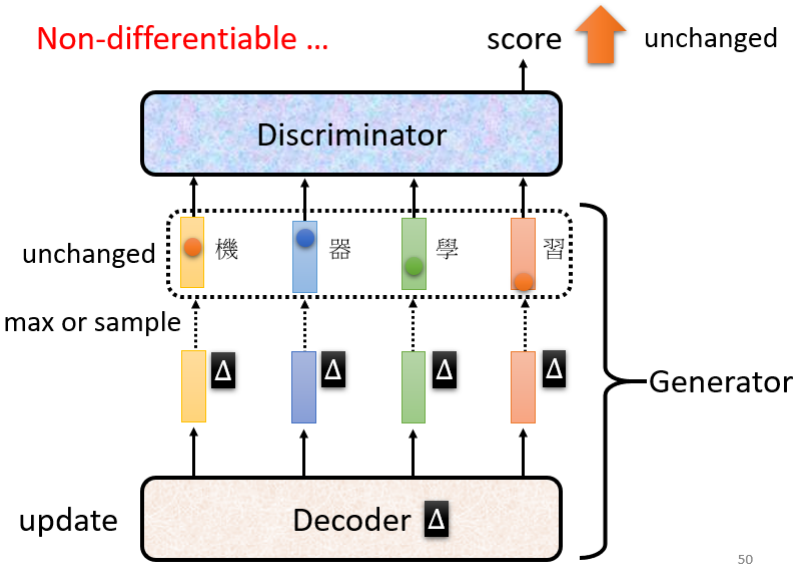
大家知道,在用Gradient Descent方法中计算微分的时候,所谓的 Gradient,所谓的**微分**,其实就是**某一个参数**,它有**变化的时候**,**对你的目标造成了多大的影响**

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 的输出有怎样的影响。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**，那因为这个变化很小，所以它**不会影响概率最大**的那一个 **Token。**

Token这个词可能会觉得有点抽象，更具体一点，Token 就是你现在在处理这个问题，处理产生这个 Sequence 的单位：

- 假设我们产生一个中文的句子的时候，我们是每次产生一个方块字，那方块字就是我们的 Token。
- 那假设你在处理英文的时候，你每次产生一个英文的字母，那字母就是你的 Token。
- 假设你一次是产生一个英文的词，英文的词和词之间是以空白分开的，那词就是你的 Token。

总之 Token 的定义是人为决定的，也是基于对问题的具体理解。

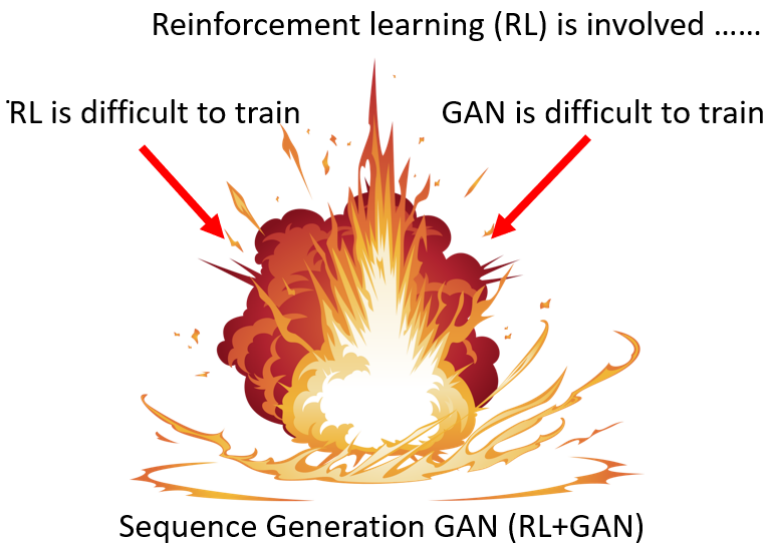


回到刚刚说的上来，因为这个变化很小，所以它不会影响概率最大的那一个 Token，那对 Discriminator 来说，它**输出的分数是一模一样的**，这样输出的分数就没有改变。所以你根本就没有办法算微分，你根本就**没有办法做 Gradient Descent**。

但遇到不能用 Gradient Descent Train 的问题，就当做 Reinforcement Learning 的问题，硬做一下就结束了。所以实际上确实可以用 Reinforcement Learning 来 Train 你的 Generator。

> 强化学习是智能体（Agent）以"试错"的方式进行学习，通过与环境进行交互获得的奖赏指导行为，目标是使智能体获得最大的奖赏，强化学习不同于连接主义学习中的监督学习，主要表现在强化信号上，强化学习中由环境提供的强化信号是对产生动作的好坏作一种评价(通常为标量信号)，而不是告诉强化学习系统RLS(reinforcement learning system)如何去产生正确的动作。由于外部环境提供的信息很少，RLS必须靠自身的经历进行学习。通过这种方式，RLS在行动-评价的环境中获得知识，改进行动方案以适应环境。

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效果可想而知，肯定非常非常地难训练。

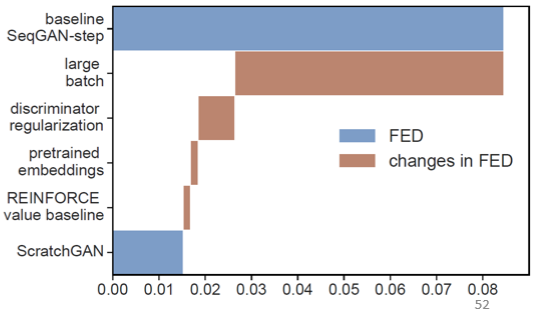


所以要用 GAN 产生一段文字，过去一直被认为是一个非常大的难题，所以有很长一段时间，没有人可以成功地把 Generator 训练起来产生文字，通常你需要先做 Pretrain，直到有一篇 Paper 叫做 ScrachGAN 出现了：

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

Training language GANs from Scratch

https://arxiv.org/abs/1905.09922



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**关键的就是爆调 Hyperparameter，还使用了一大堆的 Tips。**

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

有关 GAN 的部分这里也只是讲了一个大概，更完整的内容呢可以参考下面的链接：

- This lecture: Generative Adversarial Network (GAN)

Full version

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

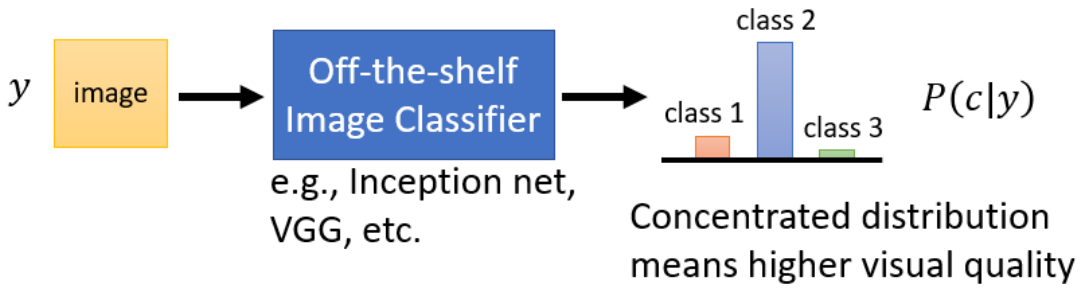
3 怎么评估一个 Generator 的好坏

要**评估一个 Generator 的好坏其实并没有想象中那么容易**，那最直觉的做法也许是**找人来看**，直接让真人来看这个 Generator 产生出来的图片到底像不像动画的人物就结束了。

其实很长一段时间，尤其是人们刚开始研究 Generative 这样技术的时候，很长一段时间没有好的评判标准，那时候要评估 Generator 的好坏都是人眼看。

但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，所以我们需要一个比较客观，并且全自动的方法，来评判一个 Generator 的好坏。

那有一个方法，是使用一个图片的分类系统，**把你的 GAN 产生出来的图片丢到一个图片的分类系统里面**，看它产生什么样的结果：



图片分类系统输入是一张图片，输出是一个机率分布，接下来我们就看，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里面有什么东西，不知道它是猫还是狗，但是**如果丢到了一个图片分类系统以后输出的分布非常集中，代表图片分类系统非常肯定它现在看到了什么。**这就代表说你产生出来的图片，也许就是比较接近真实的图片，所以图片分类系统才辨识得出来。

如果你**产生出来的图片是一个四不像**，根本看不出是什么动物，那图片分类系统就会非常困惑，它产生出来的这个**机率分布**就会非常地**平坦**，这就代表说你的 GAN 产生出来的图片可能是比较奇怪的，所以图片分类系统才会辨识不出来。

3.1 多样性问题 - 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，有时候输出的图片来来回回就那几张。

- 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** 输出的分布。

你会发现说 GAN 它**输出来的图片来来去去就那几张**，可能每一张拿出来看，觉得好像还做得不错，但让它多产生几张就会露出马脚。

那为什么会有 Mode Collapse 这种现象发生，直觉上还是比较容易理解的，可以想到这个地方就是 Discriminator 的一个盲点，当 Generator 学会产生这种图片以后，它发现这种图片总能骗过 Discriminator，于是它就一个劲的产生这种图片，就发生 Mode Collapse 的状况。

那要**怎么避免Mode Collapse** 的状况呢，今天**其实还没有一个非常好的解答**，就留给大家自己探究。

3.2 多样性问题 - Mode Dropping

其实还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，但是你产生出来的资料其实只有真实资料的一部分，单纯看产生出来的资料，你可能会觉得还不错，而且多样性也够：

但你不知道真实的资料**多样性的分布其实更大**，这边举一个真实的例子，有人 Train 了一个人脸生成的 GAN，那它在某一个 Iteration 的时候，它的 Generator 产生出下面这些人脸：

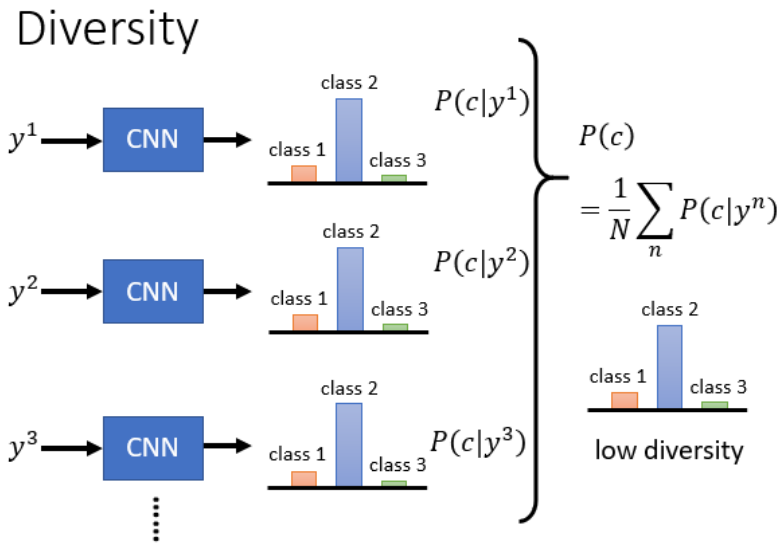
看起来好像图片的质量没有问题，而且人脸的多样性也够，有男有女，有向左看有向右看，各式各样的人脸都有，但如果你再看下一个 Iteration，Generator 产生出来的图片是这样子的：

(BEGAN on CelebA)

你会发现它只是简单调换了图片中人物的肤色，实际上还是只有这些图片而已。所以今天也许 **Mode Dropping 的问题也还没有获得本质上的解决。**

但是我们会需要一个度量标准，来评判现在我们的 Generator 产生出来的图片到底多样性够不够。

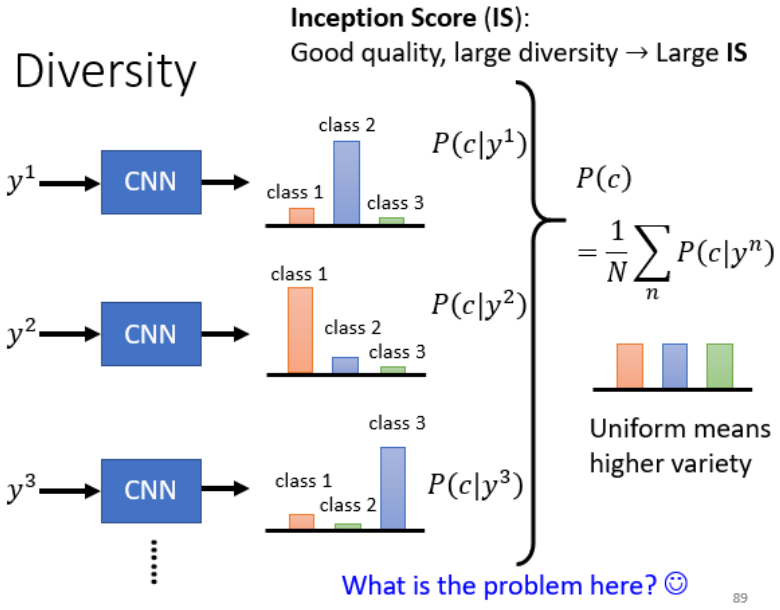
过去有一个做法，还是**借助我们的 Image Classifier**，假设你的 Generator 产生 1000 张图片，把这 1000 张图片都丢到 Image Classifier 里面，**看它们分别被判断成哪一个 Class。**



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 长什么样子。

如果平均的 **Distribution 非常集中**，就代表现在**多样性不够**，如果什么图片丢进去，你的影像分类系统都说是看到了 Class 2，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，产生出来的输出的分布都非常地不同，**平均的 Distribution 非常平坦，那这个时候代表你的多样性是很可能足够的。**



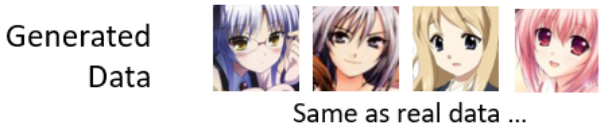
过去有一个常被使用的分数，叫做 Inception Score，缩写是 IS， Inception Score 是一个同时考虑生成图片的质量和多样性的指标。这里还有一种评分方式叫做 Fréchet Inception Distance，缩写是 FID，这里也是提供一个链接供参考，不会具体讲解。

【深度理解】如何评价GAN网络的好坏？IS（inception score）和FID（Fréchet Inception Distance）_月下花弄影-CSDN博客

3.3 我们不想要一个没有创造力的 GAN.

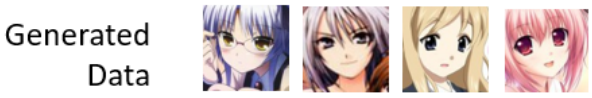
那其实刚才那些方法也并没有完全解决 GAN 的 Evaluation 的问题，现在看下面的状况，假设这是你的真实资料

你训练了一个 Generator，它产生出来的 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么样子，你光看这个 Generator 的输出，你会觉得它做得很棒，那 FID 算出来一定是非常小的。但问题是这个肯定不是我们要的，我们 **Train Generator 其实是希望它产生新的图片，产生训练资料里面没有的人脸。**

你可能会说，那我们就把 **Generator 产生出来的图片跟真实资料比个相似度**，如果相似度很高就代表 Generator 只是把那个训练资料背了下来而已。但是假设你的 Generator 学到的是把所有训练资料里面的图片都**左右反转**：



Simply flip real data ...

它实际上也是什么事都没有做，但问题是你比相似度的时候又比不出来，所以 **GAN 的 Evaluation 是非常地困难的**，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 https://arxiv.org/abs/1802.03446，里面就列举了二十几种 GAN Generator 的评估的方式：

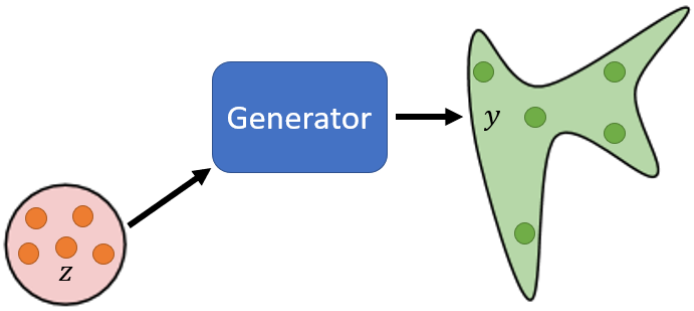
	Measure	Description
Quantitative	1. Average Log-likelihood [18, 22]	• Log likelihood of explaining realworld held out/test data using a density estimated from the generated data (*e.g.* using KDE or Parzen window estimation). $L = \frac{1}{N}\sum_i \log P_{model}(\mathbf{x}_i)$
	2. Coverage Metric [13]	• The probability mass of the true data "covered" by the model distribution $C := P_{data}(dP_{model} > t)$ with t such that $P_{model}(dP_{model} > t) = 0.95$
	3. Inception Score (IS) [3]	• KLD between conditional and marginal label distributions over generated data. $\exp(\mathbb{E}_{\mathbf{x}}[\mathrm{KL}\ p(y \mid \mathbf{x}) \parallel p(y)])$
	4. Modified Inception Score (m-IS) [34]	• Encourages diversity within images sampled from a particular category. $\exp(\mathbb{E}_{\mathbf{x}_c}[\mathrm{KL}(P(\mathbf{y}\|\mathbf{x}_c)\|P(\mathbf{y}\|\mathbf{x}_j))])$
	5. Mode Score (MS) [35]	• Similar to IS but also takes into account the prior distribution of the labels over real data. $\exp(\mathbb{E}_{\mathbf{x}}[\mathrm{KL}(p(y \mid \mathbf{x}) \parallel p(y^{train})] - \mathrm{KL}(p(y) \parallel p(y^{train}))])$
	6. AM Score [36]	• Takes into account the KLD between distributions of training labels vs. predicted labels, as well as the entropy of predictions. $\mathrm{KL}(p(y^{train}) \parallel p(y)) + \mathbb{E}_{\mathbf{x}}[H(y\|\mathbf{x})]$
	7. Fréchet Inception Distance (FID) [37]	• Wasserstein-2 distance between multi-variate Gaussians fitted to data embedded into a feature space $FID(r, g) = \|\mu_r - \mu_g\|_2^2 + Tr(\Sigma_r + \Sigma_g - 2(\Sigma_r\Sigma_g)^{\frac{1}{2}})$
	8. Maximum Mean Discrepancy (MMD) [18]	• Measures the dissimilarity between two probability distributions P_r and P_g using samples drawn independently from each distribution. $M_k(P_r, P_g) = \mathbb{E}_{\mathbf{x},\mathbf{x}'\sim P_r}[k(\mathbf{x},\mathbf{x}')] - 2\mathbb{E}_{\mathbf{x}\sim P_r,\mathbf{y}\sim P_g}[k(\mathbf{x},\mathbf{y})] + \mathbb{E}_{\mathbf{y},\mathbf{y}'\sim P_g}[k(\mathbf{y},\mathbf{y}')]$
	9. The Wasserstein Critic [39]	• The critic (*e.g.* an NN) is trained to produce high values at real samples and low values at generated samples $W(\mathbf{x}_{real}, \mathbf{x}_g) = \frac{1}{N_{batch}}\sum_{i}^{N} f(\mathbf{x}_{real}[i]) - \frac{1}{N_{batch}}\sum_{i}^{N} f(\mathbf{x}_g[i])$
	10. Birthday Paradox Test [27]	• Measures the support size of a discrete (continuous) distribution by counting the duplicates (near duplicates)
	11. Classifier Two Sample Test (C2ST) [40]	• Answers whether two samples are drawn from the same distribution (*e.g.* by training a binary classifier)
	12. Classification Performance [1, 15]	• An indirect technique for evaluating the quality of unsupervised representations (*e.g.* feature extraction; FCN score). See also the GAN Quality Index (GQI) [41].
	13. Boundary Distortion [42]	• Measures diversity of generated samples and covariate shift using classification methods.
	14. Number of Statistically-Different Bins (NDB) [43]	• Given two sets of samples from the same distribution, the number of samples that fall into a given bin should be the same up to sampling noise
	15. Image Retrieval Performance [44]	• Measures the distributions of distances to the nearest neighbors of some query images (i.e. diversity)
	16. Generative Adversarial Metric (GAM) [31]	• Compares two GANs by having them engaged in a battle against each other by swapping discriminators or generators. $p(\mathbf{x}\|y=1; M_1)/p(\mathbf{x}\|y=1; M_2) = (p(y=1)\mathbf{x}; D_1)p(\mathbf{x}; G_2))/(p(y=1)\mathbf{x}; D_2)p(\mathbf{x}; G_1))$
	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 fake data or a generator that attempts to fool the discriminators into accepting fake data as real.
	18. Normalized Relative Discriminative Score (NRDS) [12]	• Compares n GANs based on the idea that if the generated samples are closer to real ones, more epochs would be needed to distinguish them from real samples.
	19. Adversarial Accuracy and Divergence [46]	• Adversarial Accuracy: Computes the classification accuracies achieved by the two classifiers, one trained on real data and another on generated data, on a labeled validation set to approximate $P_g(y\|\mathbf{x})$ and $P_r(y\|\mathbf{x})$. Adversarial Divergence: Computes $\mathrm{KL}(P_g(y\|\mathbf{x}), P_r(y\|\mathbf{x}))$
	20. Geometry Score [47]	• Compares geometrical properties of the underlying data manifold between real and generated data.
	21. Reconstruction Error [48]	• Measures the reconstruction error (*e.g.* L_2 norm) between a test image and its closest generated image by optimizing for z (*i.e.* $\min_{\mathbf{z}} \|G(\mathbf{z}) - \mathbf{x}^{(test)}\|^2$)
	22. Image Quality Measures [49, 50, 51]	• Evaluates the quality of generated images using measures such as SSIM, PSNR, and sharpness difference
	23. Low-level Image Statistics [52, 53]	• Evaluates how similar low-level statistics of generated images are to those of natural scenes in terms of mean power spectrum, distribution of random filter responses, contrast distribution, etc.
	24. Precision, Recall and F_1 score [23]	• These measures are used to quantify the degree of overfitting in GANs, often over toy datasets.
Qualitative	1. Nearest Neighbors	• To detect overfitting, generated samples are shown next to their nearest neighbors in the training set
	2. Rapid Scene Categorization [18]	• In these experiments, participants are asked to distinguish generated samples from real images in a short presentation time (*e.g.* 100 ms); i.e. real v.s fake
	3. Preference Judgment [54, 55, 56, 57]	• Participants are asked to rank models in terms of the fidelity of their generated images (*e.g.* pairs, triples)
	4. Mode Drop and Collapse [58, 59]	• Over datasets with known modes (*e.g.* a GMM or a labeled dataset), modes are computed as by measuring the distances of generated data to mode centers
	5. Network Internals [1, 60, 61, 62, 63, 64]	• Regards exploring and illustrating the internal representation and dynamics of models (*e.g.* space continuity) as well as visualizing learned features

Pros and cons of GAN evaluation measures
https://arxiv.org/abs/1802.03446

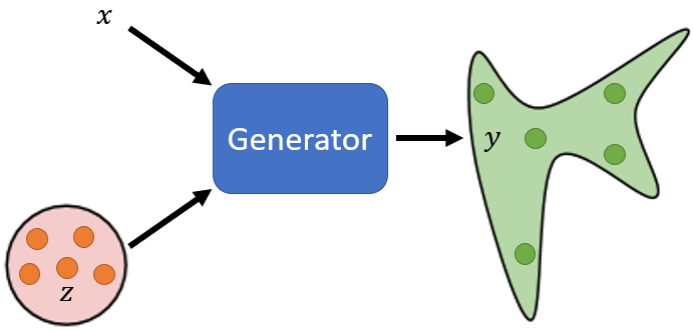
---

4 有条件的生成

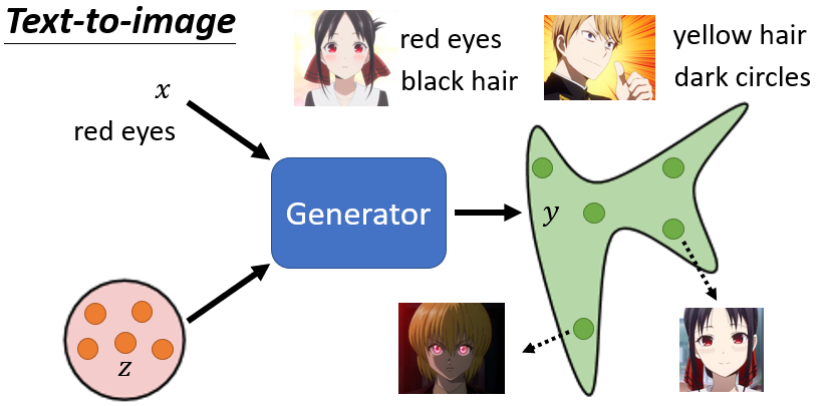
到目前为止我们讲的 Generator，它的输入都是一个随机的分布而已。



我们现在想要更进一步的是，尝试操控 Generator 的输出，我们给它一个 Condition x，让它根据 x 跟 z 来产生 y



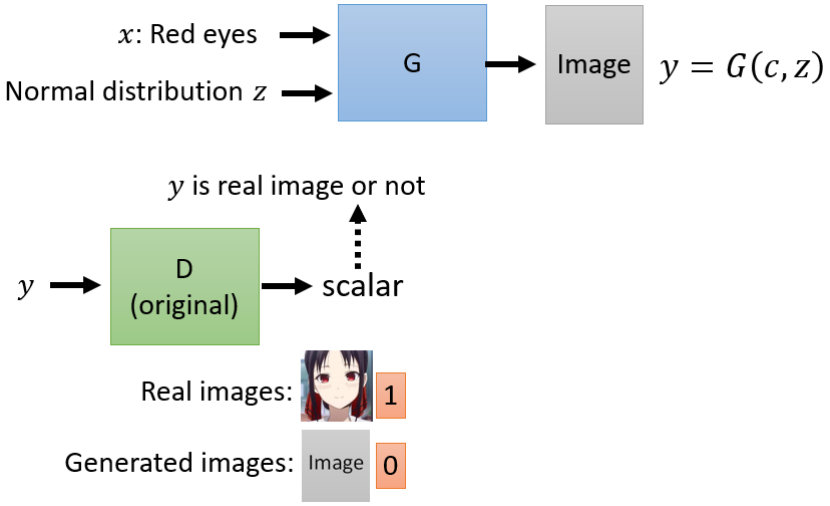
这样的模型其中的一个应用就是，可以做文字到图片的生成，这其实是一个 Supervised Learning 的问题，你需要一些有标签的数据，就比如搜集一些人脸，这些人脸都要有对应的文字的描述，告诉我们说，这个是红眼睛,这个是黑头发等等。



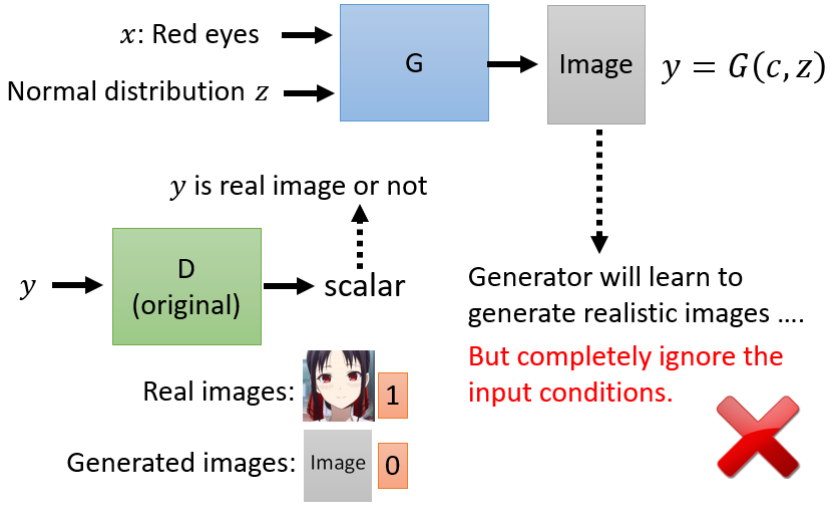
所以在 Text To Image 这样的任务里面，我们的 **x 就是一段文字**，然后我们期待输入 Red Eyes，机器就可以画一个红眼睛的角色，但每次画出来的角色都不一样，画出来什么样的角色，取决于你 Sample 到什么样的 z，这个就是 Text To Image 想要做的事情。

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 就是输入一张图片 y ，输出一个数值，这个数值代表输入的图片多像真实的图片：

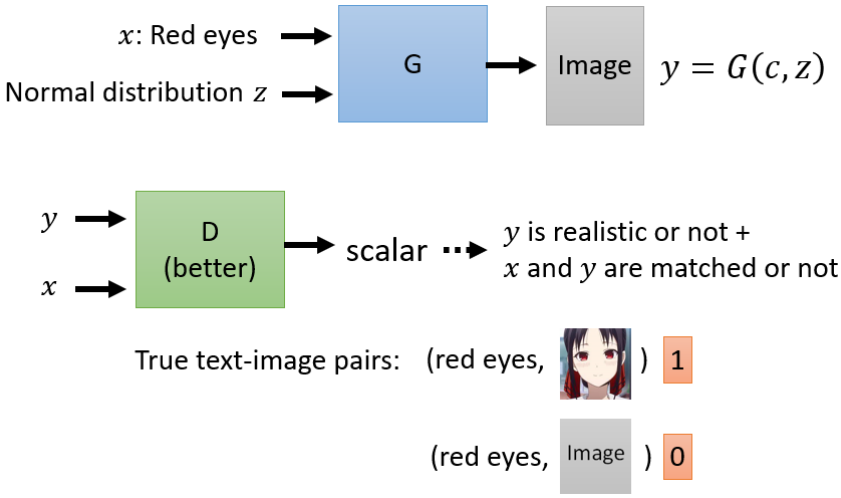


但这样的方法没有办法真的解决 Conditional GAN 的问题，因为存在这样的情况，我们的 Generator 会产生清晰的图片，但是**跟输入 x 完全没有任何关系**，因为对 Generator 来说，它只要产生清晰的图片就可以骗过 Discriminator 了，它何必要去管 Input 文字是什么，但这**显然不是我们要的。**



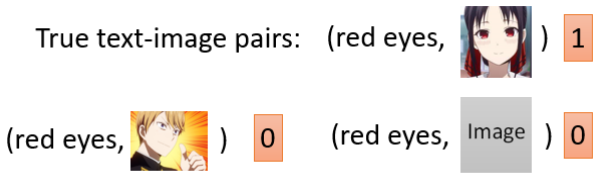
所以在 Conditional GAN 里面，我们的 **Discriminator 不是只输入图片 y**，它**还要输入 Condition x。**然后产生一个数值，那这个数值不只是看 y 好不好，还要看这个**图片跟文字的叙述是否匹配。**

那怎样训练这样的 Discriminator 呢？ 我们需要文字跟图片成对的资料来训练。



有这些成对资料，那你就告诉你的 Discriminator，看到这些真正的成对的资料就给它一分，看到 Red Eyes，但是搭配的是机器产生出来的杂讯图片，那就给 0 分。这样训练下去，就可以训练一个 conditional GAN。

但在实际操作上还需要加上一种状况是，**已经产生了好的图片，但是文字叙述对不上的状况。**



所以**我们通常会把训练资料拿一部分出来，然后故意把文字跟图片配一些错的**，然后告诉你的 Discriminator，看到这种状况也要说是不好的，这样反复的训练下去，最后才会得到好的结果。