# One Day One GAN

Hi ,my name is Chen Yang ,I am a sophomore in ocean university of China .I do some scientific research in my spare time. Based on the current hot direction of artificial intelligence, I hope to share my research progress with **Generative adversarial network**

嗨，我的名字是陈扬，我是中国海洋大学的二年级学生。我在业余时间做了一些科学研究。基于当前人工智能的热点方向，我希望与**生成对抗网络分享我的研究进展**

## 前言

**ODOG**,顾名思义就我我希望能每天抽出一个小时的时间来讲讲到目前为止,GAN的前沿发展和研究,笔者观察了很多深度学习的应用,特别是在图像这一方面,GAN已经在扮演着越来越重要的角色,我们经常可以看到老黄的NVIDIA做了各种各样的application,而且其中涉及到了大量GAN的理论及其实现,再者笔者个人也觉得目前国内缺少GAN在pytorch,keras,tensorflow等主流的框架下的实现教学.

我的老师曾经对我说过:"**深度学习是一块未知的新大陆,它是一个大的黑箱系统,而GAN则是黑箱中的黑箱,谁要是能打开这个盒子,将会引领一个新的时代**"

## infoGAN

论文地址：[InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets](http://arxiv.org/abs/1606.03657)

源码地址：[InfoGAN in TensorFlow](https://github.com/JonathanRaiman/tensorflow-infogan)

生成对抗网络（Generative Adversarial Nets）是一类新兴的生成模型，由两部分组成：一部分是判别模型（discriminator），用来判别输入数据是真实数据还是生成出来的数据；另一部分是是生成模型（generator），由输入的噪声生成目标数据。GAN 的优化问题可以表示为：

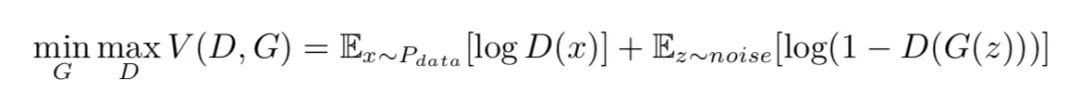


image-20190406235920213

其中$ Pdata$ 是生成样本，$noise c$ 来表示这一标签，作为生成模型的一个输入，这样我们有：

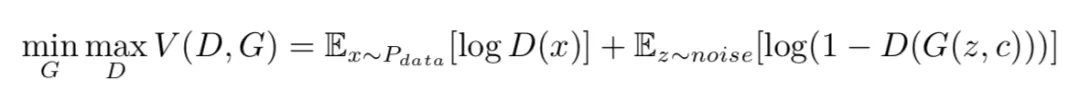


image-20190407000040736

然而当我们遇到存在潜在的类别差别而没有标签数据，要使 GAN 能够在这类数据上拥有更好表现，我们就需要一类能够无监督地辨别出这类潜在标签的数据，InfoGAN 就给出了一个较好的解决方案----利用互信息来对c进行约束，这是因为如果c对于生成数据G(z,c)具有可解释性，那么c和G(z,c)应该具有高度相关性，即互信息大，而如果是无约束的话，那么它们之间没有特定的关系，即互信息接近于0。因此我们希望c与G(z,c)的互信息I(c;G(z,c))越大越好，因此，模型的目标函数也变为：

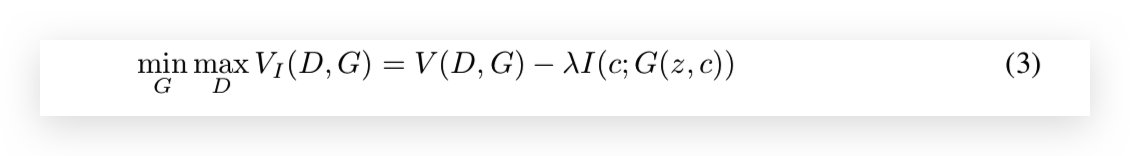


image-20190407005056917

但是在I(c;G(z,c))的计算中，真实的P(c|x)并不清楚，因此在具体的优化过程中，作者采用了变分推断的思想，引入了变分分布Q(c|x)来逼近P(c|x)，它是基于最优互信息下界的轮流迭代实现最终的求解，于是InfoGAN的目标函数变为：

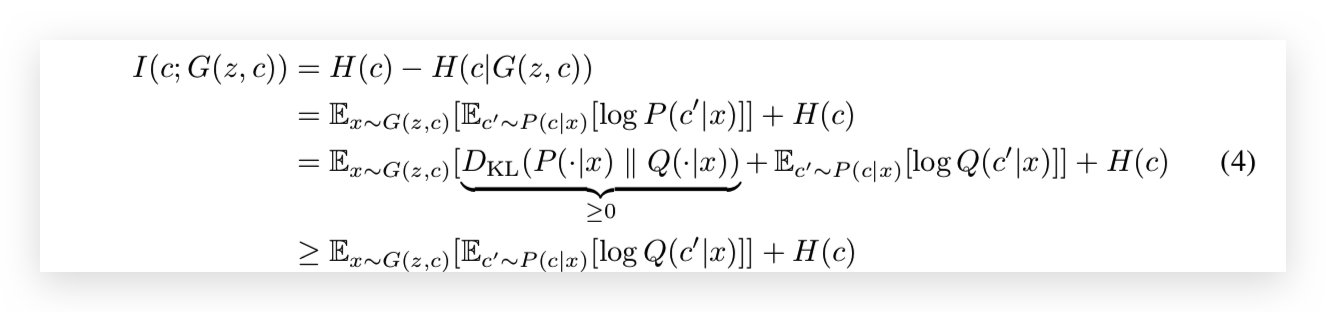


image-20190407004821018

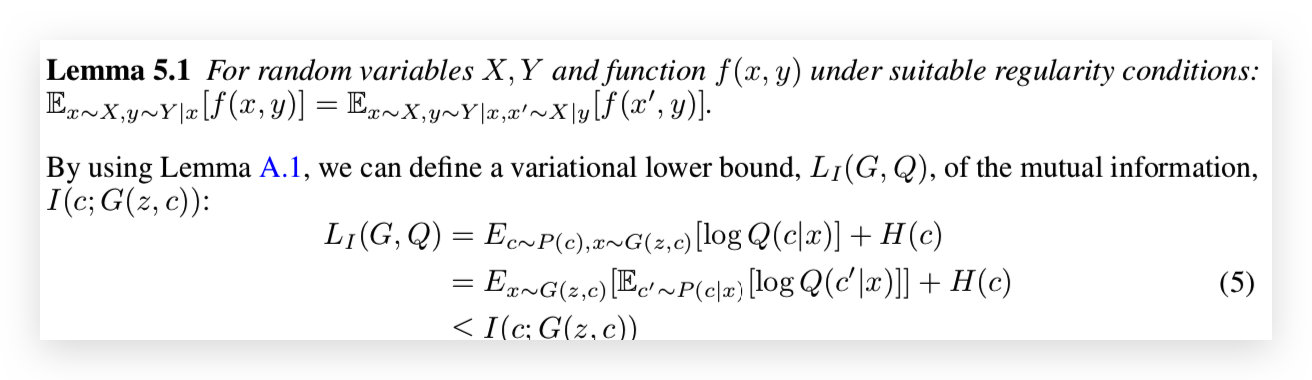


image-20190407004904021

把作为互信息的下界带入(3)式,使KL 散度最小化,用蒙特卡罗模拟（Monte Carlo simulation）去逼近$ L\_I (G, Q) $

==>

==>

所以:

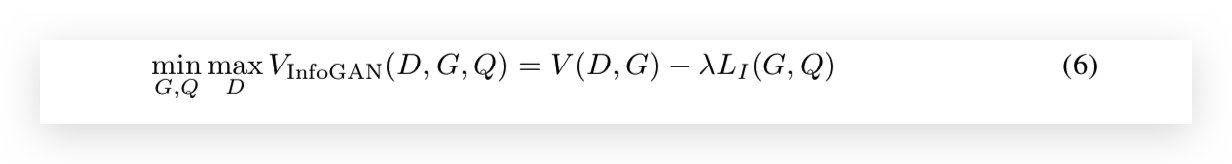


image-20190407005215167

那么这里整个网络的架构图就是:

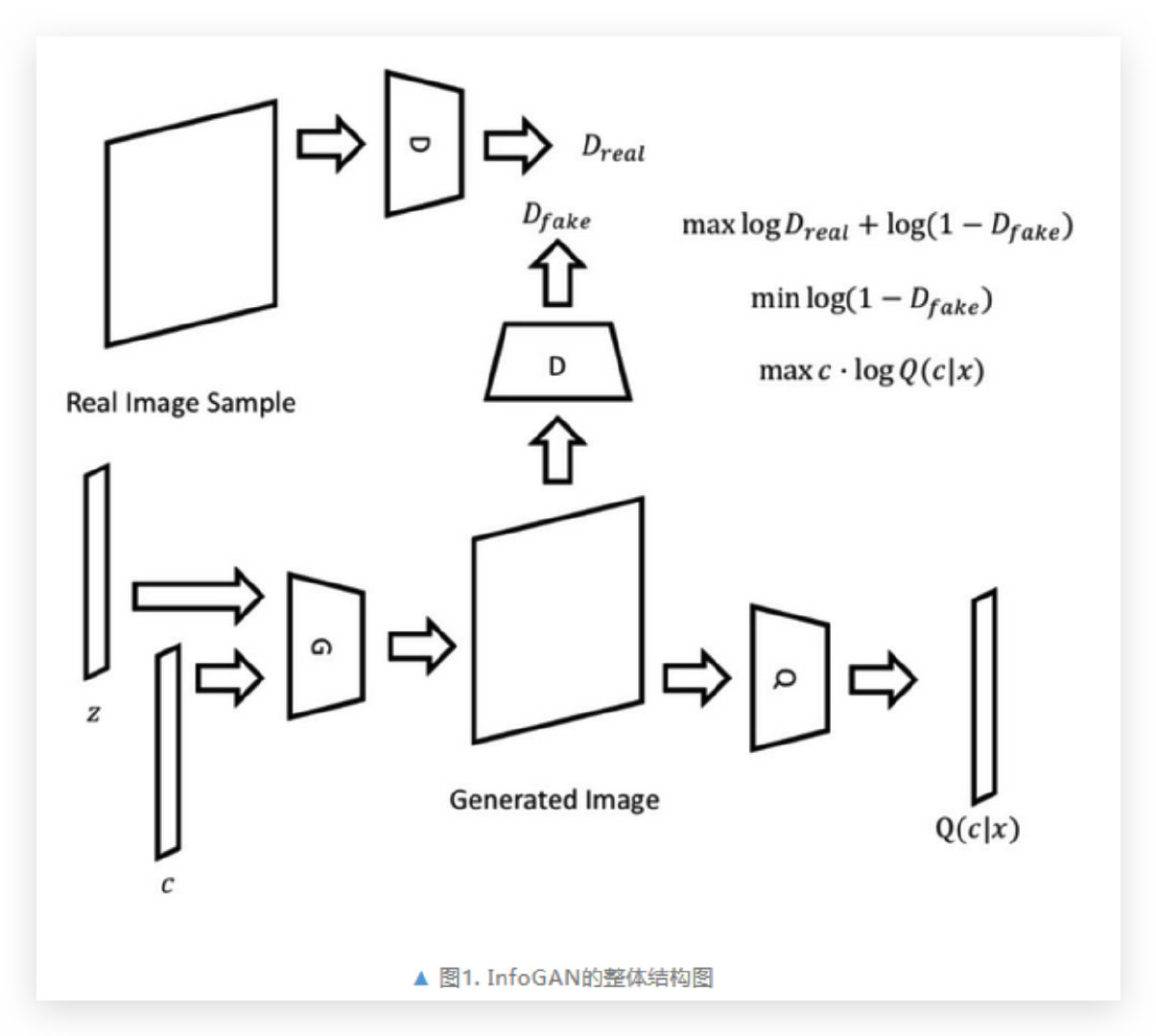


image-20190407004702678

### 互信息

简单的来说:互信息指的是两个随机变量之间的关联程度，即给定一个随机变量后，另一个随机变量不确定性的削弱程度，因而互信息取值最小为0，意味着给定一个随机变量对确定一另一个随机变量没有关系，最大取值为随机变量的熵，意味着给定一个随机变量，能完全消除另一个随机变量的不确定性

在[概率论](https://zh.wikipedia.org/wiki/%E6%A6%82%E7%8E%87%E8%AE%BA)和[信息论](https://zh.wikipedia.org/wiki/%E4%BF%A1%E6%81%AF%E8%AE%BA)中，两个[随机变量](https://zh.wikipedia.org/wiki/%E9%9A%8F%E6%9C%BA%E5%8F%98%E9%87%8F)的**互信息**（Mutual Information，简称MI）或**转移信息**（transinformation）是变量间相互依赖性的量度。不同于相关系数，互信息并不局限于实值随机变量，它更加一般且决定着联合分布 p(X,Y) 和分解的边缘分布的乘积 p(X)p(Y) 的相似程度。互信息是[点间互信息](https://zh.wikipedia.org/w/index.php?title=%E7%82%B9%E9%97%B4%E4%BA%92%E4%BF%A1%E6%81%AF&action=edit&redlink=1)（PMI）的期望值。互信息最常用的[单位](https://zh.wikipedia.org/wiki/%E8%AE%A1%E9%87%8F%E5%8D%95%E4%BD%8D)是[bit](https://zh.wikipedia.org/wiki/%E4%BD%8D%E5%85%83)。

一般地，两个离散随机变量 X 和 Y 的互信息可以定义为：

其中 p(x,y) 是 X 和 Y 的联合概率分布函数，而$ {p(x)} p(x) 和 {p(y)} p(y) $分别是 X 和 Y 的边缘概率分布函数。

$ I(X;Y)=*{{yY}}*{{xX}}p(x,y),,!$

在连续随机变量的情形下，求和被替换成了二重定积分：

其中 p(x,y) 当前是 X 和 Y 的联合概率密度函数，而$ {p(x)} p(x) 和 {p(y)} p(y) $分别是 X 和 Y 的边缘概率密度函数。

$ I(X;Y)=*{Y}*{X}p(x,y);dx,dy,$

如果对数以 2 为基底，互信息的单位是bit。

直观上，互信息度量 X 和 Y 共享的信息：它度量知道这两个变量其中一个，对另一个不确定度减少的程度。例如，如果 X 和 Y 相互独立，则知道 X 不对 Y 提供任何信息，反之亦然，所以它们的互信息为零。在另一个极端，如果 X 是 Y 的一个确定性函数，且 Y 也是 X 的一个确定性函数，那么传递的所有信息被 X 和 Y 共享：知道 X 决定 Y 的值，反之亦然。因此，在此情形互信息与 Y（或 X）单独包含的不确定度相同，称作 Y（或 X）的熵。而且，这个互信息与 X 的熵和 Y 的熵相同。（这种情形的一个非常特殊的情况是当 X 和 Y 为相同随机变量时。）

互信息是 X 和 Y 的联合分布相对于假定 X 和 Y 独立情况下的联合分布之间的内在依赖性。 于是互信息以下面方式度量依赖性：I(X; Y) = 0 当且仅当 X 和 Y 为独立随机变量。从一个方向很容易看出：当 X 和 Y 独立时，p(x,y) = p(x) p(y)，因此：

此外，互信息是非负的，而且是对称的（即 I(X;Y) = I(Y;X)）。

### 实现细节

# Build and the 判别器 and 识别网络  
self.discriminator, self.auxilliary =self.build\_disk\_and\_q\_net()

# Build the generator  
self.generator = self.build\_generator()  
  
# The generator takes noise and the target label as input  
# and generates the corresponding digit of that label  
gen\_input = Input(shape=(self.latent\_dim,))  
img = self.generator(gen\_input)  
#生成器依赖标签信息

def build\_generator(self):  
  
 model = Sequential()  
  
 model.add(Dense(128 \* 7 \* 7, activation="relu", input\_dim=self.latent\_dim))  
 model.add(Reshape((7, 7, 128)))  
 model.add(BatchNormalization(momentum=0.8))  
 model.add(UpSampling2D())  
 model.add(Conv2D(128, kernel\_size=3, padding="same"))  
 model.add(Activation("relu"))  
 model.add(BatchNormalization(momentum=0.8))  
 model.add(UpSampling2D())  
 model.add(Conv2D(64, kernel\_size=3, padding="same"))  
 model.add(Activation("relu"))  
 model.add(BatchNormalization(momentum=0.8))  
 model.add(Conv2D(self.channels, kernel\_size=3, padding='same'))  
 model.add(Activation("tanh"))  
  
 gen\_input = Input(shape=(self.latent\_dim,))  
 img = model(gen\_input)  
  
 model.summary()  
  
 return Model(gen\_input, img)

生成器的输入是一个72维的向量,输出是一张图片

def build\_disk\_and\_q\_net(self):  
  
 img = Input(shape=self.img\_shape)  
  
 # Shared layers between discriminator and recognition network  
 model = Sequential()  
 model.add(Conv2D(64, kernel\_size=3, strides=2, input\_shape=self.img\_shape, padding="same"))  
 model.add(LeakyReLU(alpha=0.2))  
 model.add(Dropout(0.25))  
 model.add(Conv2D(128, kernel\_size=3, strides=2, padding="same"))  
 model.add(ZeroPadding2D(padding=((0,1),(0,1))))  
 model.add(LeakyReLU(alpha=0.2))  
 model.add(Dropout(0.25))  
 model.add(BatchNormalization(momentum=0.8))  
 model.add(Conv2D(256, kernel\_size=3, strides=2, padding="same"))  
 model.add(LeakyReLU(alpha=0.2))  
 model.add(Dropout(0.25))  
 model.add(BatchNormalization(momentum=0.8))  
 model.add(Conv2D(512, kernel\_size=3, strides=2, padding="same"))  
 model.add(LeakyReLU(alpha=0.2))  
 model.add(Dropout(0.25))  
 model.add(BatchNormalization(momentum=0.8))  
 model.add(Flatten())  
  
 img\_embedding = model(img)  
  
 # Discriminator  
 validity = Dense(1, activation='sigmoid')(img\_embedding)  
  
 # Recognition  
 q\_net = Dense(128, activation='relu')(img\_embedding)  
 label = Dense(self.num\_classes, activation='softmax')(q\_net)  
  
 # Return discriminator and recognition network  
 return Model(img, validity), Model(img, label)

判别器要判别真假,q\_net 和要判别 label.

def sample\_generator\_input(self, batch\_size):  
 # Generator inputs  
 sampled\_noise = np.random.normal(0, 1, (batch\_size, 62))  
 sampled\_labels = np.random.randint(0, self.num\_classes, batch\_size).reshape(-1, 1)  
 sampled\_labels = to\_categorical(sampled\_labels, num\_classes=self.num\_classes)

噪声 Z 的输入包括高斯噪声和一个10维的 label-one-hot 向量.

其实写到这里,我发现了一点啊,他真的和 ACGAN 好像啊……

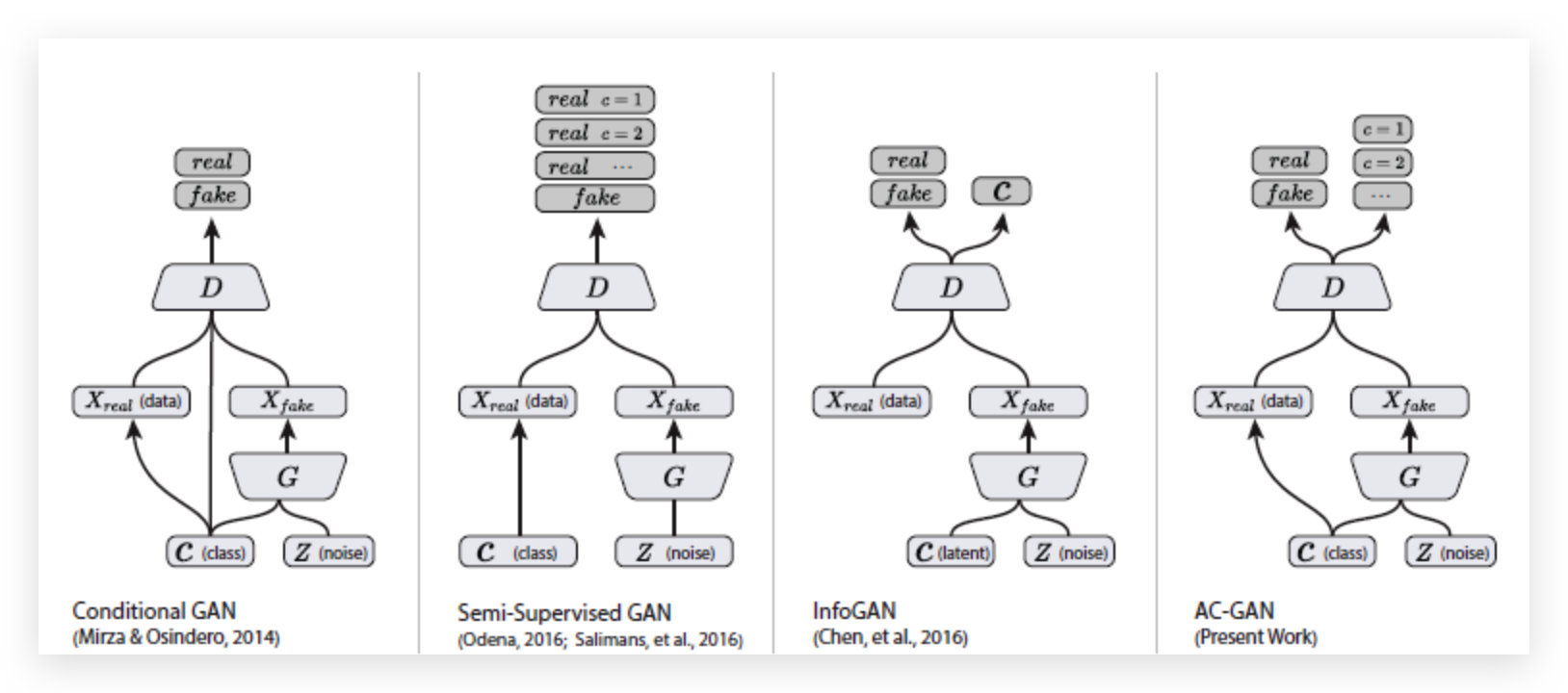


image-20190407011433732

像到什么程度,自行体会,不过说实话原理还是有所不同的,说实话,我很无语

接下来继续填 semi-supervised GAN 的坑吧,最近self supervised GAN 出来了,行有余力可以看看.

总感觉接下来要开 cycleGAN 系列了,这个有意思.看到这里了要是看官老爷觉得interesting,不如麻烦关注转发好看三连一波,你的每一点小小的鼓励都是对作者莫大的鼓舞鸭😄.

参考:

https://www.zhihu.com/question/24059517/answer/37430101

https://blog.csdn.net/wspba/article/details/54808833

<https://www.jiqizhixin.com/articles/2018-10-29-21>

<https://aistudio.baidu.com/aistudio/#/projectdetail/29156>

<https://aistudio.baidu.com/aistudio/#/projectdetail/29156>

<https://zh.wikipedia.org/zh-hant/%E4%BA%92%E4%BF%A1%E6%81%AF>