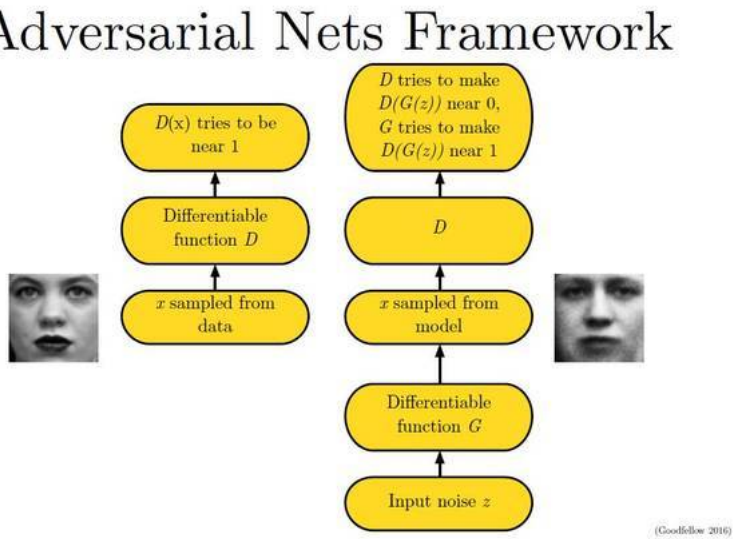
生成对抗网络GAN & DCGAN

A brief introduction on GAN and DCGAN according to my own acknowledge.

GAN简单原理介绍

> GAN的网络结构包含了两个部分：Discriminator（判别器） 和 Generator （生成器）
>
> > **A.** $G()$是一个生成图像的网络，$G()$ 接收一个噪声 z， 生成的图片为 $G(z)$。
> > **B.** $D()$以图像作为输入，用来判别 $G(z)$的正确性，输出 $D(G(z))$是推断图片为真实的概率，取值范围 $0-1$。

> 简单讲，生成对抗网络的训练过程就是，就是 $G(z)$和 $D(x)$之间的博弈。生成器目标是生成的图片让判别器难辨真假; 判别器的目标是不断提升自身判断的能力。
> 博弈到理想情况时，$D(G(z))=0.5$ (难辨真假)。

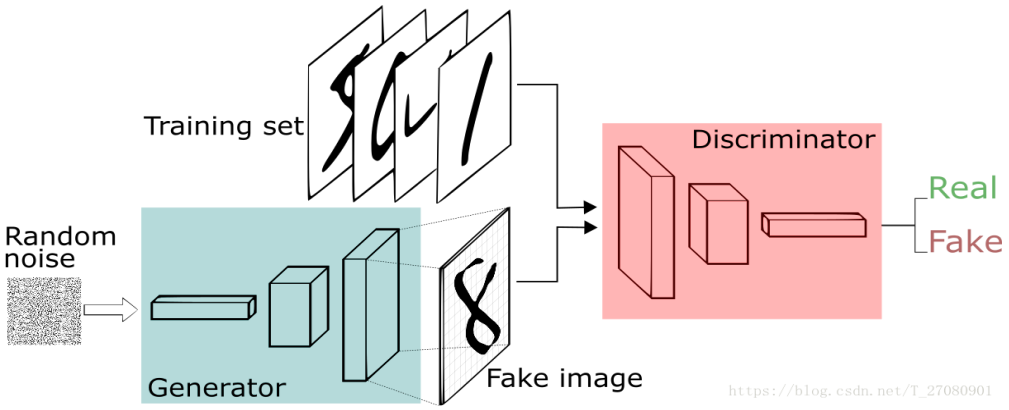


生成对抗网络的应用

1. 通过无监督的方式，使网络生成难辨真假的图片；
2. 训练好的生成器网络 和 判别器网络，可作为一种优秀的特征表示，用于其他任务。

训练过程

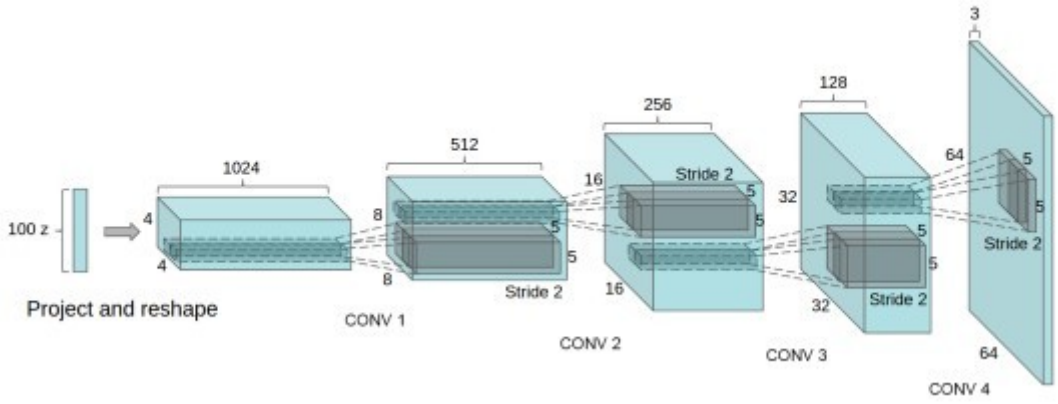
> 一个step的训练分为两部分：
> ① 先采样一部分真实图片 + 一部分 $G()$ 生成的假图片，训练判别器。目的是让判别器学会分辨真假。
> ② 冻结判别器 $D()$ 所有参数（参数不更新），采样一部分 $noises$，输入 $G()$，生成的假图片 $G(noises)$ 输入判别器 $D()$ 中， 此时给假图片打上为“真”的标签。
> ③ 循环迭代①②




```
# 训练
for epoch in range(epochs):
    for i in range(int(x_train.shape[0]/batch_size)):

        ################  先训练判别器  ################

        noises = np.random.uniform(-1, 1, size=(batch_size, LATENT_DIM))
        generated_imgs = g.predict(noises)
        generated_imgs_labels = [0]*batch_size
        imgs = x_train[i*batch_size : (i+1)*batch_size]
        imgs_labels = [1]*batch_size
        x_d = np.concatenate((imgs, generated_imgs))
        y_d = np.concatenate((imgs_labels, generated_imgs_labels))
        # 训练判别器
        d_loss = d.train_on_batch(x_d, y_d)
        # 注意：冻结判别器参数
        d.trainable = False

        ################ 再训练生成器  ################

        noises = np.random.uniform(-1, 1, size=(batch_size, LATENT_DIM))
        noises_labels = [1]*batch_size
        # 训练生成器网络参数
        a_loss = a.train_on_batch(noises, noises_labels)
        # 解锁判别器参数，进行下一次迭代
        d.trainable = True
```

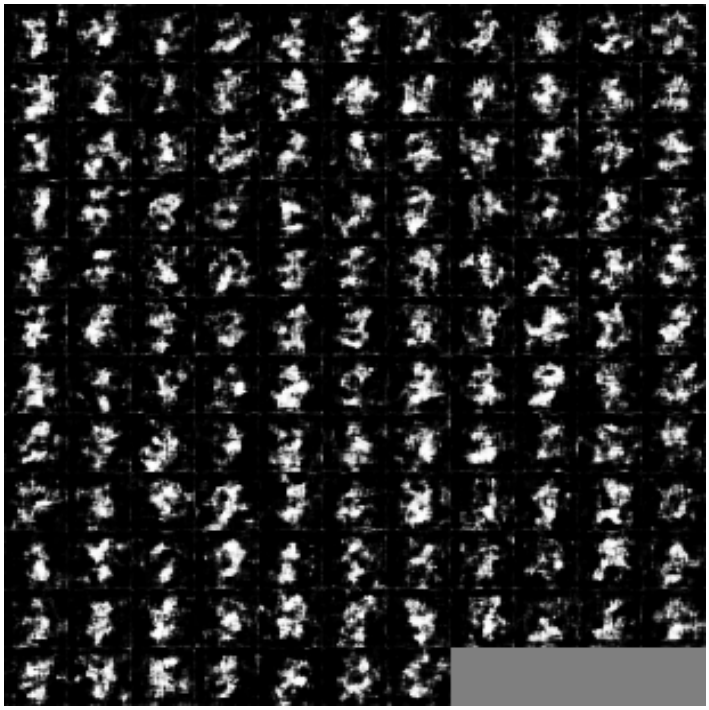
Deep Convolutional Generative Adversatial Networks

> 与GAN的最大区别在于，把Generator中的全连接部分换成了全卷积。
>
> - 取消所有 $pooling$ 层。$G()$ 网络中使用转置卷积 transposed convolutional layer进行上采样，$D()$ 网络中用加入 $stride$ 的卷积代替 $pooling$；
> - 在 $D()$ 和 $G()$ 中均使用 $BatchNormalization$；
> - 去掉 FC 层，使网络变为全卷积网络；
> - $G()$ 网络中使用 $ReLU$ 作为激活函数，最后一层使用 $tanh$；
> - $D()$ 网络中使用 $LeakyReLU$ 作为激活函数 。

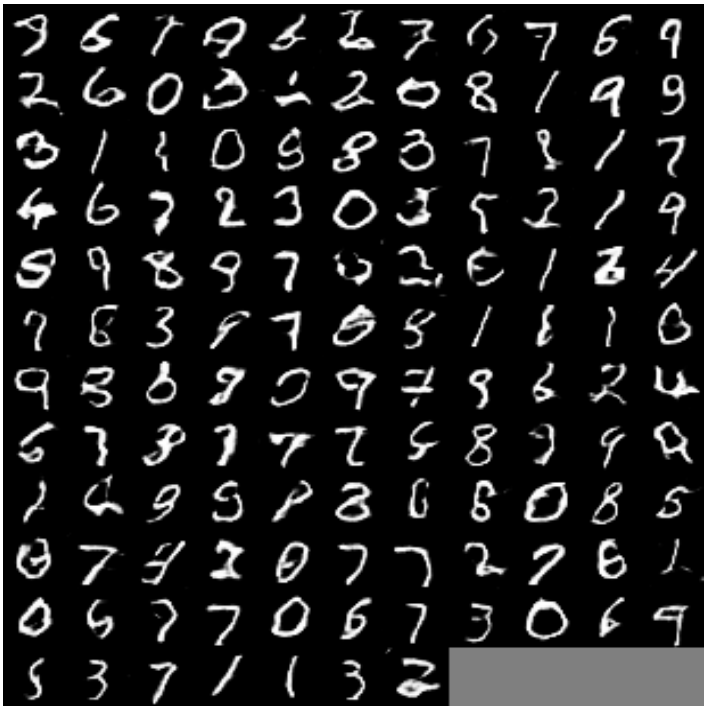


Training Results

1 epochs



50 epochs

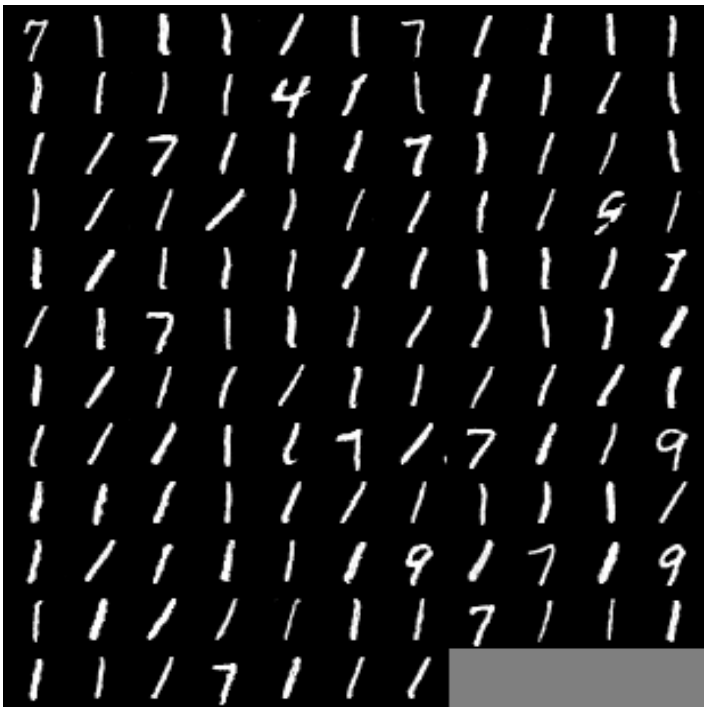


100 epochs

150 epochs

200 epochs

600 epochs



> 600 epochs时，由noises生成的图片中只包含了数字1,4,7,9,其他数字基本不出现。
> 可能是这几个数字最有利于 **生成器** 生成让 **判别器** 认为是真实的照片。