前言

GAN 网络是近两年深度学习领域的新秀，火的不行，本文旨在浅显理解传统 GAN，分享学习心得。现有 GAN 网络大多数代码实现使用 Python、torch 等语言，这里，后面用 matlab 搭建一个简单的 GAN 网络，便于理解 GAN 原理。

GAN 的鼻祖之作是 2014 年 NIPS 一篇文章：Generative Adversarial Net（https://arxiv.org/abs/1406.2661）,可以细细品味。

- 分享一个目前各类 GAN 的一个论文整理集合

https://deephunt.in/the-gan-zoo-79597dc8c347

- 再分享一个目前各类 GAN 的一个代码整理集合

https://github.com/zhangqianhui/AdversarialNetsPapers

开始

我们知道 GAN 的思想是是一种二人零和博弈思想（two-player game），博弈双方的利益之和是一个常数，比如两个人掰手腕，假设总的空间是一定的，你的力气大一点，那你就得到的空间多一点，相应的我的空间就少一点，相反我力气大我就得到的多一点，但有一点是确定的就是，我两的总空间是一定的，这就是二人博弈，但是呢总利益是一定的。

引申到 GAN 里面就是可以看成，GAN 中有两个这样的博弈者，一个人名字是生成模型（G），另一个人名字是判别模型（D）。他们各自有各自的功能。

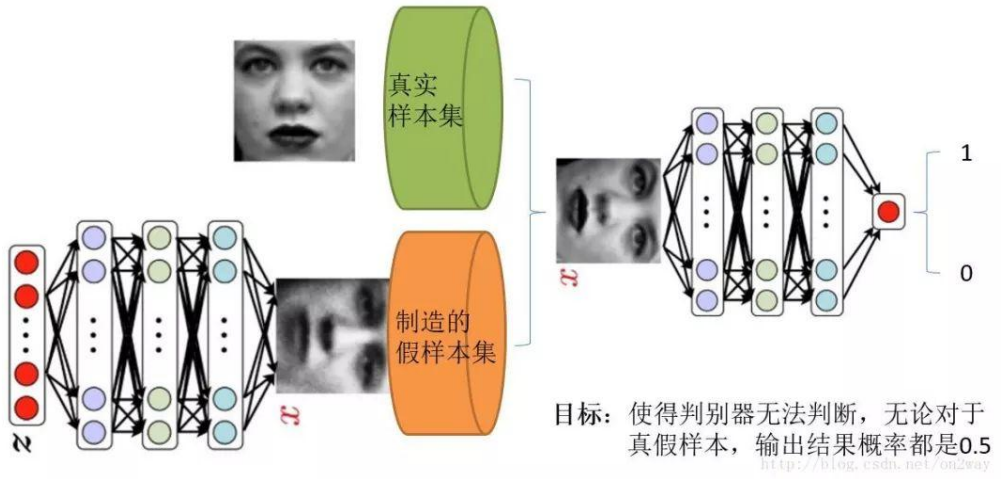
相同点是：

- 这两个模型都可以看成是一个黑匣子，接受输入然后有一个输出，类似一个函数，一个输入输出映射。

不同点是：

- 生成模型功能：比作是一个样本生成器，输入一个噪声/样本，然后把它包装成一个逼真的样本，也就是输出。
- 判别模型：比作一个二分类器（如同 0-1 分类器），来判断输入的样本是真是假。（就是输出值大于 0.5 还是小于 0.5）

直接上一张个人觉得解释的好的图说明：



在之前，我们首先明白在使用 GAN 的时候的 2 个问题

- **我们有什么？**

  比如上面的这个图，我们有的只是真实采集而来的人脸样本数据集，仅此而已，而且很关键的一点是我们连人脸数据集的类标签都没有，也就是我们不知道那个人脸对应的是谁。

- **我们要得到什么？**

  至于要得到什么，不同的任务得到的东西不一样，我们只说最原始的 GAN 目的，那就是我们想通过输入一个噪声，模拟得到一个人脸图像，这个图像可以非常逼真以至于以假乱真。

好了再来理解下 GAN 的两个模型要做什么。

首先**判别模型**，就是图中右半部分的网络，直观来看就是一个简单的神经网络结构，输入就是一副图像，输出就是一个概率值，用于判断真假使用（概率值大于 0.5 那就是真，小于 0.5 那就是假），真假也不过是人们定义的概率而已。

其次是**生成模型**，生成模型要做什么呢，同样也可以看成是一个神经网络模型，输入是一组随机数 Z，输出是一个图像，不再是一个数值而已。从图中可以看到，会存在两个数据集，一个是真实数据集，这好说，另一个是假的数据集，那这个数据集就是有生成网络造出来的数据集。好了根据这个图我们再来理解一下 GAN 的目标是要干什么：

- **判别网络的目的**：就是能判别出来属于的一张图它是来自真实样本集还是假样本集。假如输入的是真样本，网络输出就接近 1，输入的是假样本，网络输出接近 0，那么很完美，达到了很好判别的目的。

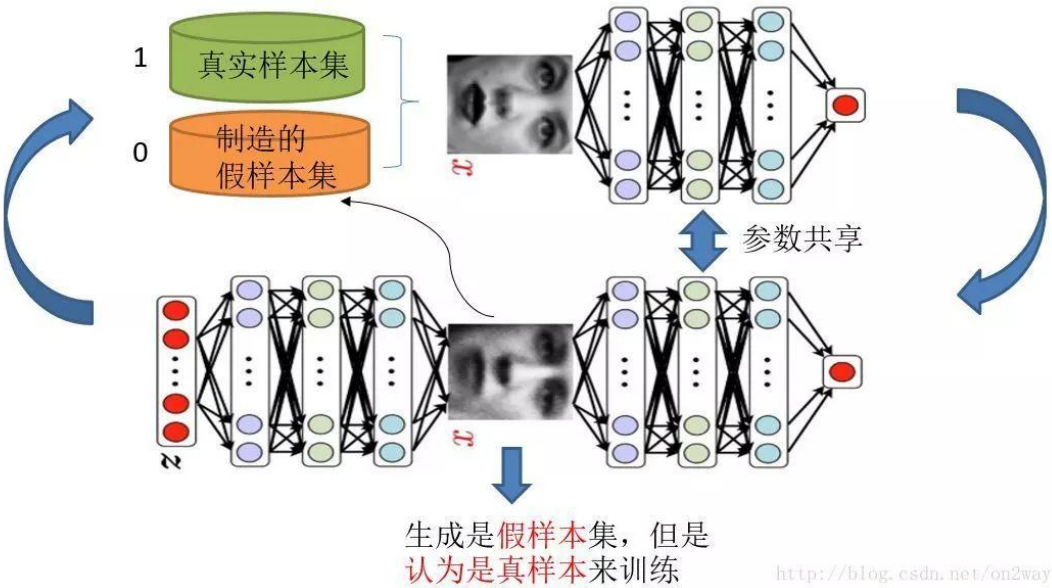
- **生成网络的目的**：生成网络是造样本的，它的目的就是使得自己造样本的能力尽可能强，强到什么程度呢，你判别网络没法判断我是真样本还是假样本。

有了这个理解我们再来看看为什么叫做对抗网络了。判别网络说，我很强，来一个样本我就知道它是来自真样本集还是假样本集。生成网络就不服了，说我也很强，我生成一个假样本，虽然我生成网络知道是假的，但是你判别网络不知道呀，我包装的非常逼真，以至于判别网络无法判断真假，那么用输出数值来解释就是，生成网络生成的假样本进去了判别网络以后，判别网络给出的结果是一个接近 0.5 的值，极限情况就是 0.5，也就是说判别不出来了，这就是纳什平衡了。

由这个分析可以发现，生成网络与判别网络的目的正好是相反的，一个说我能判别的好，一个说我让你判别不好。所以叫做对抗，叫做博弈。那么最后的结果到底是谁赢呢？这就要归结到设计者，也就是我们希望谁赢了。作为设计者的我们，我们的目的是要得到以假乱真的样本，那么很自然的我们希望生成样本赢了，也就是希望生成样本很真，判别网络能力不足以区分真假样本位置。

再理解

知道了 GAN 大概的目的与设计思路，那么一个很自然的问题来了就是我们该如何用数学方法解决这么一个对抗问题。这就涉及到如何训练这样一个生成对抗网络模型了，还是先上一个图，用图来解释最直接：



需要注意的是生成模型与对抗模型可以说是完全独立的两个模型，好比就是完全独立的两个神经网络模型，他们之间没有什么联系。

好了那么训练这样的两个模型的大方法就是：**单独交替迭代训练**。

什么意思？因为是 2 个网络，不好一起训练，所以才去交替迭代训练，我们一一来看。

假设现在生成网络模型已经有了（当然可能不是最好的生成网络），那么给一堆随机数组，就会得到一堆假的样本集（因为不是最终的生成模型，那么现在生成网络可能就处于劣势，导致生成的样本就不咋地，可能很容易就被判别网络判别出来了说这货是假冒的），但是先不管这个，假设我们现在有了这样的假样本集，真样本集一直都有，现在我们人为的定义真假样本集的标签，因为我们希望真样本集的输出尽可能为 1，假样本集为 0，很明显这里我们就已经默认真样本集所有的类标签都为 1，而假样本集的所有类标签都为 0.

有人会说，在真样本集里面的人脸中，可能张三人脸和李四人脸不一样呀，对于这个问题我们需要理解的是，我们现在的任务是什么，我们是想分样本真假，而不是分真样本中那个是张三 label、那个是李四 label。况且我们也知道，原始真样本的 label 我们是不知道的。回过头来，我们现在有了真样本集以及它们的 label（都是 1）、假样本集以及它们的 label（都是 0），这样单就判别网络来说，此时问题就变成了一个再简单不过的**有监督的二分类问题**了，直接送到神经网络模型中训练就完事了。假设训练完了，下面我们来看生成网络。

对于生成网络，想想我们的目的，是生成尽可能逼真的样本。那么原始的生成网络生成的样本你怎么知道它真不真呢？就是送到判别网络中，所以在训练生成网络的时候，我们需要联合判别网络一起才能达到训练的目的。什么意思？就是如果我们单单只用生成网络，那么想想我们怎么去训练？误差来源在哪里？细想一下没有，但是如果我们把刚才的判别网络串接在生成网络的后面，这样我们就知道真假了，也就有了误差了。所以对于生成网络的训练其实是对生成-判别网络串接的训练，就像图中显示的那样。好了那么现在来分析一下样本，原始的噪声数组 Z 我们有，也就是生成了假样本我们有，此时很关键的一点来了，我们要**把这些假样本的标签都设置为 1**，也就是认为这些假样本在生成网络训练的时候是真样本。

那么为什么要这样呢？我们想想，是不是这样才能起到迷惑判别器的目的，也才能使得生成的假样本逐渐逼近为正样本。好了，重新顺一下思路，现在对于生成网络的训练，我们有了样本集（只有假样本集，没有真样本集），有了对应的 label（全为 1），是不是就可以训练了？有人会问，这样只有一类样本，训练啥呀？谁说一类样本就不能训练了？只要有误差就行。还有人说，你这样一训练，**判别网络的网络参数不是也跟着变吗**？没错，这很关键，所以在训练这个串接的网络的时候，一个很重要的操作就是**不要判别网络的参数发生变化**，也就是不让它参数发生更新，只是把误差一直传，传到生成网络那块后更新生成网络的参数。这样就完成了生成网络的训练了。

在完成生成网络训练好，那么我们是不是可以根据目前新的生成网络再对先前的那些噪声 Z 生成新的假样本了，没错，并且训练后的假样本应该是更真了才对。然后又有了新的真假样本集（其实是新的假样本集），这样又可以重复上述过程了。我们把这个过程称作为单独交替训练。我们可以实现定义一个迭代次数，交替迭代到一定次数后停止即可。这个时候我们再去看一看噪声 Z 生成的假样本会发现，原来它已经很真了。

看完了这个过程是不是感觉 GAN 的设计真的很巧妙，个人觉得最值得称赞的地方可能在于这种**假样本在训练过程中的真假变换**，这也是博弈得以进行的关键之处。

进一步

文字的描述相信已经让大多数的人知道了这个过程，下面我们来看看原文中几个重要的数学公式描述，首先我们直接上原始论文中的目标公式吧：

$$\min_G \max_D V(D,G) = E_{x\sim p_{data}(x)}[log(D(x))] + E_{z\sim p_z(z)}[log(1-D(G(z)))]$$

上述这个公式说白了就是一个最大最小优化问题，其实对应的也就是上述的两个优化过程。有人说如果不看别的，能达看到这个公式就拍案叫绝的地步，那就是机器学习的顶级专家，哈哈，真是前路漫漫。同时也说明这个简单的公式意义重大。

这个公式既然是最大最小的优化，那就不是一步完成的，其实对比我们的分析过程也是这样的，这里现优化 D，然后在取优化 G，本质上是两个优化问题，把拆解就如同下面两个公式：
优化 D：

$$\max_D V(D,G) = E_{x\sim p_{data}(x)}[log(D(x))] + E_{z\sim p_z(z)}[log(1-D(G(z)))]$$

优化 G：

$$\min_G V(D,G) = E_{z\sim p_z(z)}[log(1-D(G(z)))]$$

可以看到，优化 D 的时候，也就是判别网络，其实没有生成网络什么事，后面的 G(z) 这里就相当于已经得到的假样本。优化 D 的公式的第一项，使的真样本 x 输入的时候，得到的结果越大越好，可以理解，因为需要真样本的预测结果越接近于 1 越好嘛。对于假样本，需要优化是的其结果越小越好，也就是 D(G(z))越小越好，因为它的标签为 0。但是呢第一项是越大，第二项是越小，这不矛盾了，所以呢把第二项改成 1-D(G(z))，这样就是越大越好，两者合起来就是越大越好。 那么同样在优化 G 的时候，这个时候没有真样本什么事，所以把第一项直接却掉了。这个时候只有假样本，但是我们说这个时候是希望假样本的标签是 1 的，所以是 D(G(z))越大越好，但是呢为了统一成 1-D(G(z))的形式，那么只能是最小化 1-D(G(z))，本质上没有区别，只是为了形式的统一。之后这两个优化模型可以合并起来写，就变成了最开始的那个最大最小目标函数了。
所以回过头来我们来看这个最大最小目标函数，里面包含了判别模型的优化，包含了生成模型的以假乱真的优化，完美的阐释了这样一个优美的理论。

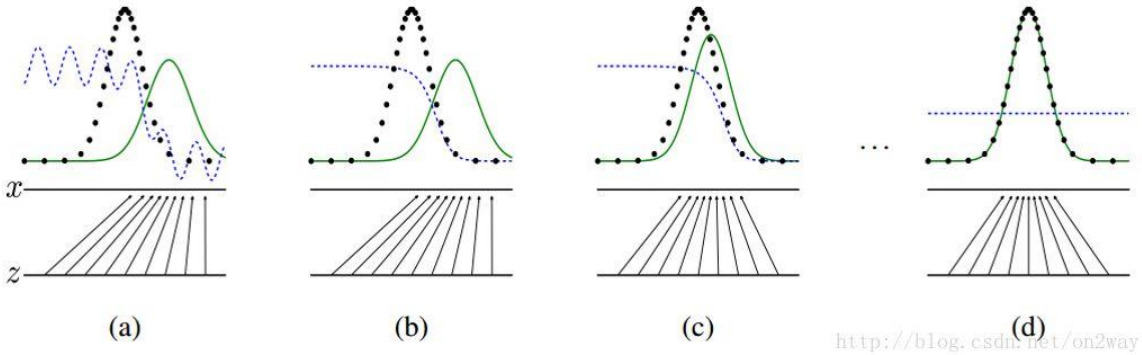
再进一步

有人说 GAN 强大之处在于可以自动的学习原始真实样本集的数据分布，不管这个分布多么的复杂，只要训练的足够好就可以学出来。针对这一点，感觉有必要好好理解一下为什么别人会这么说。
我们知道，传统的机器学习方法，我们一般都会定义一个什么模型让数据去学习。比如说假设我们知道原始数据属于高斯分布呀，只是不知道高斯分布的参数，这个时候我们定义高斯分布，然后利用数据去学习高斯分布的参数得到我们最终的模型。再比如说我们定义一个分类器，比如 SVM，然后强行让数据进行东变西变，进行各种高维映射，最后可以变成一个简单的分布，SVM 可以很轻易的进行二分类分开，其实 SVM

已经放松了这种映射关系了，但是也是给了一个模型，这个模型就是核映射（什么径向基函数等等），说白了其实也好像是你事先知道让数据该怎么映射一样，只是核映射的参数可以学习罢了。

所有的这些方法都在直接或者间接的告诉数据你该怎么映射一样，只是不同的映射方法能力不一样。那么我们再来看看 GAN，生成模型最后可以通过噪声生成一个完整的真实数据（比如人脸），说明生成模型已经掌握了从随机噪声到人脸数据的分布规律了，有了这个规律，想生成人脸还不容易。然而这个规律我们开始知道吗？显然不知道，如果让你说从随机噪声到人脸应该服从什么分布，你不可能知道。这是一层层映射之后组合起来的非常复杂的分布映射规律。然而 GAN 的机制可以学习到，也就是说 GAN 学习到了真实样本集的数据分布。

再拿原论文中的一张图来解释：



这张图表明的是 GAN 的生成网络如何一步步从均匀分布学习到正太分布的。原始数据 x 服从正太分布，这个过程你也没告诉生成网络说你得用正太分布来学习，但是生成网络学习到了。假设你改一下 x 的分布，不管什么分布，生成网络可能也能学到。这就是 GAN 可以自动学习真实数据的分布的强大之处。

还有人说 GAN 强大之处在于可以自动的定义潜在损失函数。什么意思呢，这应该说的是判别网络可以自动学习到一个好的判别方法，其实就是等效的理解为可以学习到好的损失函数，来比较好或者不好的判别出来结果。虽然大的 loss 函数还是我们人为定义的，基本上对于多数 GAN 也都这么定义就可以了，但是判别网络潜在学习到的损失函数隐藏在网络之中，不同的问题这个函数就不一样，所以说可以自动学习这个潜在的损失函数。

开始做小实验

本节主要实验一下如何通过随机数组生成 mnist 图像。mnist 手写体数据库应该都熟悉的。这里简单的使用 matlab 来实现，方便看到整个实现过程。这里用到了一个工具箱 DeepLearnToolbox,关于该工具箱的一些其他使用说明：

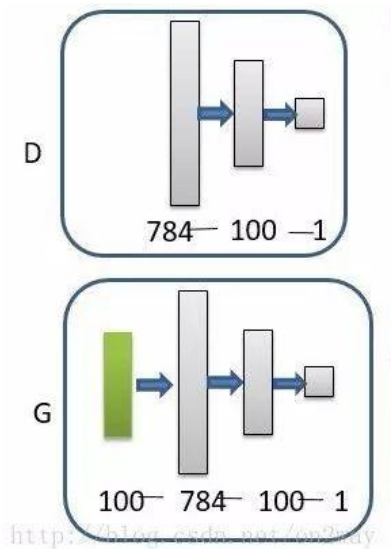
DeepLearnToolbox

https://github.com/rasmusbergpalm/DeepLearnToolbox

其他使用说明

https://blog.csdn.net/dark_scope/article/details/9447967

网络结构很简单，就定义成下面这样子：



将上述工具箱添加到路径，然后运行下面代码：

```
clc
clear
%% 构造真实训练样本 60000 个样本 1*784 维（28*28 展开）
load mnist_uint8;
train_x = double(train_x(1:60000,:)) / 255;
% 真实样本认为为标签 [1 0]； 生成样本为[0 1];
train_y = double(ones(size(train_x,1),1));
% normalize
train_x = mapminmax(train_x, 0, 1);
rand('state',0)
%% 构造模拟训练样本 60000 个样本 1*100 维
test_x = normrnd(0,1,[60000,100]); % 0-255 的整数
test_x = mapminmax(test_x, 0, 1);
test_y = double(zeros(size(test_x,1),1));
test_y_rel = double(ones(size(test_x,1),1));
%%
nn_G_t = nnsetup([100 784]);
nn_G_t.activation_function = 'sigm';
nn_G_t.output = 'sigm';
nn_D = nnsetup([784 100 1]);
nn_D.weightPenaltyL2 = 1e-4;  %  L2 weight decay
```

```
nn_D.dropoutFraction = 0.5;   %  Dropout fraction
nn_D.learningRate = 0.01;               %  Sigm require a lower learning rate
nn_D.activation_function = 'sigm';
nn_D.output = 'sigm';
% nn_D.weightPenaltyL2 = 1e-4;  %  L2 weight decay
nn_G = nnsetup([100 784 100 1]);
nn_G.weightPenaltyL2 = 1e-4;  %  L2 weight decay
nn_G.dropoutFraction = 0.5;   %  Dropout fraction
nn_G.learningRate = 0.01;               %  Sigm require a lower learning rate
nn_G.activation_function = 'sigm';
nn_G.output = 'sigm';
% nn_G.weightPenaltyL2 = 1e-4;  %  L2 weight decay
opts.numepochs =  1;        %  Number of full sweeps through data
opts.batchsize = 100;       %  Take a mean gradient step over this many samples
%%
num = 1000;
tic
for each = 1:1500
    %----------计算 G 的输出：假样本------------------
    for i = 1:length(nn_G_t.W)   %共享网络参数
        nn_G_t.W{i} = nn_G.W{i};
    end
    G_output = nn_G_out(nn_G_t, test_x);
    %-----------训练 D-----------------------------
    index = randperm(60000);
    train_data_D = [train_x(index(1:num),:);G_output(index(1:num),:)];
    train_y_D = [train_y(index(1:num),:);test_y(index(1:num),:)];
    nn_D = nntrain(nn_D, train_data_D, train_y_D, opts);%训练 D
    %-----------训练 G-----------------------------
    for i = 1:length(nn_D.W)  %共享训练的 D 的网络参数
        nn_G.W{length(nn_G.W)-i+1} = nn_D.W{length(nn_D.W)-i+1};
    end
    %训练 G：此时假样本标签为 1，认为是真样本
    nn_G = nntrain(nn_G, test_x(index(1:num),:), test_y_rel(index(1:num),:), opts);
end
toc
for i = 1:length(nn_G_t.W)
    nn_G_t.W{i} = nn_G.W{i};
end
fin_output = nn_G_out(nn_G_t, test_x);
```

函数 nn_G_out 为：

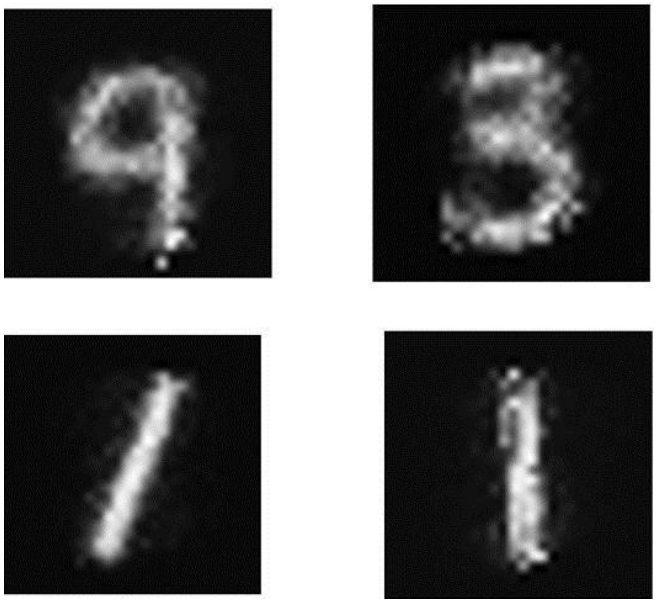
```
function output = nn_G_out(nn, x)
    nn.testing = 1;
    nn = nnff(nn, x, zeros(size(x,1), nn.size(end)));
    nn.testing = 0;
```

```
    output = nn.a{end};
end
```

看一下这个及其简单的函数，其实最值得注意的就是中间那个交替训练的过程，这里我分了三步列出来：

- 重新计算假样本（假样本每次是需要更新的，产生越来越像的样本）
- 训练 D 网络，一个二分类的神经网络；
- 训练 G 网络，一个串联起来的长网络，也是一个二分类的神经网络（不过只有假样本来训练），同时 D 部分参数在下一次的时候不能变了。

就这样调一调参数，最终输出在 fin_output 里面，多运行几次显示不同运行次数下的结果：



可以看到的是结果还是有点像模像样的。

实验总结

运行上述简单的网络我发现几个问题：

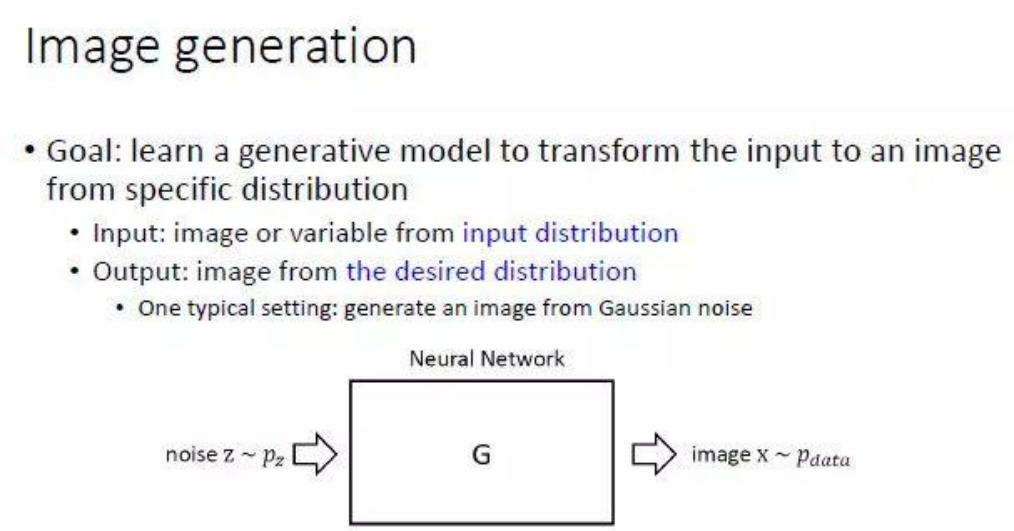
- 网络存在着不收敛问题；网络不稳定；网络难训练；读过原论文其实作者也提到过这些问题，包括 GAN 刚出来的时候，很多人也在致力于解决这些问题，当你实验自己碰到的时候，还是很有意思的。那么这些问题怎么体现的呢，举个例子，可能某一次你会发现训练的误差很小，在下一代训练时，马上又出现极限性的上升的很厉害，过几代又发现训练误差很小，震荡太严重。
- 其次网络需要调才能出像样的结果。交替迭代次数的不同结果也不一样。比如每一代训练中，D 网络训练 2 回，G 网络训练一回，结果就不一样。

- 这是简单的无条件 GAN，所以每一代训练完后，只能出现一个结果，那就是 0-9 中的某一个数。要想在一代训练中出现好几种结果，就需要使用到条件 GAN 了。

再提升一下

图像生成

这里尝试给「图像生成」一个大致定义：图像生成的目的是，学习一个生成模型，能够将来自于输入分布的一幅图像或变量转变成为一幅输出图像。这里，我们不仅要求「输入」满足一个输入分布，同样，我们还要求「输出」满足一个预期的期望分布。通过定义不同的输入分布和期望分布，就对应着不同的图像生成问题。



一开始，最标准的 GAN 的假设是，输入要服从随机噪声分布，期望分布是所有的真实图像。这个问题一开始定义得太大，所以虽然 GAN 在 2014 年就出现了，在 2014 年到 2016 年这段时间其实发展得并不快。

后来大家就去思考，输入的分布也可以不是随机分布，于是大家开始根据各种实际问题的需要来定义自己需要的输入分布和期望分布。比如，输入分布可以是来自于所有斑马的一幅图像，输出分布是所有正常马的图像，这样系统要学习的其实是这两种图像之间的映射（mapping）。

同样，如果我们输入的是一个低分辨率图像，输出的是一个高分辨率图像，那么希望系统学习到的是低分辨率和高分辨率之间的映射。去区块（deblocking），输入的是 JPEG 压缩图像，输出的是真实高清图像，我们也是希望学到两者之间的映射。人脸领域也是一样，比如我们做的超分辨和性别转换。输入是男性图像，输出是女性图像，学习二者之间的映射。

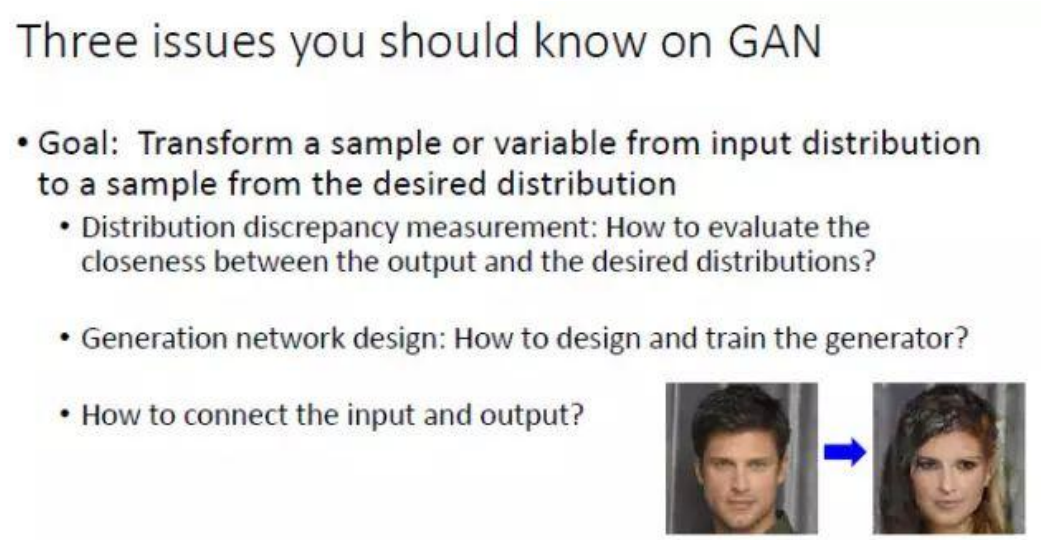
另外一个有意思的是图像文本描述的自动生成（Image captioning），输入的是图像，输出的是句子。大家以前就认为，这是一个一对一的映射，其实不是。它实际上是一对多的映射。不同的人来描述一幅图，就会产生不同的语句。所以如果用 GAN 来做这件事情，应该是很有趣的，今年有几篇投 ICCV 的文章做的就是这方面的工作。

关于 GAN 的三个问题

第一，度量复杂分布之间差异性。我们希望输出分布达到期望分布，那么我们需要找两个分布之间差异的度量方式，这是我认为在 GAN 里面需要研究的第一个关键性问题。

第二，如何设计生成器。如果我们想要学习映射，就需要一个生成器，那么就要对它的训练、可学习性进行设计。这是 GAN 里面另一个可以研究的角度。

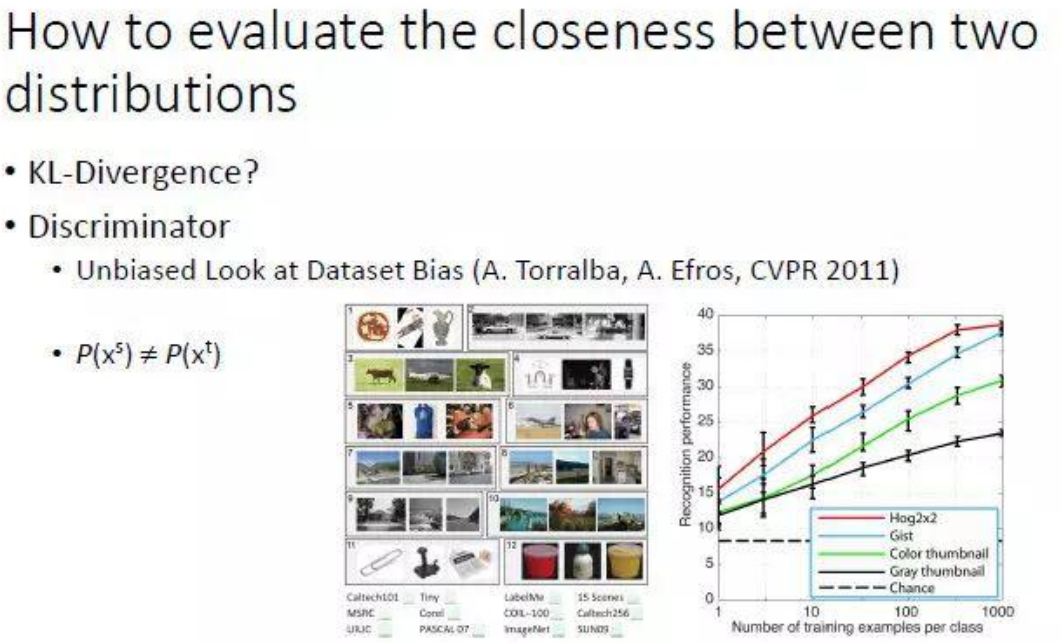
第三，连接输入和输出。如下图右边性别转换的例子，输入是一张男性图像，输出是一张女性图像。显然我们需要的并不是从输入到任意一幅女性人脸图像的映射，二是要求输出的女性图像要跟输入的男性图像尽可能像，这个转换才是有意义的。所以，这就是 GAN 里面另外一个重要的研究方向，就是如何将输入和输出连接起来。



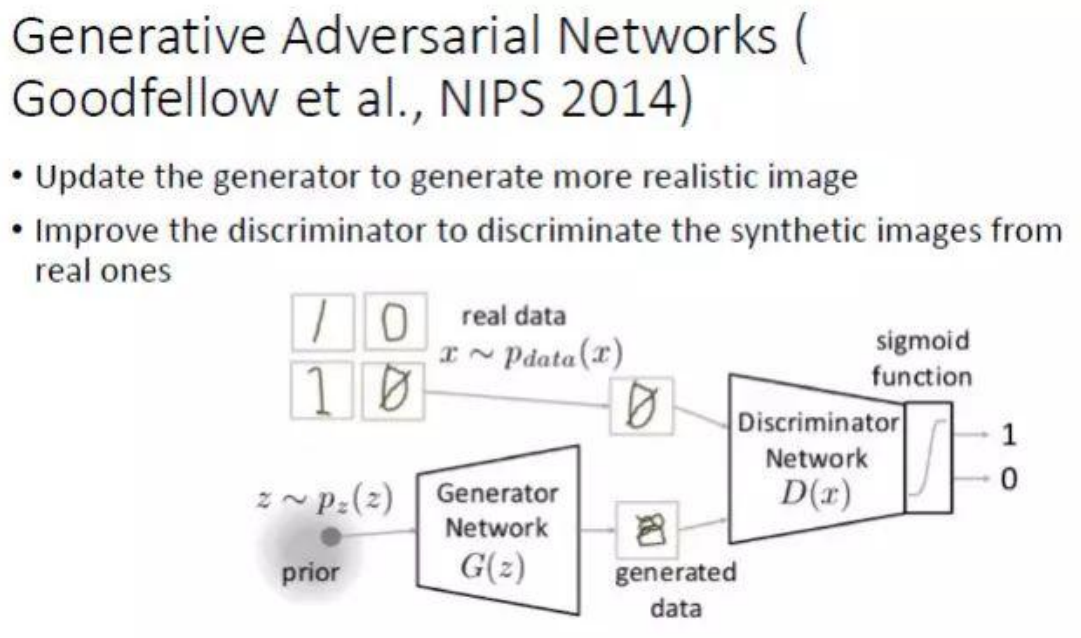
下面针对这三个问题，进行详细的讲解。

如何度量两个分布之间的差异性

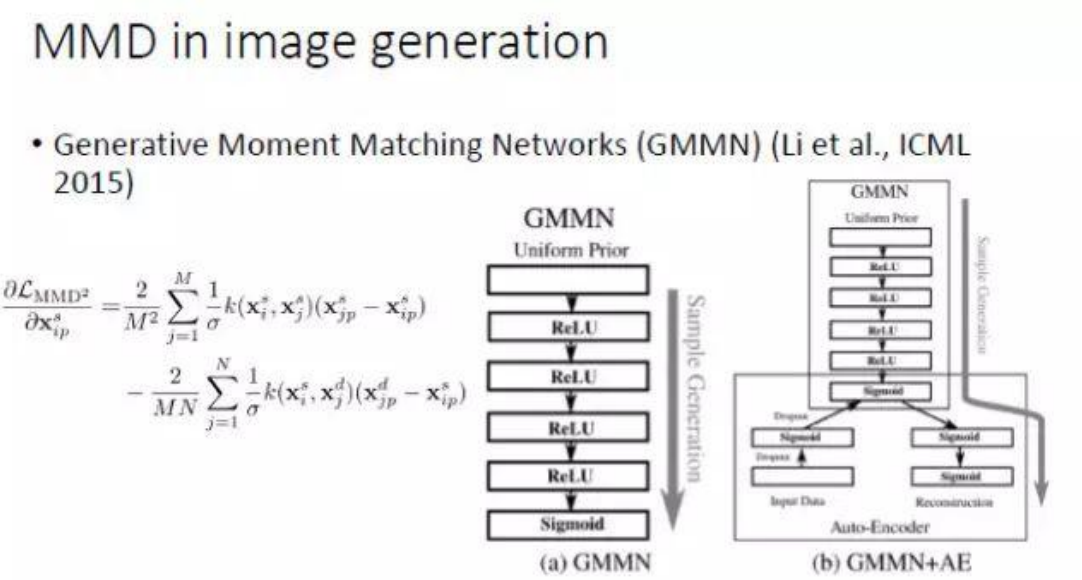
GAN 使用了一个分类器来度量输出分布和期望分布的差异性。实际上，Torralba 和 Efros 在 2011 年的时候也考虑过用一个分类器来分析两个分布之间差异，这也是当时做 domain adaption 的学者喜欢引用的一篇论文。他们设计了一个实验，给你三张图像，让你猜是来自 12 个数据集（包括 ImageNet、COCO 和 PASCAL VOC 等）中的那一个。如果是随机猜的话，显然猜中的概率是 1/12。但是人猜中的准确度往往能达到 30% 左右，说明不同数据集刻画的分布是不一致的。这里人其实可视为一个分类器，通过判断样本来自于那个数据集来分析两个分布之间差异。



虽然 NIPS 2014 年的这篇 GAN 论文没有引用 Torralba 的工作，其实它也是采用了一个判别器来度量两个分布的差异化程度。基本的过程是，固定生成器，得到一个最好的判别器，再固定判别器，学到一个最好的生成器。但是其中有一个最令人担心的问题，那就是如果我们学到的是一个很复杂的分布，就会出现模式崩溃（Mode Collapse）的问题，即无法学习复杂分布的全局，只能学习其中的一部分。

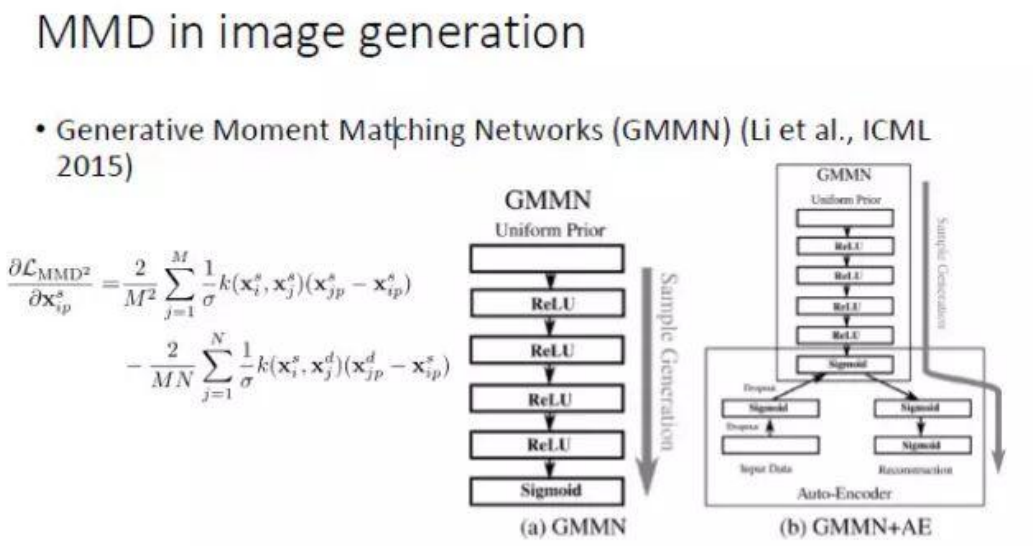


对此，最早的解决方案，是调整生成器（G）和判别器（D）的优化次序，但这也不是一个终极方案。从去年开始大家开始关注要去找到一个终极解决方案。
那之前，大家怎么去解决这个问题呢？使用的是原来机器学习里常用的方法：最大化均值差异（Maximum Mean Discrepancy，MMD）。

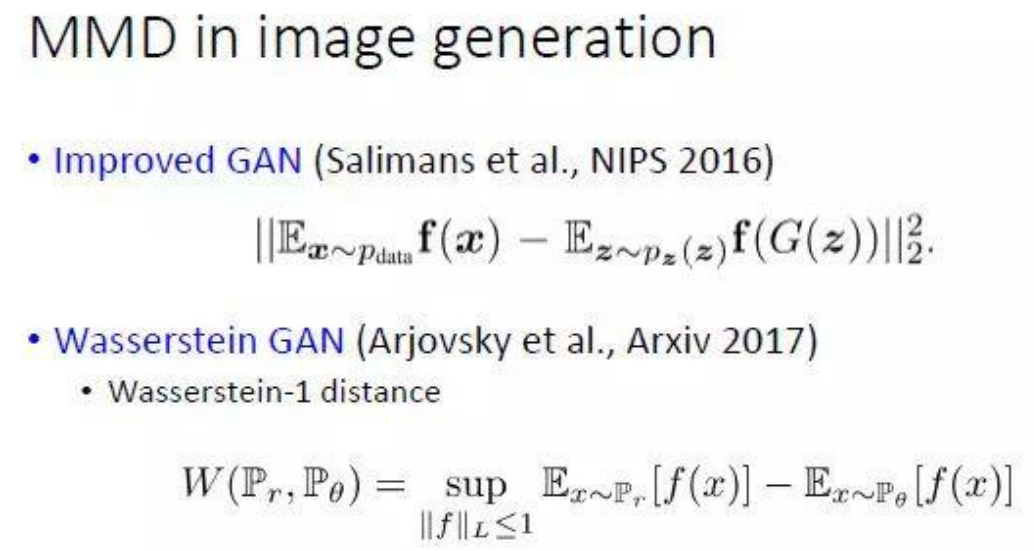


如果两个分布相同的话，那么两个分布的数学期望显然也应该相同；然而，如果两个分布的数学期望相同，并不能保证两个分布相同。因而，我们需要更好地建立「分布相同」和「期望相同」之间的连接关系。幸运的是，我们可以对来自于两个分布的变量施加同样的非线性变换。如果对于所有的非线性变换下两个分布的数学期望均相同（即：两个分布的期望的最大差别为 0），在统

计学意义上就可以保证两个分布是相同的。不幸的是，这种方法需要我们遍历所有的非线性变换，从实践的角度似乎任由一定难度。一开始，在机器学习领域，大家倾向于用线性 kernel 或 Gaussian RBF kernel 来进行非线性变换，后来开始采用 multi-kernel。从去年开始，大家开始用 CNN 来近似所有的非线性变换，在 MMD 框架下进行图像生成。首先，固定生成器并最大化 MMD，然后固定判别器里 MMD 的 f，然后通过最小化 MMD 来更新生成器。
最常用的一种方法，就是拿 MMD 来代替判别器，去学习一个 CNN，这是 ICML 2015 的一篇文章中尝试的方法，我们自己也在这个基础上做了一些工作。

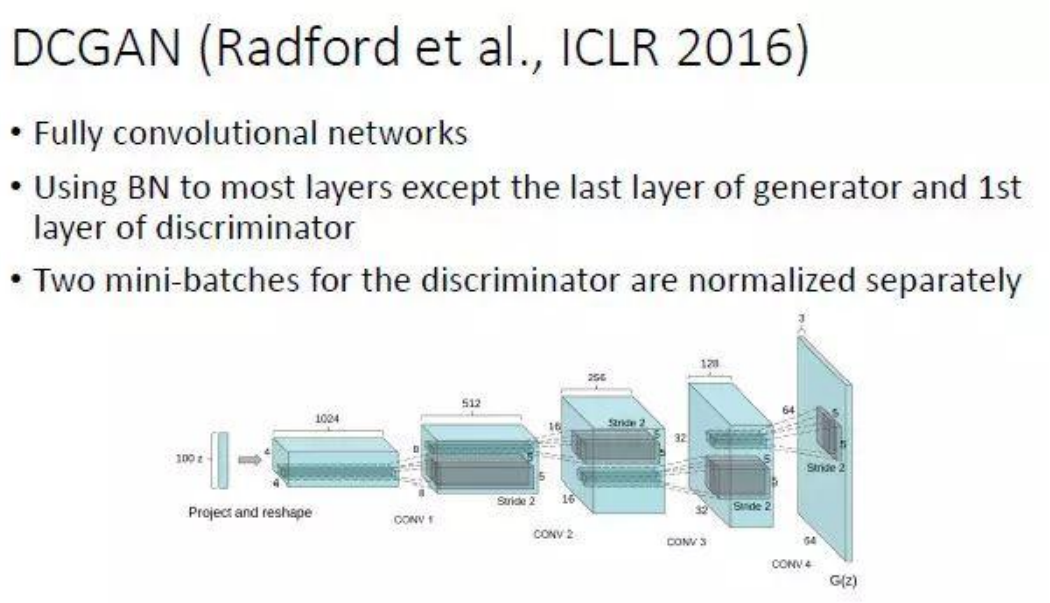


但实际上，如果直接拿 MMD 去替换生成器，虽然有一定效果，但不是特别成功。所以，从 NIPS 2016 开始，就出现了一个 Improved GAN，这个工作虽然没有引用 MMD 的论文，但实际上在更新判别器的同时也最小化了 MMD。等到了 Wasserstein GAN 的时候，它就明确解释了与 MMD 之间的联系，虽然论文里写的是一个「减」的关系，但我们看它的代码，它也是要加上一个范数的，因为只是让两个分布的期望最大化或最小化都不能保证分布的差异化程度最小。
然后，最近 ICLR 2017 的一篇论文也明确指出要用 MMD 来作为 GAN 网络的停止条件和学习效果的评价手段。

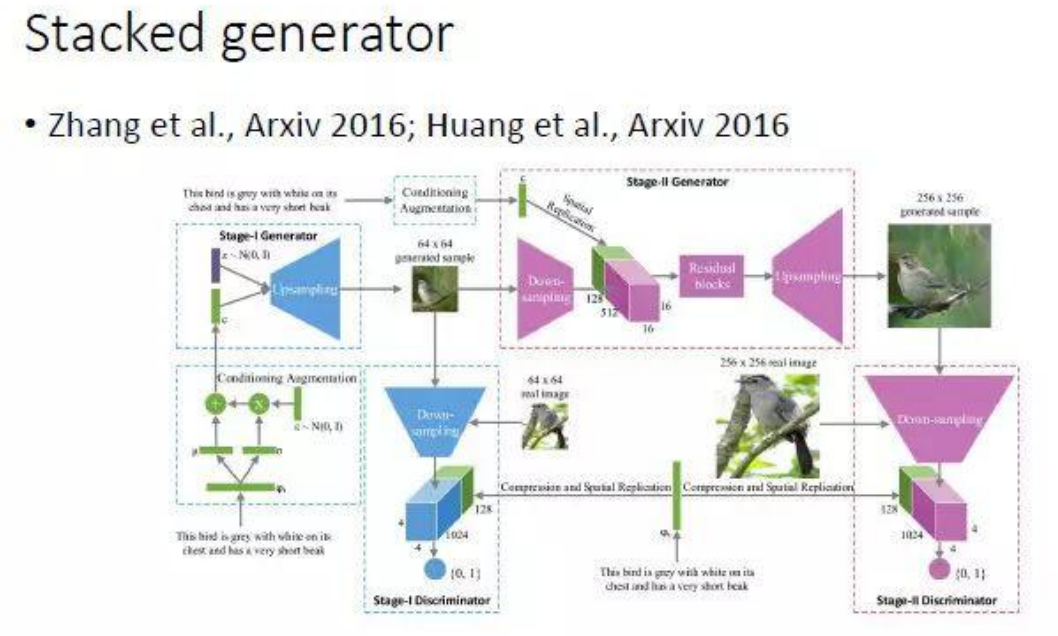


如何设计一个生成器

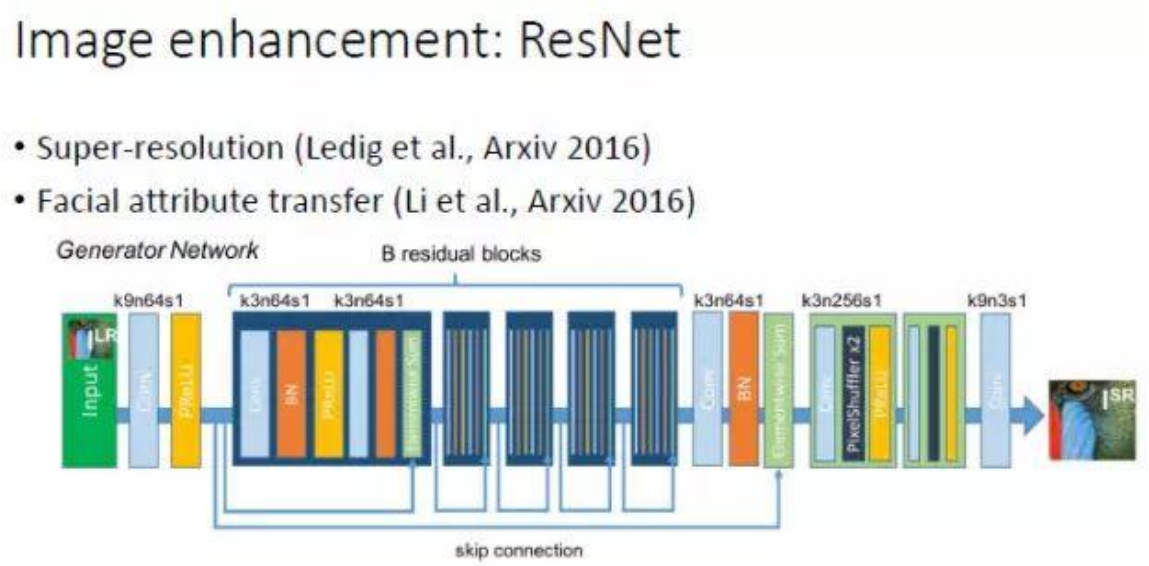
这个部分相对来说就比较容易一些。早期的时候，GAN 的一个最大的进步就是 DCGAN，用于图像生成时，比较合适的选择就是用全卷积网络加上 batch normalizaiton。



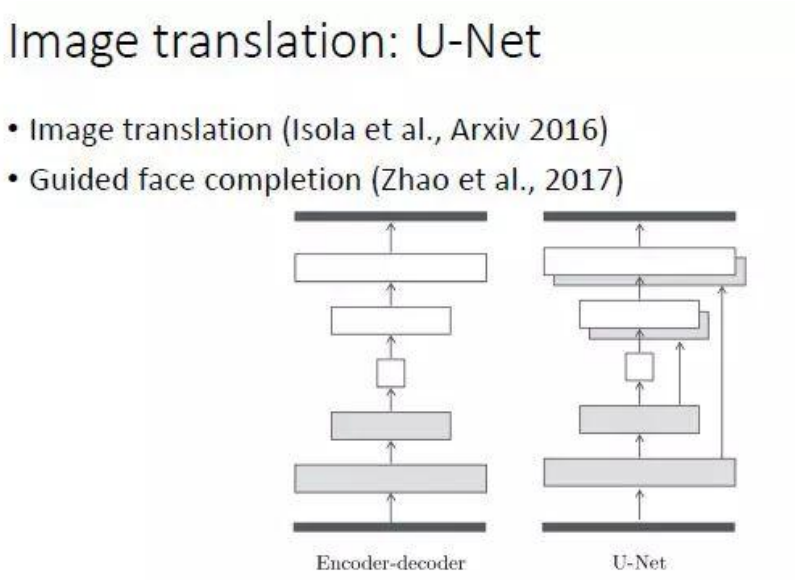
对于复杂的图像生成，可以使用分阶段的方式。比如，第一步可以生成小图，然后由小图生成大图。沿着这个方向，香港中文大学王晓刚老师和康奈尔大学 John Hopcroft 都做了一些工作。



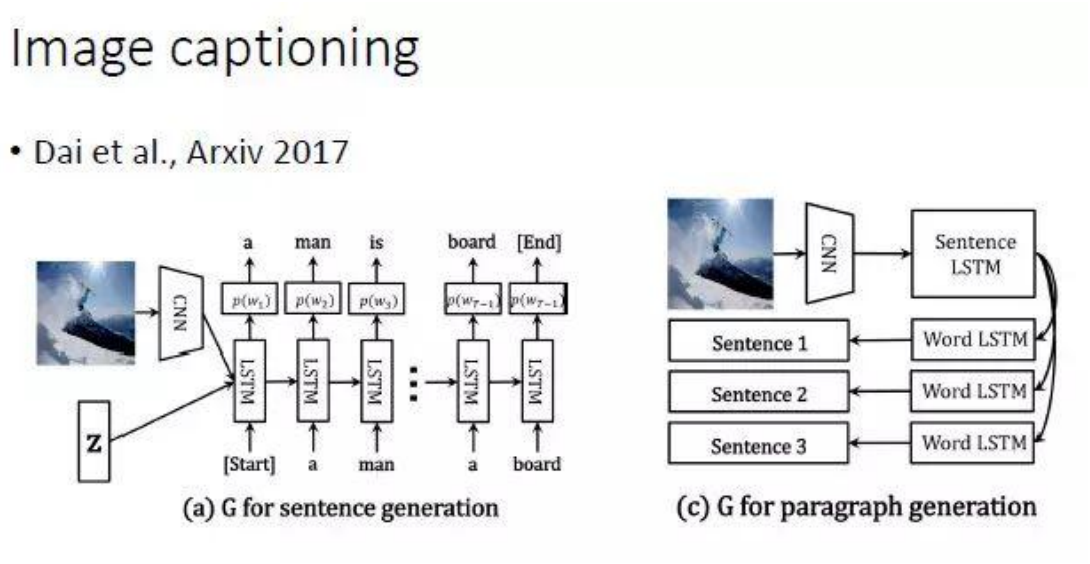
对于图像增强（image enhancement）相关的一些任务，包括超分辨率和人脸属性转换（Face Attribute Transfer），目前在有监督时表现最好网络是 ResNet ，所以我们在这些任务中实用 GAN 时一般也会采用 ResNet 结构。



同样，对于图像转换（image translation），基本上用的是 U-Net 结构。我们在做基于引导图像的人脸填充（guided face completion）时也采用了 U-Net 结构。

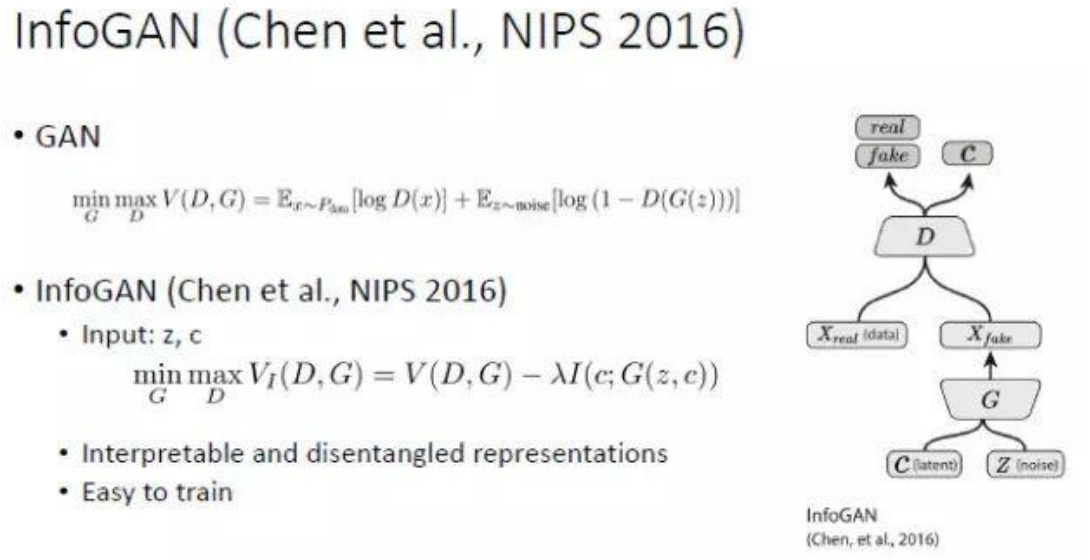


对于图像文本描述的自动生成，显然应该采用 CNN+RNN 这样的网络结构。总而言之，一个比较好的建议就是根据任务的特点和前任的经验来设计生成器网络。

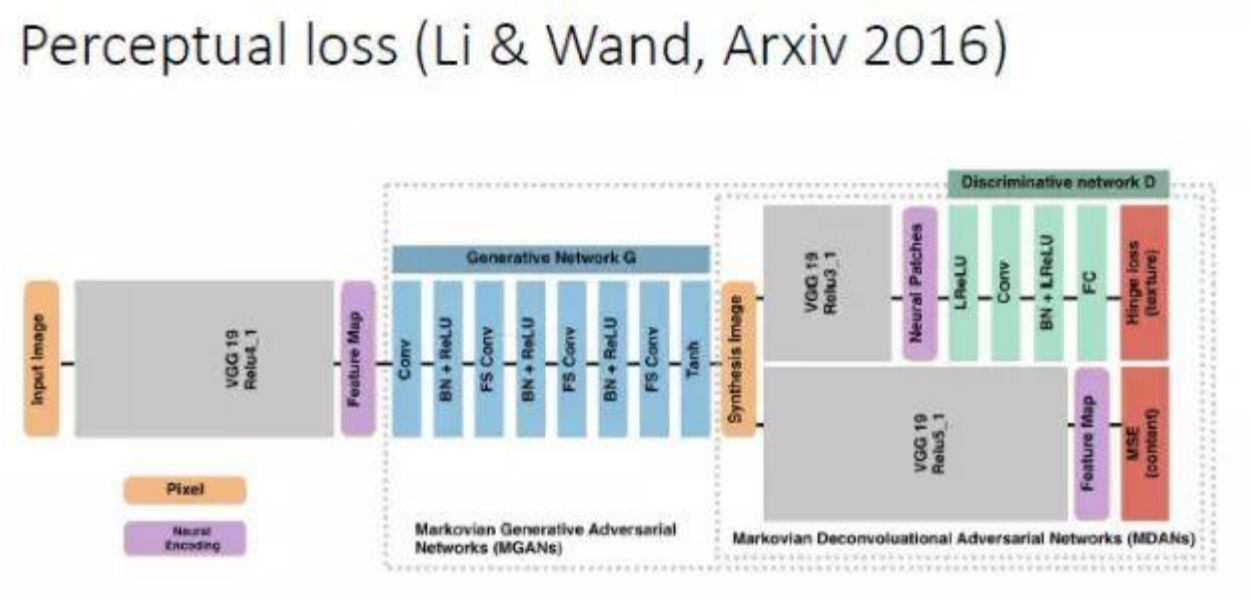


如何连接输入和输出

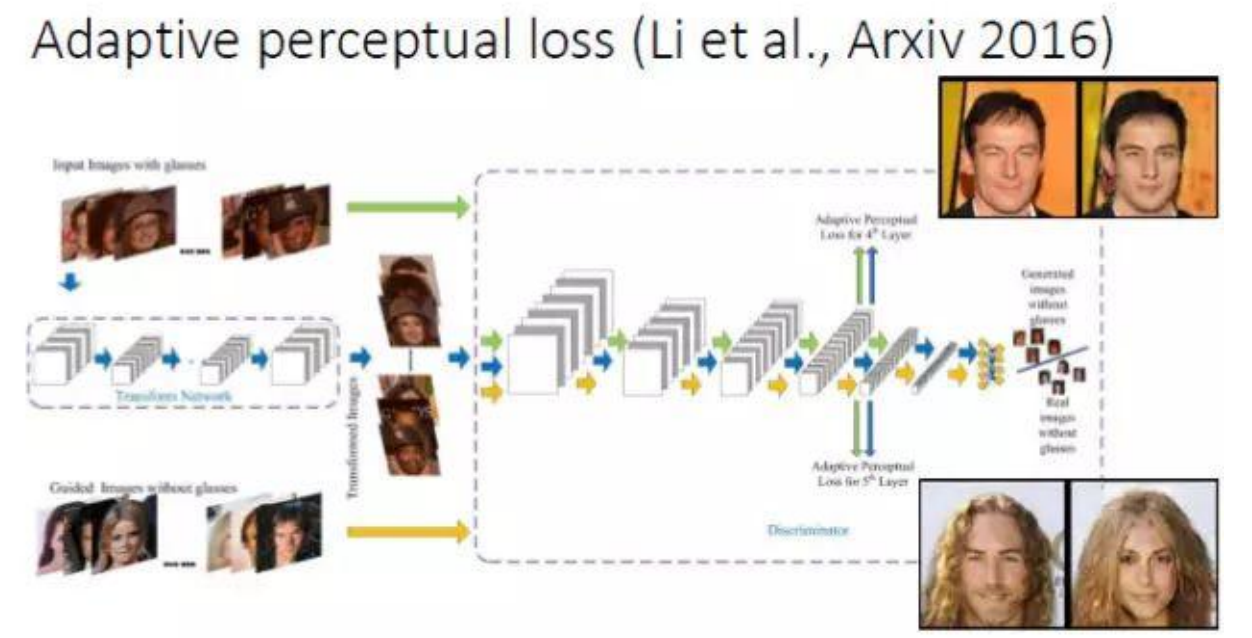
如何通过连接输入和输出的方式来改善 GAN 的可学习性，这个问题是从 NIPS 2016 开始得到了较多的关注，同时这也是我自己非常感兴趣的一个方向。比较早的一个工作就是 InfoGAN，其特点就是输入包括两个部分：C（隐变量）和 Z（噪声）。InfoGAN 生成图像之后，不仅要求生成图像和真实图像难以区分，还要求能够从生成图像中预测出 C，这样就为输入和输出建立起了一个联系。



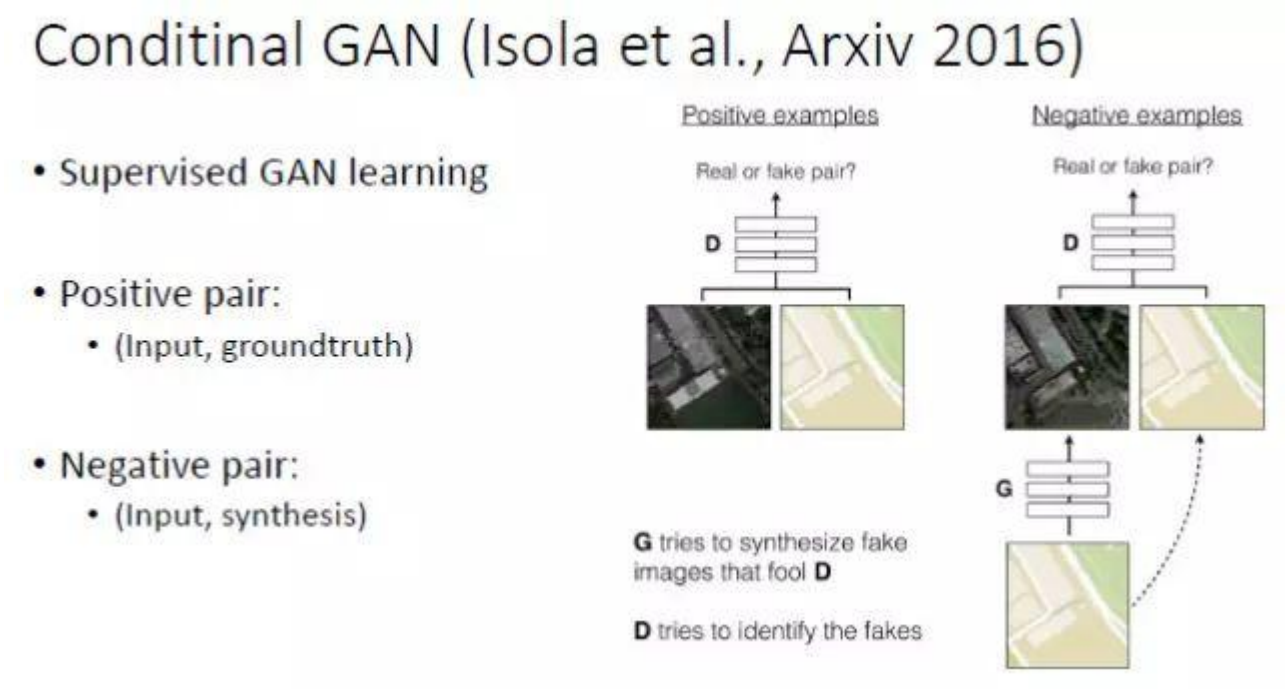
另外针对一些任务，比如超分辨，可以用 Perceptual loss 的方式来建立输入和输出的联系。



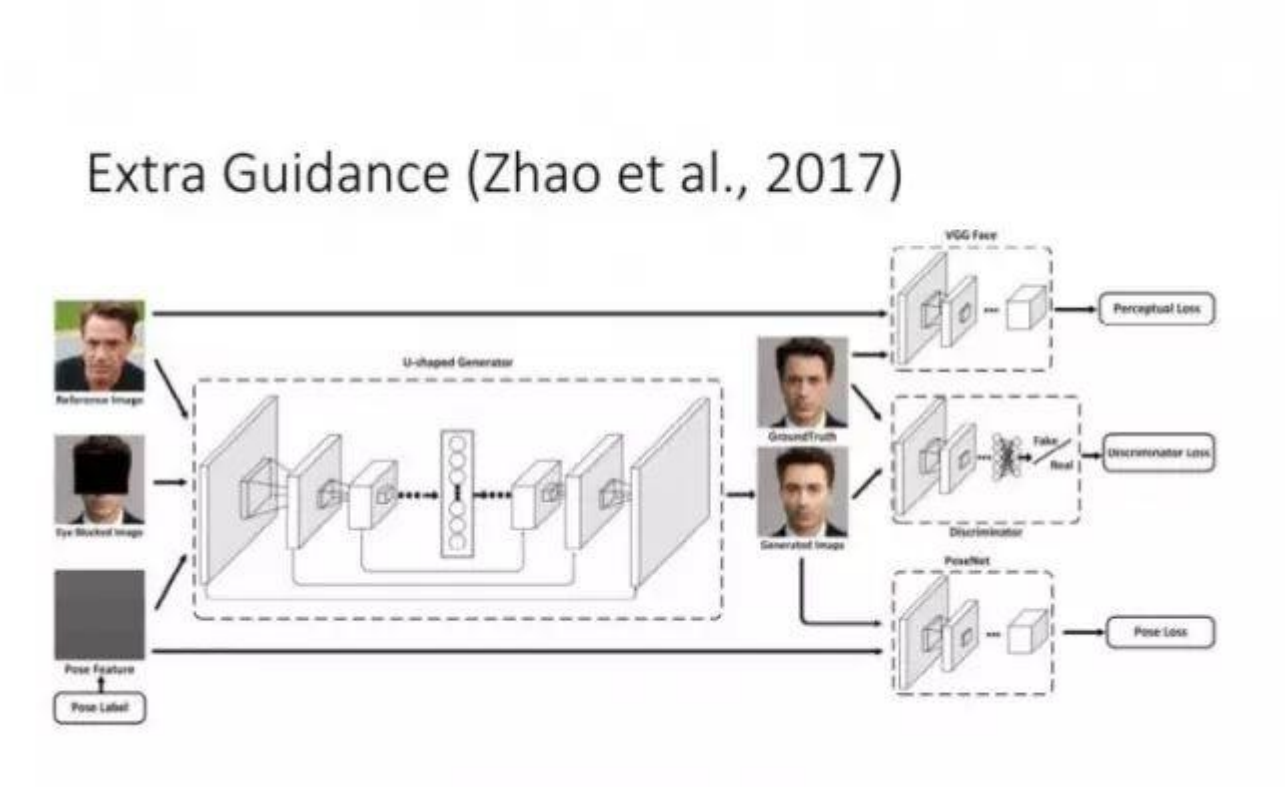
我们在做人脸属性转换时发现，现有的 Perceptual loss 往往是定义在一个现有的网络基础上的，我们就想能不能把 Perceptual loss 网络和判别器结合起来，所以就提出了一个 Adaptive perceptual loss。结果表明 Adaptive perceptual loss 能够具有更好的自适应性，能够更好地建立输入和输出的联系和显著改善生成图片的视觉效果。



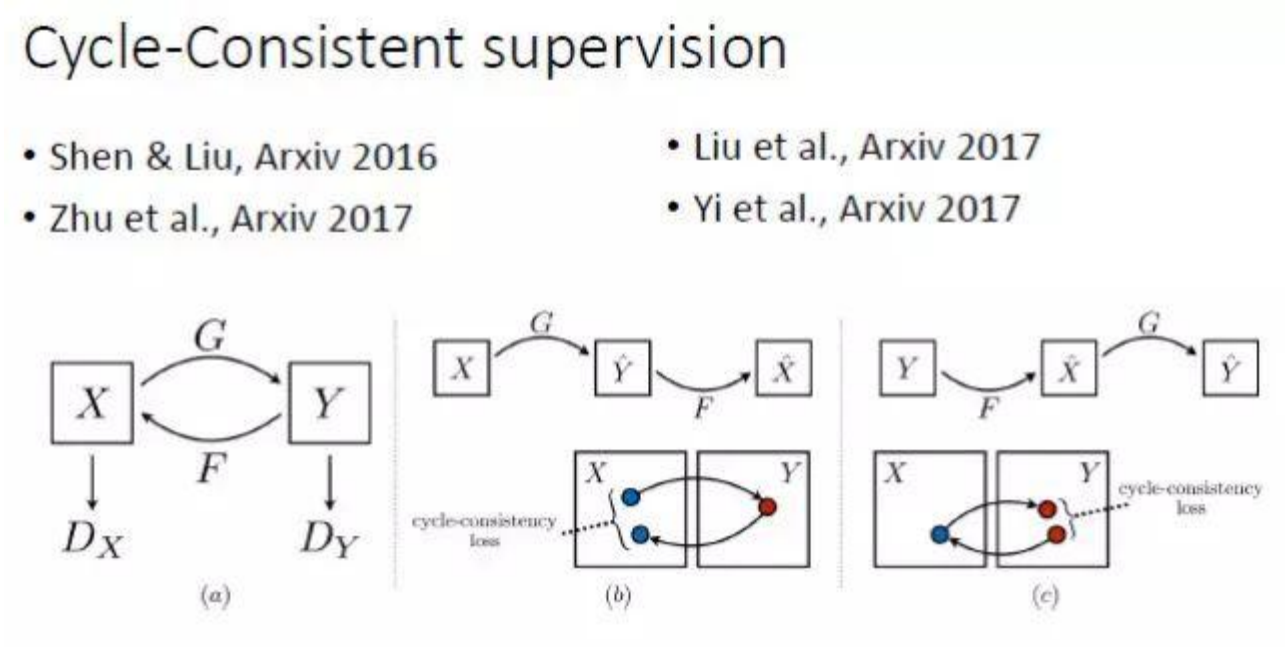
当输入和输出都是已知时（比如图像超分辨和图像转换），要用什么方式来连接输入和输出呢？以前是用 Perceptual loss 来连，现在更好的方式是用 Conditional GAN。假设有一个 Positive Pair（输入和 groundtruth 图像）和 Negative Pair（输入和生成图像），那么判别器就不是在两幅图像之间做判别，而是在两个「Pair」之间做判别。这样的话，输入就很自然地引入到了判别器中。



在此基础上，我们还考虑了当有一些额外的 Guidance 时，如何来更好地建立输入和输出的联系。



上面提到，在有监督的情况下 Conditional GAN 是一个比较好的选择。但如果在 unpair 的情况下做图像转换，要如何建立输入和输出的联系？谭平老师他们组和 Efros 组今年就做了这方面的工作，其实去年投 CVPR2017 的一篇论文也做了类似的工作。我们知道，由于是 unpair 的，原则上训练阶段输入和输出不能直接建立联系。这时他们采用的是一种 Cycle-Consistent 的方式。从 X 可以预测和生成 Y，再从 Y 重新生成 X'，那么由 Y 生成的 X' 就能跟输入的 X 建立联系。这样的话，我们实际上相当于隐式地建立了从 X 到 Y 的联系。



总结

现在的 GAN 已经到了五花八门的时候了，各种 GAN 应用也很多，理解底层原理再慢慢往上层扩展。GAN 还是一个很厉害的东西，它使得现有问题从有监督学习慢慢过渡到无监督学习，而无监督学习才是自然界中普遍存在的，因为很多时候没有办法拿到监督信息的。要不 Yann Lecun 赞叹 GAN 是机器学习近十年来最有意思的想法。

原文地址：

https://blog.csdn.net/on2way/article/details/72773771