生成模型---流模型（Flow-based Model）

目 录

前言

· **Flow-based 模型的不同之处**

从去年 GLOW 提出之后，我就一直对基于流（flow）的生成模型是如何实现的充满好奇，但一直没有彻底弄明白，直到最近观看了李宏毅老师的教程之后，很多细节都讲解地比较清楚，就想好好写篇笔记来梳理一下流模型的运作原理。

首先来简单介绍一下流模型，它是一种比较独特的生成模型——它选择直接直面生成模型的概率计算，也就是把分布转换的积分式（$p_G(x)=\int_z p(x|z)p(z)dz$）给硬算出来。要知道现阶段其他较火的生成模型，要么采用优化上界或采用对抗训练的方式去避开概率计算，从而寻找近似逼近真实分布的方法，但是流模型选择了一条硬路（主要是通过变换 Jacobian 行列式）来求解，在后文会详细介绍。

流模型有一个非常与众不同的特点是，它的转换通常是可逆的。也就是说，流模型不仅能找到从 A 分布变化到 B 分布的网络通路，并且该通路也能让 B 变化到 A，简言之流模型找到的是一条 A、B 分布间的双工通路。当然，这样的可逆性是具有代价的——A、B 的数据维度必须是一致的。

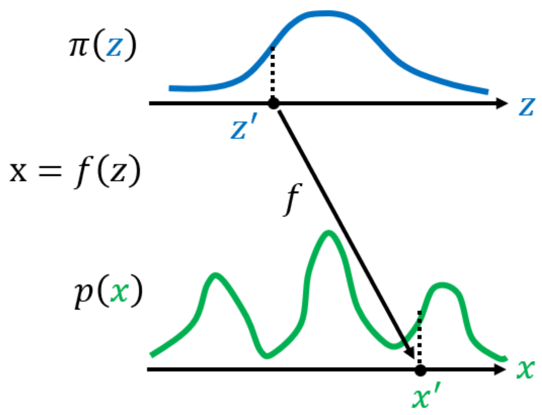
A、B 分布间的转换并不是轻易能做到的，流模型为实现这一点经历了三个步骤：最初的 NICE 实现了从 A 分布到高斯分布的可逆求解；后来 RealNVP 实现了从 A 分布到条件非高斯分布的可逆求解；而最新的 GLOW，实现了从 A 分布到 B 分布的可逆求解，其中 B 分布可以是与 A 分布同样复杂的分布，这意味着给定两堆图片，GLOW 能够实现这两堆图片间的任意转换。

下面就是流模型学习笔记的正文，尽可能较简明地讲解清楚流模型的运行机制。

---

1. Flow-based Model 的建模思维

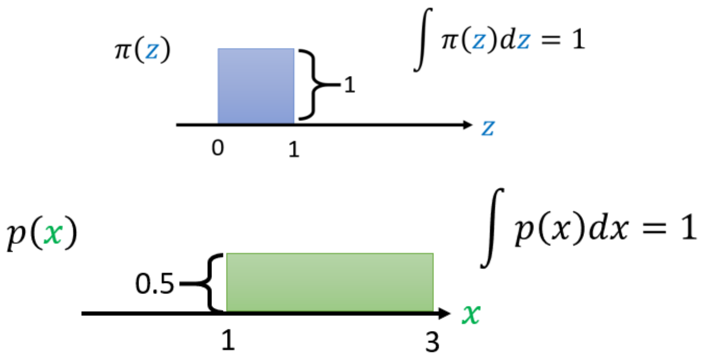
首先来回顾一下生成模型要解决的问题：



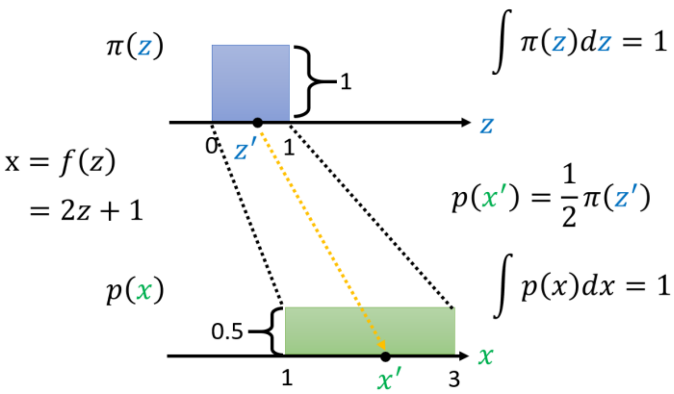
如上图所示，给定两组数据 z 和 x，其中 z 服从已知的简单先验分布π(z)（通常是高斯分布），x 服从复杂的分布 p(x)（即训练数据代表的分布），现在我们想要找到一个变换函数 f，它能建立一种 z 到 x 的映射$f: z \to x$，使得每对于π(z)中的一个采样点z'，都能在 p(x)中有一个（新）样本点x'与之对应。

如果这个变换函数能找到的话，那么我们就实现了一个生成模型的构造。因为，p(x)中的每一个样本点都代表一张具体的图片，如果我们希望机器画出新图片的话，只需要从π(z)中随机采样一个点，然后通过$f: z \to x$，得到新样本点 x，也就是对应的生成的具体图片。

所以，接下来的关键在于，这个变换函数 f 如何找呢？我们先来看一个最简单的例子。

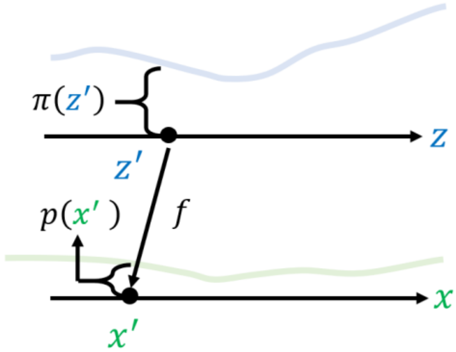


如上图所示，假设 z 和 x 都是一维分布，其中 z 满足简单的均匀分布：$\pi(z) = 1\ (z \in [0,1])$，x 也满足简单均匀分布：$p(x) = 0.5\ (x \in [1,3])$。

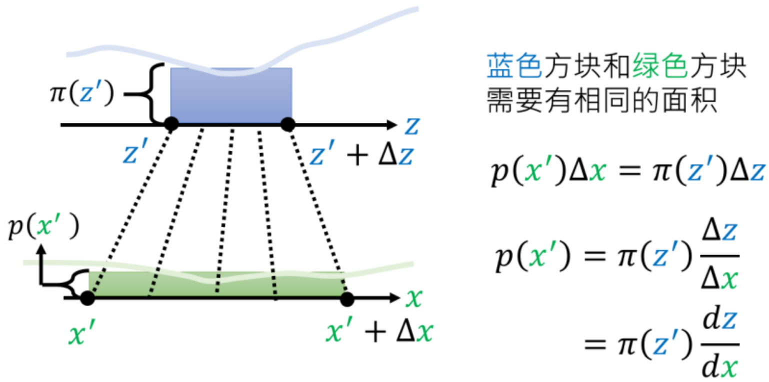


那么构建 z 与 x 之间的变换关系只需要构造一个线性函数即可：x=f(z)=2z+1。

下面再考虑非均匀分布的更复杂的情况:



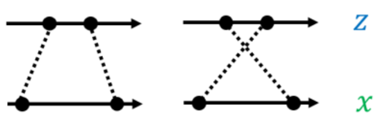
如上图所示，π(z)与 p(x)都是较为复杂的分布，为了实现二者的转化，我们可以考虑在很短的间隔上将二者视为简单均匀分布，然后应用前边方法计算小段上的f_Δ，最后将每个小段变换累加起来（每个小段实际对应一个采样样本）就得到最终的完整变换式 f。



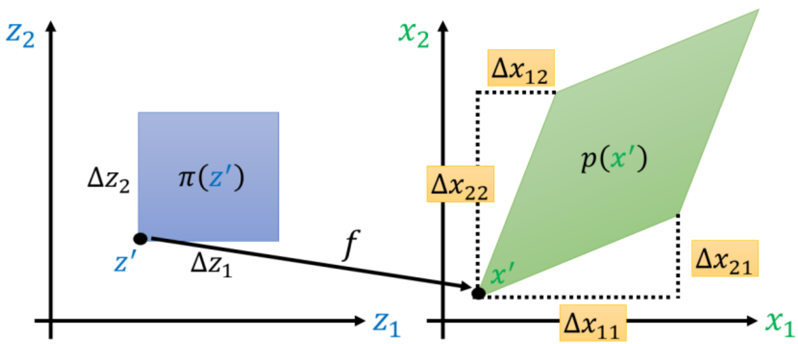
如上图所示，假设在[z′,z′+Δz]上π(z)近似服从均匀分布，在[x′,x′+Δx]上 p(x)也近似服从均匀分布，于是有$p(x')\Delta x$=$\pi(z')\Delta z$（因为变换前后的面积/即采样概率是一致的），当Δx 与Δz极小时，有:

$$p(x') = \pi(z')\frac{dz}{dx}$$

又考虑到$\frac{dz}{dx}$有可能是负值（如下图所示），而$p(x')$与$\pi(z')$都为非负，所以$p(x')$与$\pi(z')$的实际关系为：$p(x') = \pi(z')\left|\frac{dz}{dx}\right|$。



下面进一步地做推广，我们考虑 z 与 x 都是二维分布的情形。



如上图所示，z 与 x 都是二维分布，左图中浅蓝色区域表示初始点z'在z_1方向上移动Δz_1，在z_2方向上移动Δz_2所形成的区域，这一区域通过$f: z \rightarrow x$映射，形成右图所示 x 域上的浅绿色菱形区域。其中，二维分布$\pi(z)$与 $p(x)$均服从简单均匀分布，其高度在图中未画出（垂直纸面向外）。

因为蓝色区域与绿色区域具有相同的体积，所以有：

$$p(x')\left|det\begin{bmatrix}\Delta x_{11} & \Delta x_{21}\\ \Delta x_{12} & \Delta x_{22}\end{bmatrix}\right| = \pi(z')\Delta z_1 \Delta z_2$$

其中$det\begin{bmatrix}\Delta x_{11} & \Delta x_{21}\\ \Delta x_{12} & \Delta x_{22}\end{bmatrix}$代表行列式计算，它的计算结果等于上图中浅绿色区域的面积（行列式的定义）。下面我们将$\Delta z_1 \Delta z_2$移至左侧，得到：

$$p(x')\left|\frac{1}{\Delta z_1 \Delta z_2} det\begin{bmatrix}\Delta x_{11} & \Delta x_{21}\\ \Delta x_{12} & \Delta x_{22}\end{bmatrix}\right| = \pi(z')$$

即：

$$p(x')\left|det\begin{bmatrix}\Delta x_{11}/\Delta z_1 & \Delta x_{21}/\Delta z_1\\ \Delta x_{12}/\Delta z_2 & \Delta x_{22}/\Delta z_2\end{bmatrix}\right| = \pi(z')$$

在Δz_1，Δz_2很小时，有：

$$p(x')\left|det\begin{bmatrix}\partial x_1/\partial z_1 & \partial x_2/\partial z_1\\ \partial x_1/\partial z_2 & \partial x_2/\partial z_2\end{bmatrix}\right| = \pi(z')$$

即：

$$p(x')\left|det(J_f)\right| = \pi(z')$$

其中J_f表示 f 运算的雅各比行列式，根据雅各比行列式的逆运算，我们得到：

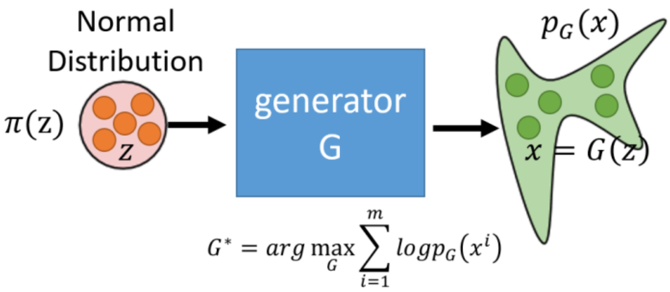
$$p(x') = \pi(z')\left|det(J_{f^{-1}})\right|$$

其中f^{-1}代表从 x 变换为 z 的变换式，即：$z = f^{-1}(x)$。

至此，我们得到了一个比较重要的结论：如果 z 与 x 分别满足两种分布，并且 z 通过函数 f 能够转变为 x，那么 z 与 x 中的任意一组对应采样点z'与x'之间的关系为：

$$\begin{cases}\pi(z') = p(x')\left|det(J_f)\right| \\ p(x') = \pi(z')\left|det(J_{f^{-1}})\right|\end{cases}$$

那么基于这一结论，再带回到生成模型要解决的问题当中，我们就得到了 Flow-based Model（流模型）的初步建模思维。



如上图所示，为了实现$z \sim \pi(z)$到$x = G(z) \sim p_G(x)$间的转化，待求解的生成器 G 的表达式为：

$$G^* = arg \max_G \sum_{i=1}^{m} log p_G(x^i)$$

基于前面推导，我们有$p_G(x)$中的样本点与$\pi(z)$中的样本点间的关系为：

$$p_G(x^i) = \pi(z^i)(\left|det(J_G)\right|)^{-1}$$

其中$z^i = G^{-1}(x^i)$。

所以，如果G^*的目标式能够通过上述关系式求解出来，那么我们就实现了一个完整的生成模型的求解。Flow-based Model 就是基于这一思维进行理论推导和模型构建，下面将会详细解释 Flow-based Model 的求解过程。

2. Flow-based Model 的理论推导&架构设计

我们关注一下上一章中引出的式子：

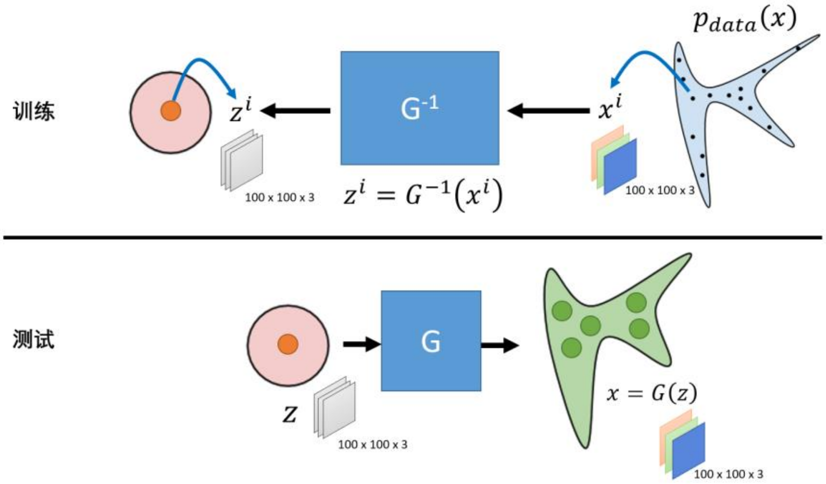
$$p_G(x^i) = \pi(z^i)(|det(J_G)|)^{-1}，\ z^i = G^{-1}(x^i)$$

将其取 log，得到：

$$log_{p_G(x^i)} = log_{\pi\left(G^{-1}(x^i)\right)} + log_{|det(J_{G^{-1}})|}$$

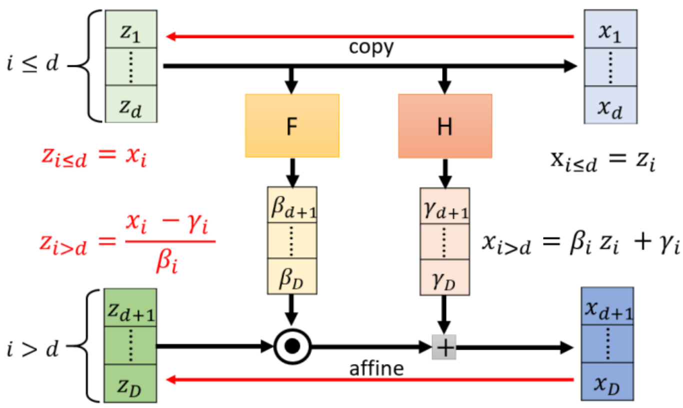
现在，如果想直接求解这个式子有两方面的困难。第一个困难是，$det(J_{G^{-1}})$是不好计算的——由于G^{-1}的 Jacobian 矩阵一般维度不低（譬如 256*256 矩阵），其行列式的计算量是异常巨大的，所以在实际计算中，我们必须对G^{-1}的 Jacobian 行列式做一定优化，使其能够在计算上变得简洁高效。第二个困难是，表达式中出现了G^{-1}，这意味着我们要知道G^{-1}长什么样子，而我们的目标是求 G，所以这需要巧妙地设计 G 的结构使得G^{-1}也是好计算的。

下面我们来逐步设计 G 的结构，首先从最基本的架构开始构思。考虑到G^{-1}必须是存在的且能被算出，这意味着 G 的输入和输出的维度必须是一致的并且 G 的行列式不能为 0。然后，既然G^{-1}可以计算出来，而$log_{p_G(x^i)}$的目标表达式只与G^{-1}有关，所以在实际训练中我们可以训练G^{-1}对应的网络，然后想办法算出 G 来并且在测试时改用 G 做图像生成。



如上图所示，在训练时我们从真实分布$p_{data}(x)$中采样出x^i，然后去训练G^{-1}，使得通过G^{-1}生成的$z^i = G^{-1}(x^i)$满足特定的先验分布；接下来在测试时，我们从 z 中采样出一个点z^j，然后通过 G 生成的样本$x^j = G(z^j)$就是新的生成图像。

接下来开始具体考虑 G 的内部设计，为了让G^{-1}可以计算并且 G 的 Jacobian 行列式也易于计算，Flow-based Model 采用了一种称为耦合层（Coupling Layer）的设计来实现。



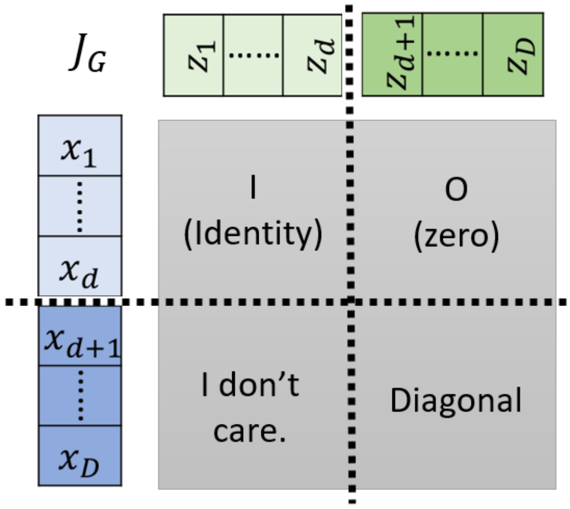
如上图所示，z 和 x 都会被拆分成两个部分，分别是前 1~d 维和后 d+1~D 维。从 z 变化为 x 的计算式为：z 的 1~d 维直接复制（copy）给 x 的 1~d 维；z 的 d+1~D 维分别通过 F 和 H 两个函数变换为$\beta_{d+1,\ldots,D}$和$\gamma_{d+1,\ldots,D}$，然后通过$x_i = \beta_i z_i + \gamma_i\,(\mathrm{i = d+1,\ldots,D})$的仿射计算（affine）传递给 x。综上，由 z 传给 x 的计算式可以写为：

$$\begin{cases} x_i = z_i, i \le d \\ x_i = \beta_i z_i + \gamma_i, i > d \end{cases}$$

其逆运算的计算式，即由 x 传给 z 的计算式，可以非常方便地推导出来为：

$$\begin{cases} z_i = x_i, i \le d \\ z_i = \dfrac{x_i - \gamma_i}{\beta_i}, i > d \end{cases}$$

上面我们说明了，这样设计的耦合层能快速计算出G^{-1}，下面我们来说明，其在 G 的 Jacobian 行列式的计算上也是非常简便。

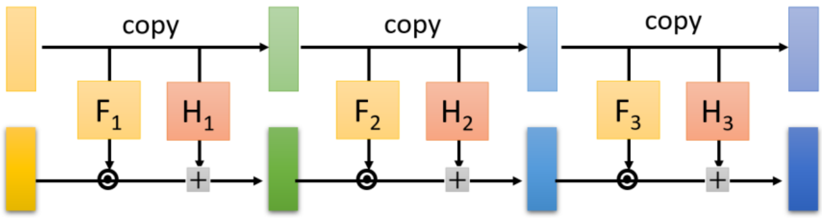


上图展示了 G 的 Jacobian 行列式的计算矩阵。首先由于$z_{1\ldots d}$直接传递给$x_{1\ldots d}$所以 Jacobian 矩阵的左上角区域是单位矩阵 I，然后$x_{1\ldots d}$完全不受$z_{d+1\ldots D}$影响，所以 Jacobian 矩阵的右上角区域是零矩阵 O，这导致 Jacobian 矩阵的左下角区域的值对 Jacobian 矩阵行列式的计算没有影响，也就无需考虑。最后我们关注 Jacobian 矩阵的右下角区域，由于$x_i = \beta_i z_i + \gamma_i\,(i > d)$，所以只有在$i = j$的情况下$\frac{\partial x_i}{\partial z_j} \ne 0$，而在$\mathrm{i \ne j}$处$\frac{\partial x_i}{\partial z_j} = 0$，所以 Jacobian 矩阵的右下角区域是一个对角矩阵。

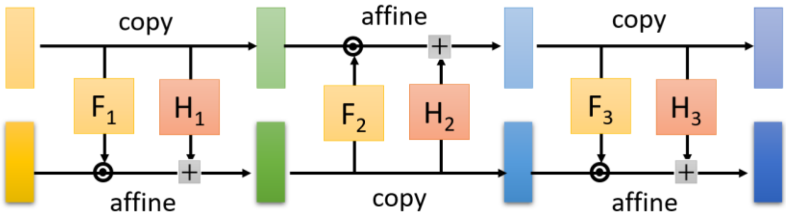
最终，该 G 的 Jacobian 的行列式计算式就表示为：

$$det(J_G) = \frac{\partial x_{d+1}}{\partial z_{d+1}} \frac{\partial x_{d+2}}{\partial z_{d+2}} \cdots \frac{\partial x_D}{\partial z_D} = \beta_{d+1}\beta_{d+2} \cdots \beta_D$$

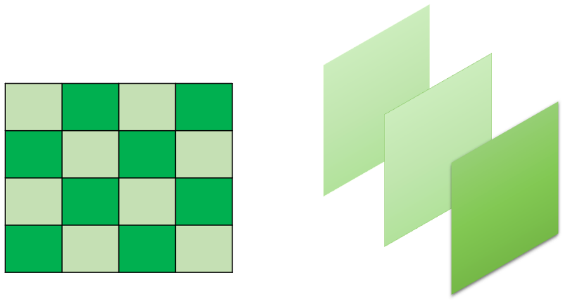
这确实是一个易于计算的简单表达式。接下来可以考虑，由于上述措施对 G 做了诸多限制，导致 G 的变换能力有限，所以我们可以堆叠多个 G，去增强模型的变换拟合能力。



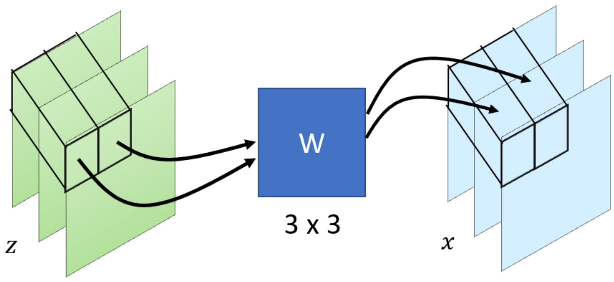
如上图所示，我们将多个耦合层堆叠在一起，从而形成一个更完整的生成器。但是这样会有一个新问题，就是最终生成数据的前 d 维与初始数据的前 d 维是一致的，这会导致生成数据中总有一片区域看起来像是固定的图样（实际上它代表着来自初始高斯噪音的一个部分），我们可以通过将复制模块（copy）与仿射模块（affine）交换顺序的方式去解决这一问题。



如上图所示，通过将某些耦合层的 copy 与 affine 模块进行位置上的互换，使得每一部分数据都能走向 copy->affine->copy->affine 的交替变换通道，这样最终的生成图像就不会包含完全 copy 自初始图像的部分。值得说明的是，在图像生成当中，这种 copy 与 affine 模块互换的方式有很多种，下面举两个例子来说明：



上图展示了两种按照不同的数据划分方式做 copy 与 affine 的交替变换。左图代表的是在像素维度上做划分，即将横纵坐标之和为偶数的划分为一类，和为奇数的划分为另外一类，然后两类分别交替做 copy 和 affine 变换（两两交替）；右图代表的是在通道维度上做划分，通常图像会有三通道，那么在每一次耦合变换中按顺序选择一个通道做 copy，其他通道做 affine（三个轮换交替），从而最终变换出我们需要的生成图形出来。



更进一步地，如何进行 copy 和 affine 的变换能够让生成模型学习地更好，这是一个可以由机器来学习的部分，所以我们引入 W 矩阵，帮我们决定按什么样的顺序做 copy 和 affine 变换，这种方法叫做 1×1 convolution（被用于知名的 GLOW 当中）。1×1 convolution 只需要让机器决定在每次仿射计算前对图片哪些区域实行像素对调，而保持 copy 和 affine 模块的顺序不变，这实际上和对调 copy 和 affine 模块顺序产生的效果是一致的。

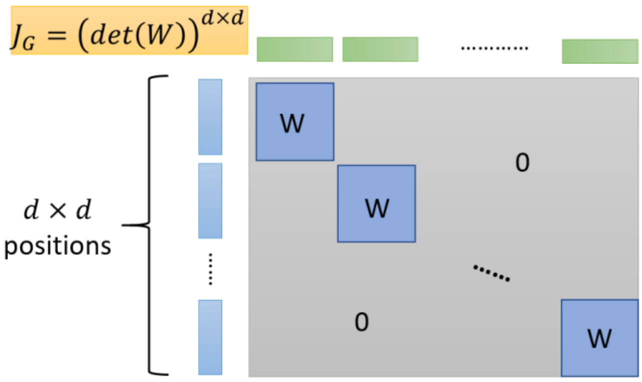
$$\begin{bmatrix}3\\1\\2\end{bmatrix}=\begin{bmatrix}0&0&1\\1&0&0\\0&1&0\end{bmatrix}\begin{bmatrix}1\\2\\3\end{bmatrix}$$

这种对调的原理非常简单，如上图所示举例，假设我们需要将(3,1,2)向量替换成(1,2,3)向量，只需要将 w 矩阵定义为图中所示矩阵即可。下面我们看一下，将 w 引入 flow 模型之后，对于原始的 Jacobian 行列式的计算是否会有影响。

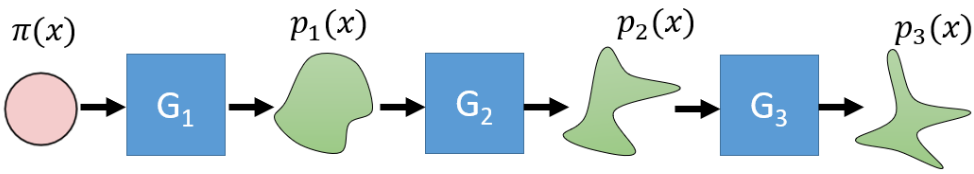
对于每一个 3*3 维划分上的仿射操作来说，由$x = f(z) = Wz$我们可以得到 f 的 Jacobian 行列式的计算结果为：

$$J_f = \begin{bmatrix} \partial x_1/\partial z_1 & \partial x_1/\partial z_2 & \partial x_1/\partial z_3 \\ \partial x_2/\partial z_1 & \partial x_2/\partial z_2 & \partial x_2/\partial z_3 \\ \partial x_3/\partial z_1 & \partial x_3/\partial z_2 & \partial x_3/\partial z_3 \end{bmatrix} = \begin{bmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \\ w_{31} & w_{32} & w_{33} \end{bmatrix} = W$$

代入到整个含有 d 个 3*3 维的仿射变换矩阵当中，得到最终的 Jacobian 行列式的计算结果就为：$(det(W))^{d\times d}$，如下图所示：



因此，引入 1×1 convolution 后的 G 的 Jacobian 行列式计算依然非常简单，所以引入 1×1 convolution 是可取的，这也是 GLOW 这篇 Paper 最有突破和创意的地方。



综上，关于 Flow-based Model 的理论讲解和架构分析就全部结束了，它通过巧妙地构造仿射变换的方式实现不同分布间的拟合，并实现了可逆计算和简化雅各比行列式计算的功能和优点，最终我们可以通过堆叠多个这样的耦合层去拟合更复杂的分布变化（如上图所示），从而达到生成模型需要的效果。

3. 致谢及引用

很感谢李宏毅老师的教程视频，讲得实在是简单通透，视频地址如下：
https://www.bilibili.com/video/av46561029/?p=59

课程资源地址如下：
http://speech.ee.ntu.edu.tw/~tlkagk/courses_ML19.html