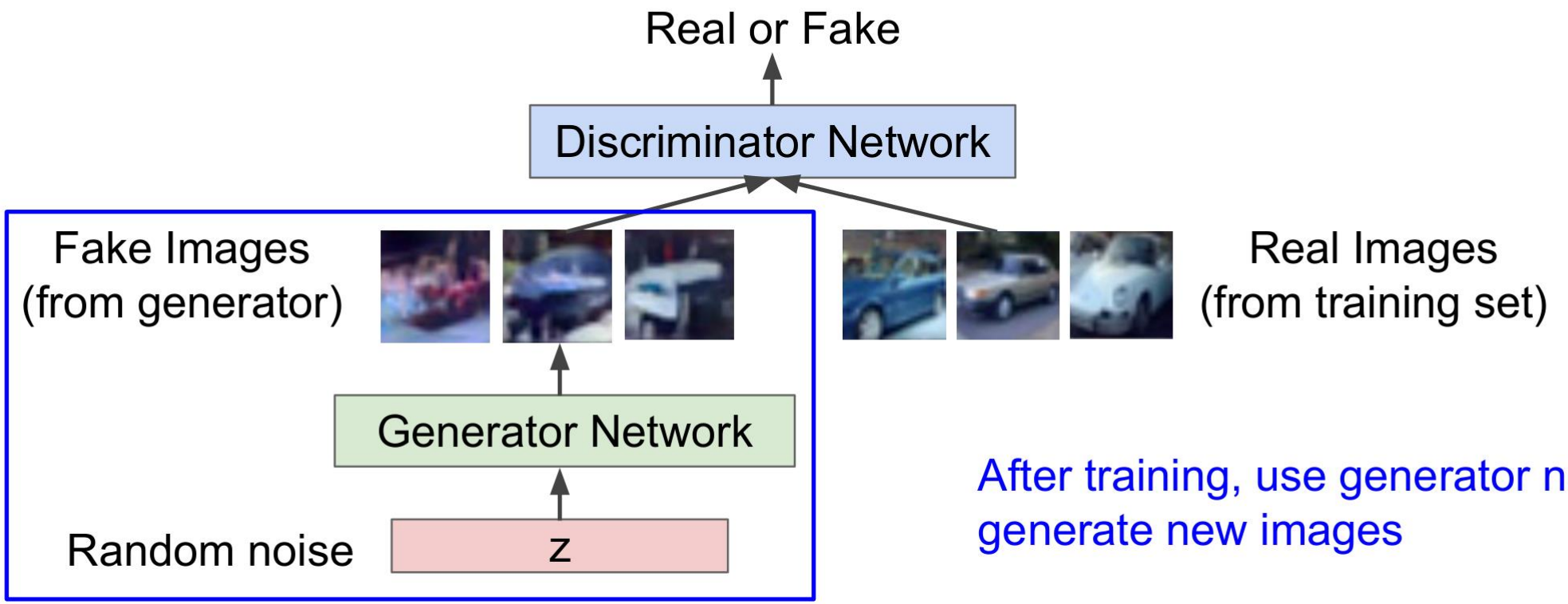
Training GANs: Two-player game

Ian Goodfellow et al., “Generat
Adversarial Nets”, NIPS 2014

Generator network: try to fool the discriminator by generating real-looking images
Discriminator network: try to distinguish between real and fake images



After training, use generator network to generate new images

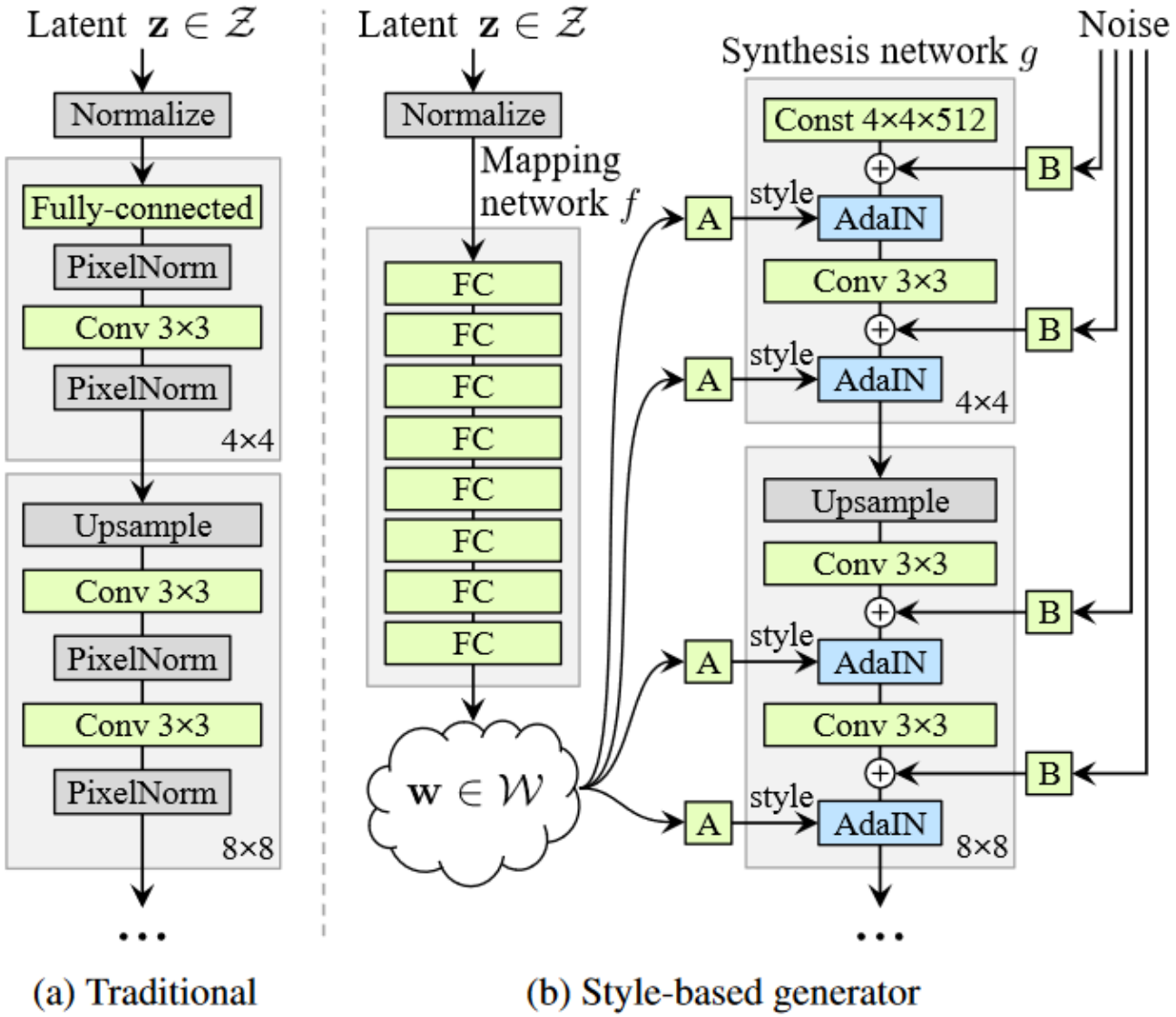
StyleGANs

StyleGAN1

StyleGAN中的“Style”是指数据集中人脸的主要属性，比如人物的姿态等信息，包括了脸型上面的表情、人脸朝向、发型等等，还包括纹理细节上的人脸肤色、人脸光照等方方面面。

StyleGAN 用风格（style）来影响人脸的姿态、身份特征等，用噪声（noise）来影响头发丝、皱纹、肤色等细节部分

模型架构



当传统的生成器只通过输入层输入latent z时，首先将输入映射到一个中间潜在空间W，然后在每个卷积层通过自适应实例标准化(AdaIN)控制生成器。高斯噪声是在每次卷积之后，在进行非线性变化之前添加的。其中，‘ A ’表示学习到的仿射变换，‘ B ’将学习到的每通道缩放因子应用于噪声输入。 **Mapping network** f由8层组成，**Synthesis network** g由18层组成，每个分辨率(4^2~ 1024^2)有2层。最后一层的输出使用单独的1 × 1卷积转换为RGB。

- **Mapping network : 用于将 latent code z 转换成为 w （利用w来实现style的作用）**
- **Synthesis network : 用于生成图像**

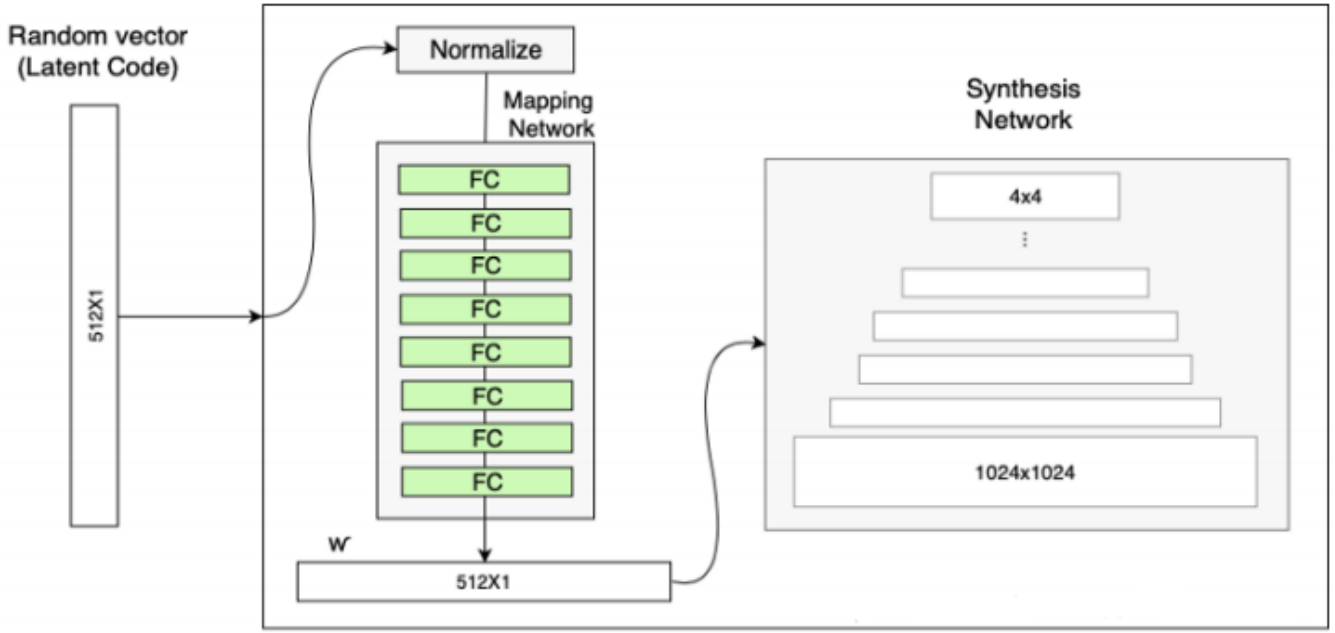
Mapping network

作用：为输入向量z的特征解耦提供一条学习通道

由于z是符合均匀分布或者高斯分布的随机变量，所以变量之间的耦合性比较大。

比如特征：头发长度和男子气概，如果按照z的分布来说，那么这两个特征之间就会存在交缠紧密的联系，头发短了你的男子气概会降低或者增加，但其实现实情况来说，短发男子、长发男子都可以有很强的男子气概。

需要将latent code z进行解耦，才能更好的后续操作，来改变其不同特征。

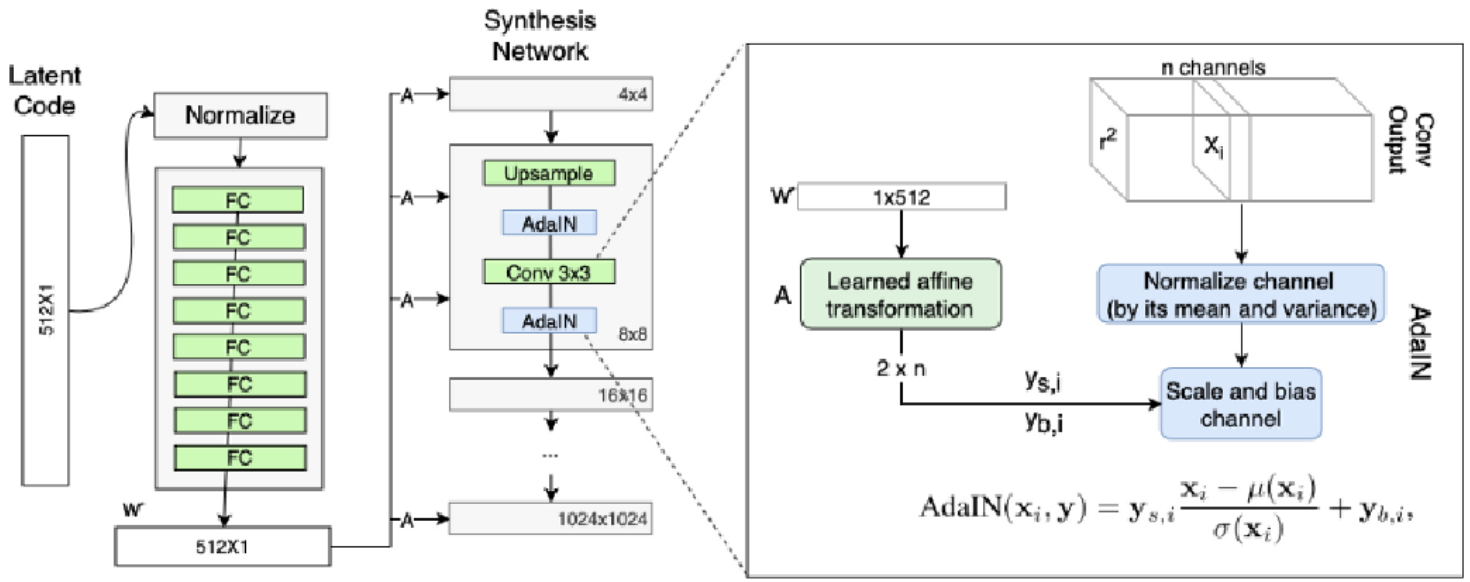


Synthesis network

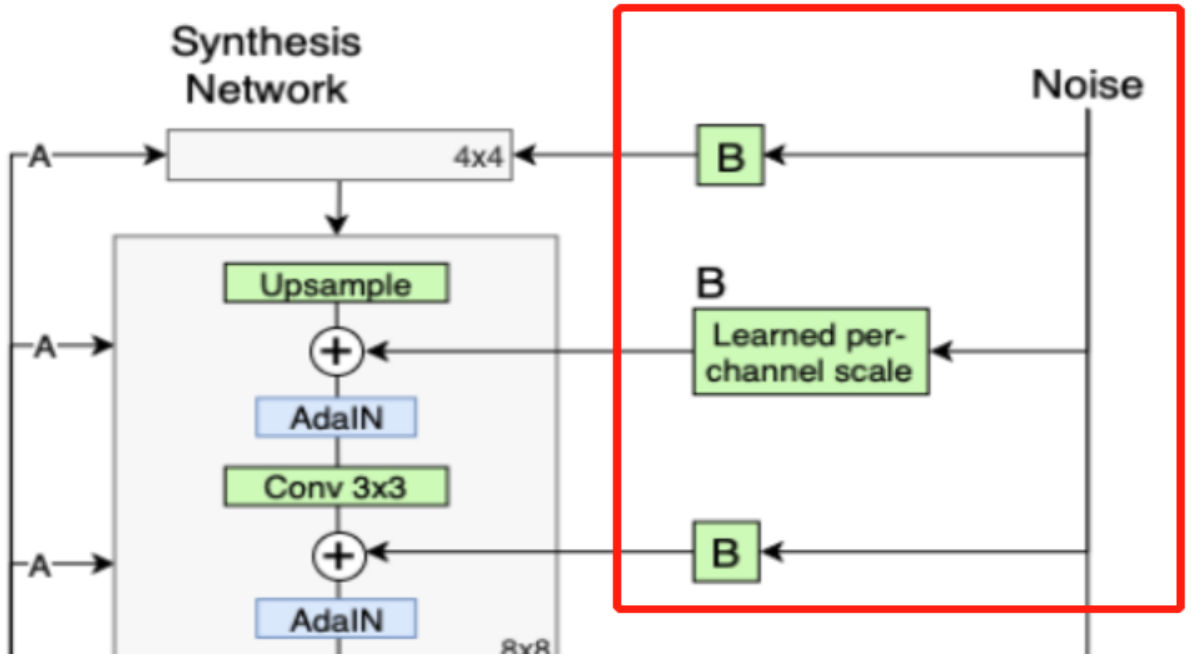
作用：用于生成图像

在Synthesis network中，最初的输入向量shape为 512 * 4 * 4，最后输出为 3 * 1024 * 1024。从4 * 4到 8 * 8 …（4，8，16，32…1024），其中每一个又包含两个卷积层（一个用于upsample，一个用于特征学习），所以一共包含18层。

18 = 1（初始进入的conv层）+8 * 2（每一个块包含的两个卷积层，将vector从4**4变到1024*1024）+1（to_rgb层，将通道变成3）。



Noise



人脸上有很多小特征，比如雀斑、发髻线的准确位置，这些都可以是随机的。将这些小特征插入GAN图像的常用方法是 在输入向量中添加noise。 为了控制噪声仅影响图片样式上细微的变化，StyleGAN 采用类似于 AdaIN 机制的方式添加噪声（噪声输入是由不相关的高斯噪声组成的单通道数据，它们被馈送到生成网络的每一层）。 即在 AdaIN 模块之前向每个通道添加一个缩放过的噪声，并稍微改变其操作的分辨率级别特征的视觉表达方式。 加入噪声后的生成人脸往往更加逼真与多样。

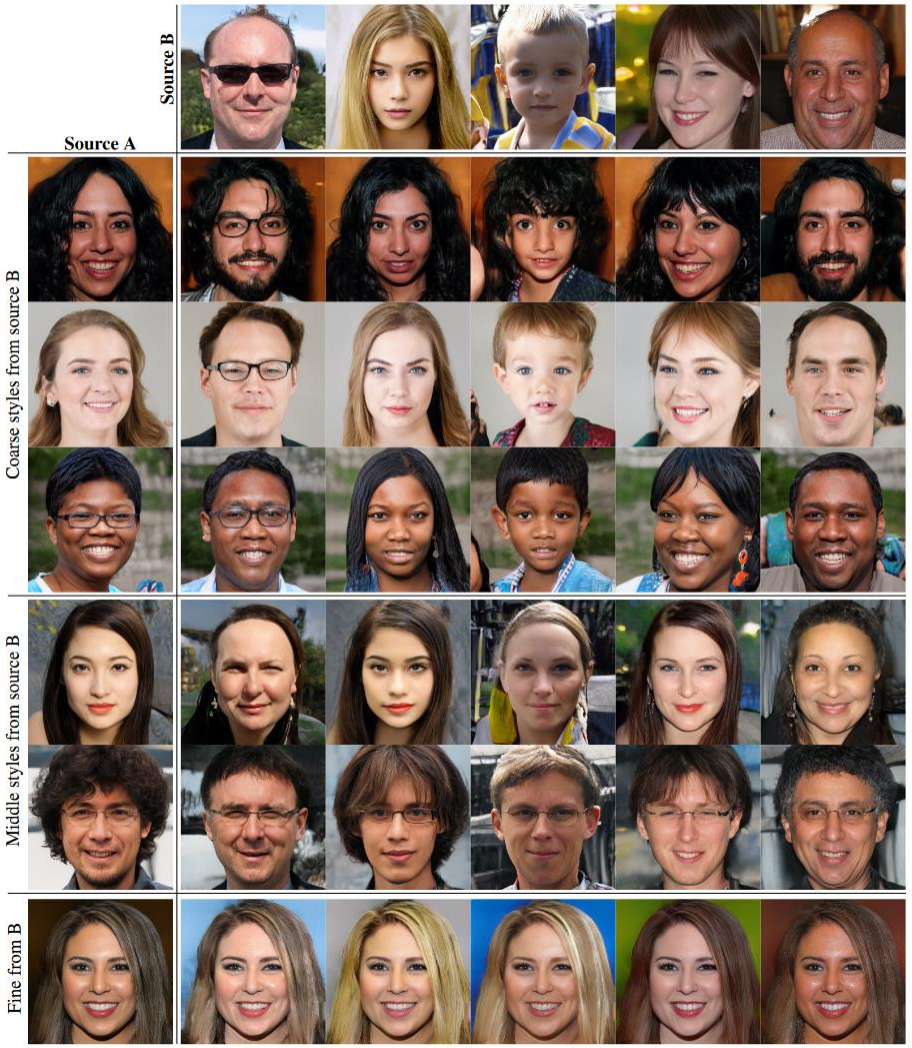
自适应实例归一化（AdaIN）

$$\mathrm{AdaIN}(x_i, y) = \sigma(y)\left(\frac{x_i - \mu(x_i)}{\sigma(x_i)}\right) + \mu(y)$$

特征图的均值和方差中带有图像的风格信息。所以在这一层中，特征图减去自己的均值除以方差，去掉自己的风格。再乘上新风格的方差加上均值，以实现转换的目的。StyleGAN的风格不是由图像得到的，而是w生成的。

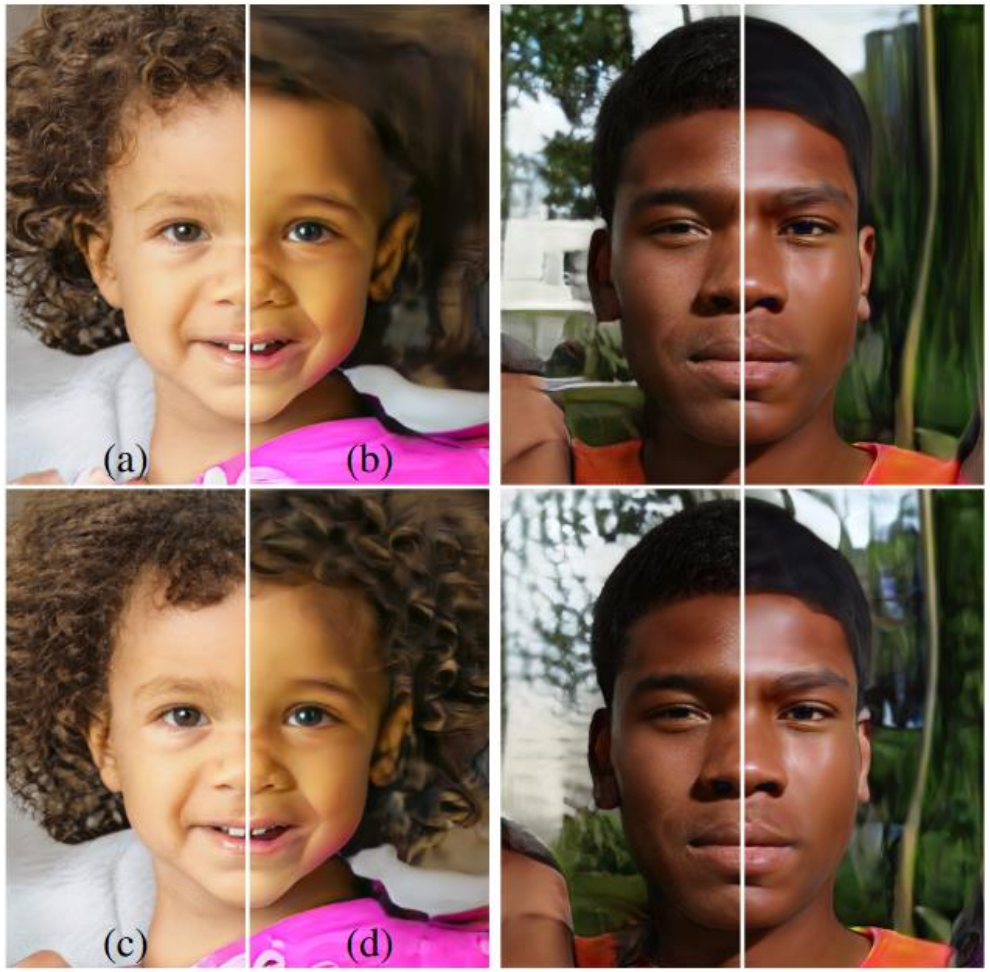
混合正则化（ Style mixing ）

在训练过程中，stylegan采用混合正则化的手段，即在训练过程中使用两个latent code w（不是1个）。通过Mapping network输入两个latent code z，得到对应的w1和w2（代表两个风格），接下来为它们生成中间变量w'。然后利用第一个w1映射转换后来训练一些网络级别，用另一个w2来训练其余的级别，于是便能生成混合了A和B的样式特征的新人脸。



第一部分是 Coarse styles from source B，分辨率(4x4 - 8x8)的网络部分使用B的style，其余使用A的style，可以看到图像的身份特征随souce B，但是肤色等细节随source A；第二部分是 Middle styles from source B，分辨率(16x16 - 32x32)的网络部分使用B的style，这个时候生成图像不再具有B的身份特性，发型、姿态等都发生改变，但是肤色依然随A；第三部分 Fine from B，分辨率(64x64 - 1024x1024)的网络部分使用B的style，此时身份特征随A，肤色随B。

低分辨率的style 控制姿态、脸型、配件 比如眼镜、发型等style，高分辨率的style控制肤色、头发颜色、背景色等style。

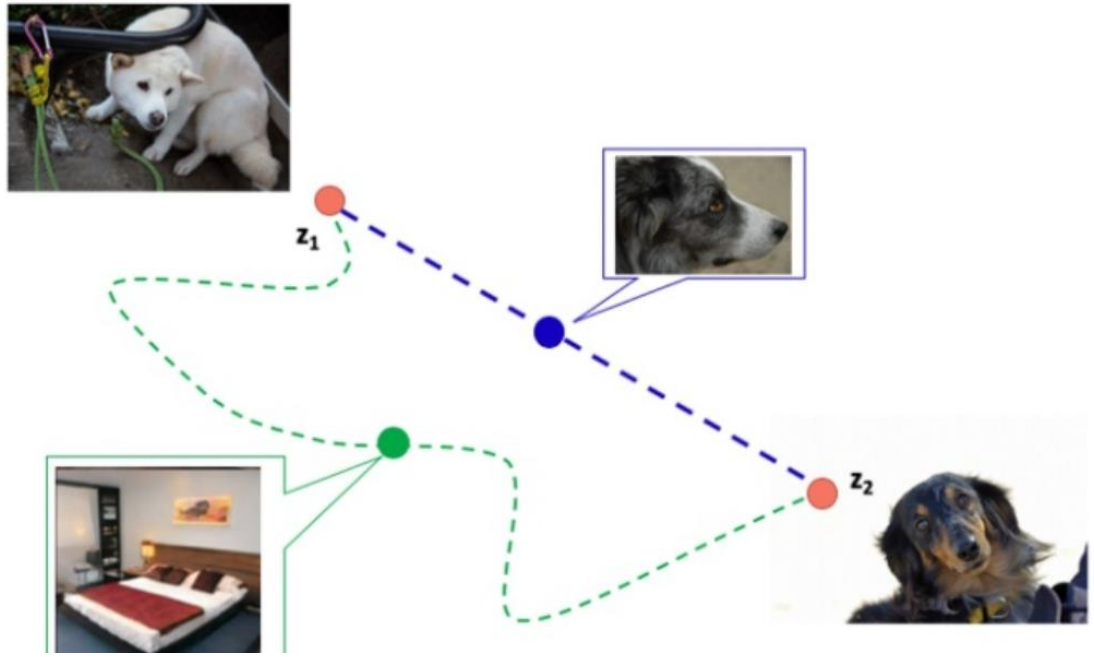


(a)对所有层施加noise。
(b)没有noise 。
(c)只有(64^2 – 1024^2)的细层noise 。
(d)只有(42 ~ 322)的粗层noise 。

我们可以看到，粗层noise会引起头发的大尺度卷曲和较大背景特征的出现，而细层noise则会带来头发更细的卷曲、更细的背景细节和皮肤毛孔变化。

如何量化解耦

1.感知路径长度（Perceptual path length）



已知input latent code 是z1，或者说白色的狗所表示的latent code是z1，目标图像是黑色的狗，黑狗图像的latent code 是z2。图中紫色的虚线是z1 到 z2 最快的路径，绿色的曲线是我们不希望的路径。

具体做法如下：

(1) 使用两个VGG16提取特征的加权差异来表示一对图像间的感知距离。

(2) 将潜在空间插值路径细分为线性段，每个段上的感知差异的总和就是感知路径长度。

(3) 使用多份样本，分别计算z和w的PPL（感知距离长度）。由于z已经归一化，所以对z使用球面插值 slerp，而对w使用线性插值 lerp。评估为裁剪后仅包含面部的图像。

$$l_z = \mathbb{E}\left[\frac{1}{\epsilon^2} d\left(G\left(\text{slerp}\left(\mathbf{z}_1, \mathbf{z}_2; t\right)\right)\right.,\right.$$
$$\left.\left. G\left(\text{slerp}\left(\mathbf{z}_1, \mathbf{z}_2; t+\epsilon\right)\right)\right)\right],$$

$$l_w = \mathbb{E}\left[\frac{1}{\epsilon^2} d\left(g\left(\text{lerp}\left(f(\mathbf{z}_1), f(\mathbf{z}_2); t\right)\right)\right.,\right.$$
$$\left.\left. g\left(\text{lerp}\left(f(\mathbf{z}_1), f(\mathbf{z}_2); t+\epsilon\right)\right)\right)\right],$$

如何量化解耦

2.线性可分性（linear separability）

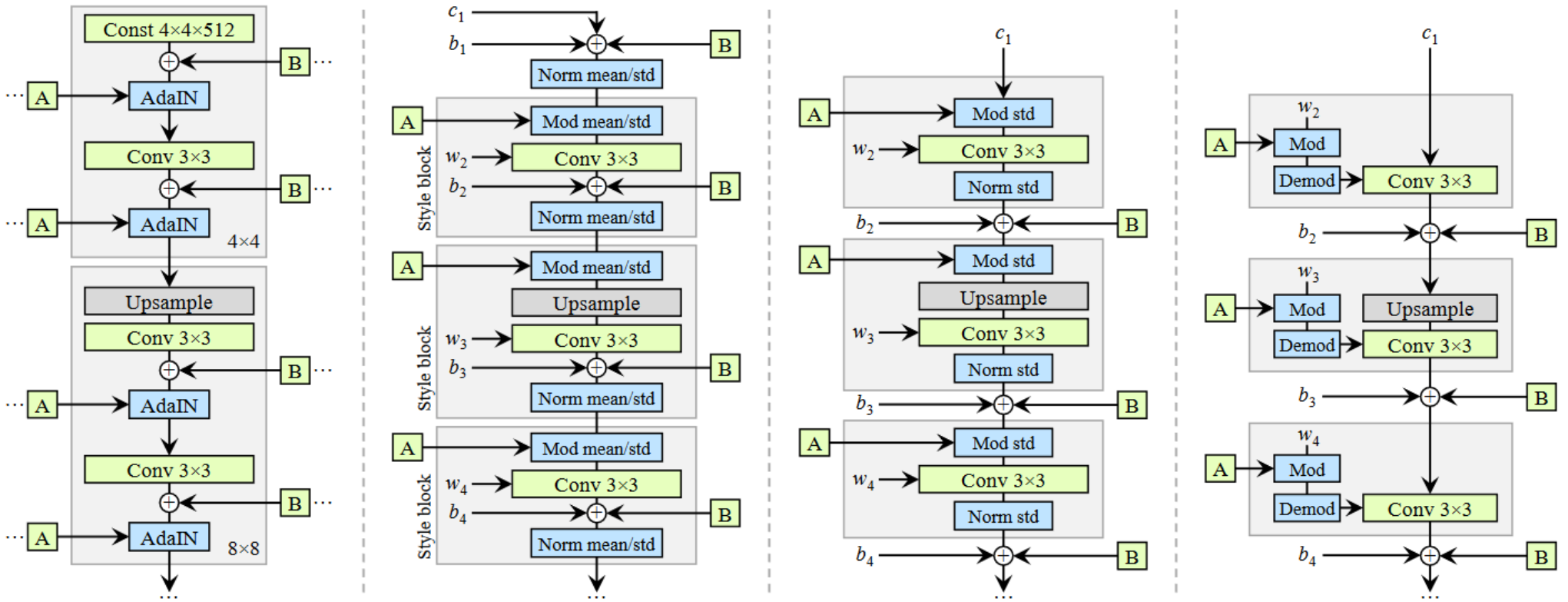
如果隐空间与图像特征足够解耦，那么隐空间中存在线性超平面，可以二分类两种特征。在stylegan的文章中，基于CelebA-HQ数据集，训练40种特征的分类器。然后用生成器生成200000张图像，用训练的分类器分类，去掉置信度最低的一半，得到隐变量和标签已知的100000张图像。对每个属性，用线性SVM拟合预测z的类别，判断z是否足够线性。

线性关系用X和Y的分布差异衡量。计算条件熵H (Y | X)，其中X为SVM预测的类别，Y为预训练分类器确定的类别。这告诉我们需要多少额外的信息来确定一个样本的真实类别，因为我们知道它位于超平面的哪一边。较低的值表明对应的变异因子具有一致的潜在空间方向。

StyleGAN2

水滴状伪影 （blob-shaped artifacts）

作者将问题归结为AdaIN操作，该操作分别对每个特征图的均值和方差进行归一化处理，从而可能破坏在特征相对大小中发现的任何信息。我们假设液滴伪影是生成器故意通过实例归一化来隐藏信号强度信息的结果：通过创建一个强大的、局部的、主导统计的尖峰，生成器可以像在其他地方一样有效地缩放信号。假设得到了以下发现的支持：当标准化步骤从生成器中移除时，水滴状伪影完全消失。

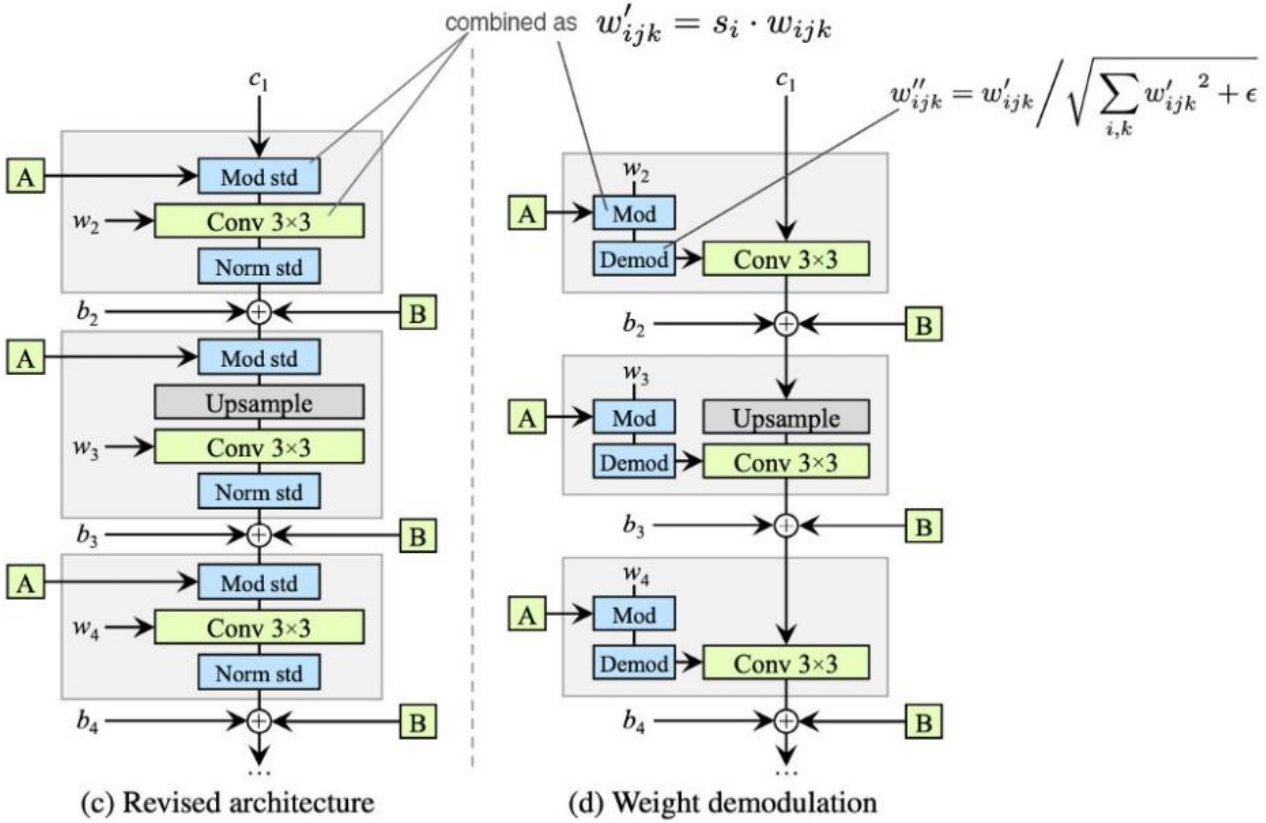


（a）初始StyleGAN，其中A表示由W学习得到的产生某种风格的仿射变换，B为噪声广播操作。
（b）具有完整细节的同一图。在这里，我们打破了AdaIN的显式标准化，然后进行调制，两者都在每个特征图的均值和标准差上操作。我们还对学习到的权重（w）、偏差（b）和常量输入（c）进行了标注，并重新绘制了灰色方框，使得每个方框有一个样式是激活的。激活函数(泄漏ReLU)总是在添加偏置后立即应用。
（c）我们对原架构作了几处改动，这些改动在主要案文中是合理的。我们在开始时去除一些冗余操作，将b和B的添加移动到一个样式的外部活动区域，并且只调整每个特征图的标准差。
（d）修改后的架构能够用“解调”操作代替实例归一化，将其应用于与每个卷积层相关的权重。

- **移除最开始的数据处理**
- **在标准化特征时取消均值**
- **将noise模块在外部style模块添加**

weight demodulation

在样式混合（style mixing）中，容易将某个特征放大一个数量级或更多，而在去除了Adain后，便无法有效控制这个问题（因为移除了mean），但style mixing又是stylegan的亮点，如何能够在保留style mixing的同时有效地解决特征放大问题呢？这便是weight demodulation设计的原因。



实例归一化（AdaIN）的目的是从卷积输出特征图的统计中本质上去除s的影响。

通过相应权重的L2范数对输出进行缩放。随后的归一化旨在将输出恢复到单位标准差。如果我们将("解调")每个输出特征映射j缩放1 / σ j，就可以实现这一点。

Lazy regularization

loss 是由损失函数和正则项组成，优化的时候也是同时优化这两项的。

lazy regularization就是正则项可以减少优化的次数，比如每16个minibatch才优化一次正则项，这样可以减少计算量，同时对效果也没什么影响。

Path length regularization

在生成人脸的同时，我们希望能够控制人脸的属性，不同的latent code能得到不同的人脸，当确定latent code变化的具体方向时，该方向上不同的大小应该对应了图像上某一个具体变化的不同幅度。为了达到这个目的，设计了 Path length regularization。

我们希望鼓励W中的一个固定大小的步骤导致图像中的非零、固定大小的变化。我们可以通过进入图像空间中的随机方向并观察相应的w个梯度来经验地衡量偏离这个理想的程度。这些梯度不管是w还是图像空间方向，都应该具有接近相等的长度，表明从潜在空间到图像空间的映射是有条件的

在单个w∈W时，生成元映射g（w）：W →Y的局部度量标度性质由Jacobian矩阵Jw =∂g（w）/∂w捕获

$$\mathbb{E}_{\mathbf{w},\mathbf{y}\sim\mathcal{N}(0,\mathbf{I})}\left(\left\|\mathbf{J}_{\mathbf{w}}^{T}\mathbf{y}\right\|_{2}-a\right)^{2},$$

其中y是像素强度服从正态分布的随机图像，w∈f（z），其中z服从正态分布。，在高维情况下，当Jw在任意w处正交(上升到全球范围)时，这个先验被最小化。一个正交矩阵保持长度且不引入任何维数的压缩。

StyleGAN3


StyleGAN2
StyleGAN3 (Ours)

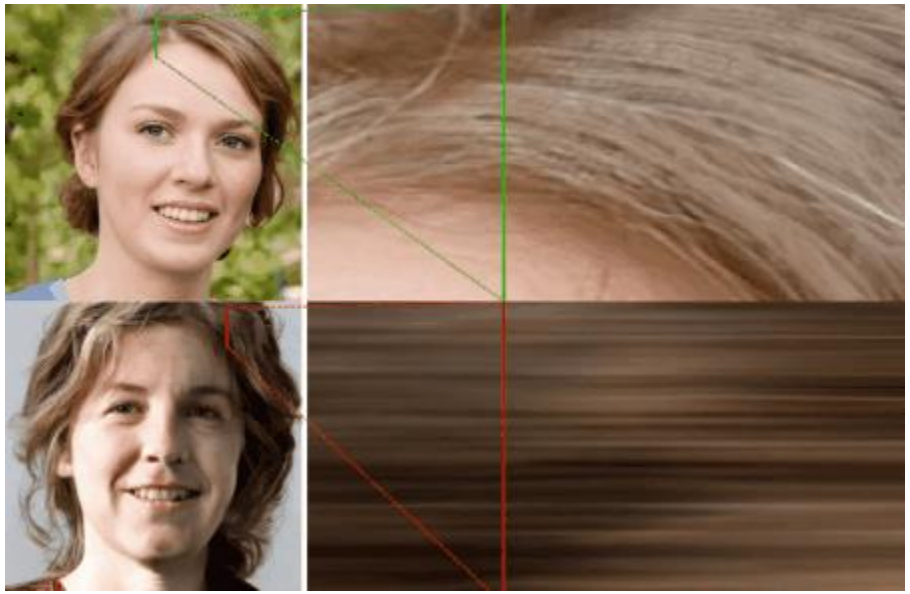

StyleGAN2
StyleGAN3 (Ours)

通过上图可以看出来stylegan2生成的毛发等细节会粘在屏幕上，和人物的形态变化不一致。

生成的过程并不是一个自然的层次化生成。
粗糙特征（GAN的浅层网络的输出特征）主要控制了精细特征（GAN的深层网络的输出特征）的存在与否，并没有精细控制它们的出现的精确位置。

简单来讲：如果训练集中大多数图片的眼睛出现在图片居中位置，那么哪怕你的浅层网络输出中图片的眼睛在其他位置，最终整个网络输出的图片眼睛还是基本会处于图片居中位置。

作者们认为产生这个现象的根本问题是：目前的生成器网络架构是**卷积+非线性+上采样等结构**，而这样的架构无法做到很好的Equivariance（等变性）



在人脸平移时，对于某个固定的坐标切片，stylegan3可以随人脸移动变化纹理（比如右图那一块固定区域，在左图人脸平移时，也是在不断移动的），而stylegan2则是倾向于生成固定的纹理。

同时stylegan2也无法满足旋转一致性问题

通过实验表明，这些问题出现原因在哪里呢?

（1）使用图像边界参考（例如padding的使用实际上提示了图像的边界位置信息）
（2）像素的噪声输入（防止头发丝粘一起）
（3）位置编码和混叠（加入一些额外的位置信息）。

如何消除这些不需要的辅助信息，从而防止网络去组合、使用这些错误信息呢?

针对边界问题，可以在稍大一点的图像边缘上进行操作。
针对像素的噪声输入，可以直接取消掉该操作。

而关于生成图像中的混叠，作者认为有两个可能的来源:
非理想上采样（如最近邻、双线性或空洞卷积）产生的像素网格的微弱的印记。
应用pointwise的非线性激活，如ReLU或者Swish。

网络会去放大哪怕是最轻微的混叠，并在多个尺度上进行组合，这就成为了固定在图像坐标中的那些混叠状网格的基础（解释了为何特征会依赖于坐标）。而实验表明，在stylegan2时，当前的上采样滤波器在抑制混叠方面根本不够积极，而且需要具有超过100dB衰减的高质量滤波器。

而针对这个问题，stylegan3根据它们在连续域的影响，提出了一种解决点态非线性引起的混叠的原理，并对结果进行了适当的低通滤波。

傅里叶特征以及基线模型简化

(1) 利用Fourier特征（傅里叶特征）代替了stylegan2生成器的常数输入
(2) 删除了noise输入（特征的位置信息要完全来自前层粗糙特征）
(3) 降低了网络深度（14层，之前是18层），禁用mixing regularization和path length regularization，并且在每次卷积前都使用简单的归一化 （这里有点直接推翻了stylegan2的一些思想）

边界及上采样问题优化

(4) 理论上特征映射有一个无限的空间范围（就是特征不能局限固定在某些坐标域），引入了一个固定大小的boundaries来近似，每一层操作后再crop到这个扩展的空间。 用理想低通滤波器来代替双线性采样。 改进后的boundaries和upsampling得到了更好的平移性，但是fid变差了。

非线性滤波改进

(5) 工程化的操作，改进了非线性滤波，并将整个upsample-LReLU-downsample过程写到一个自定义的CUDA内核

非临界采样改进

(6) 为了抑制 aliasing（混淆对于生成器的Equivariance很有害），可以简单地将截止频率降低，从而确保所有混叠频率都在阻带。
【丢掉了浅层的高频细节，因为作者认为浅层中高频细节并不重要】

傅里叶特征改进

(7) 为了应对每层的全局变换能力有限的问题，加入了一个仿射层，通过输出全局平移和旋转参数输入傅里叶特征，稍微改善了FID

平移不变性

(8) 虽然提高了平移不变性，但是一些可见的伪影仍然存在。这是因为滤波器的衰减对于最低分辨率的层来说仍然是不够的，而这些层往往在其带宽限制附近有丰富的频率信息，这就需要有非常强的衰减来完全消除混叠。提出：跳频在最低分辨率层中较高，以最大化提高阻带的衰减；在最高分辨率层中较低，以匹配高层细节

旋转不变性

(9) 为了得到旋转不变性网络，做出来两个改进：将所有层的卷积核大小从3 * 3替换为1 * 1、通过将feature map的数量增加一倍，用来弥补减少的特征容量
实验表明，一个基于 1 * 1的卷积操作能够产生强旋转的等变生成器，一旦适当地抑制了混叠，便可以迫使模型实现更自然的层次细化。