北京大学
长沙计算与数字经济研究院
PKU-Changsha Institute for Computing and Digital Economy

周一组会 AI研究分享

分享人：姚文翰
主题：Style Transfer in Deep Learning
内容：
CycleGAN 系列
StarGAN 系列
AdaIN系列
WadaIN系列
Styleformer
ControlNext

- **2024年9月23日**

一、CycleGAN系列:CycleGAN

Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks

Jun-Yan Zhu* Taesung Park* Phillip Isola Alexei A. Efros
Berkeley AI Research (BAIR) laboratory, UC Berkeley

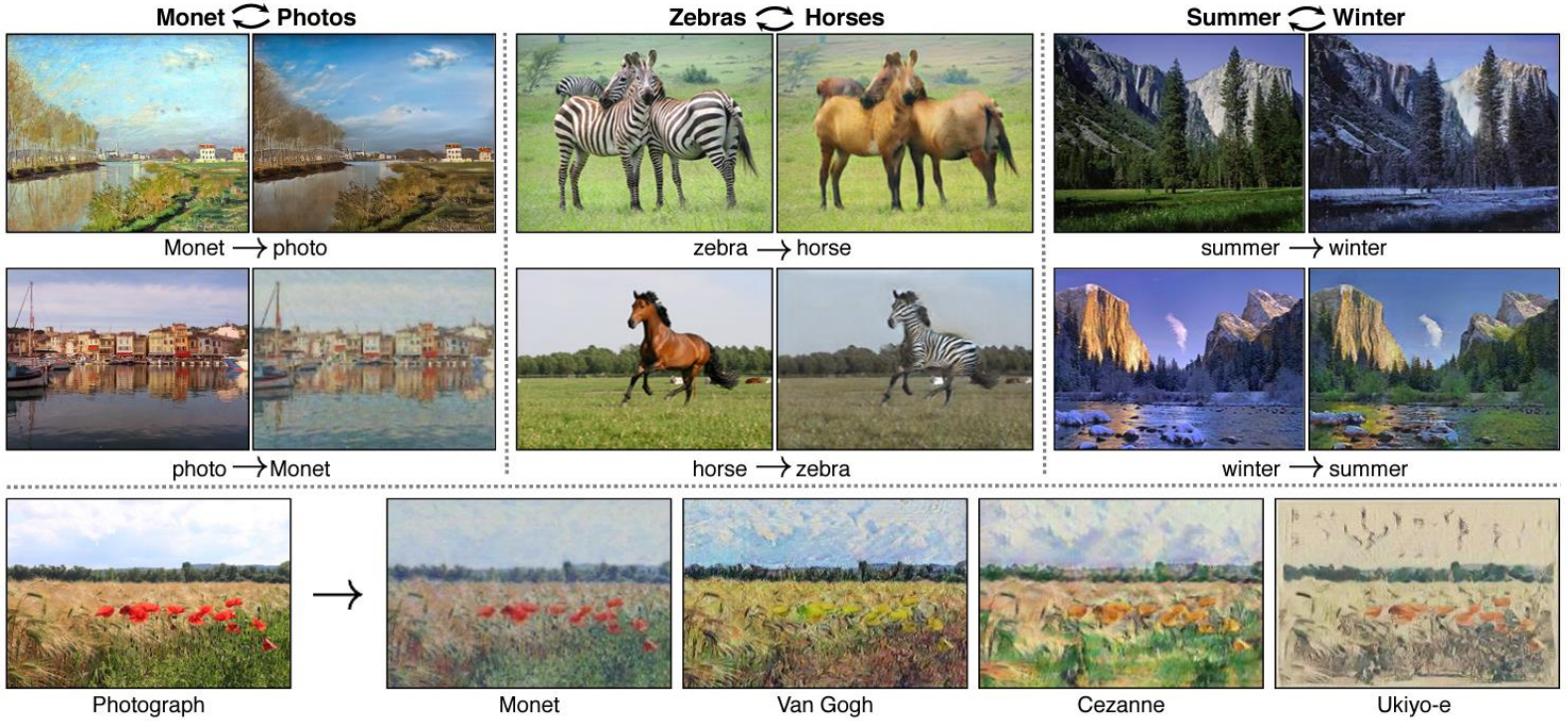


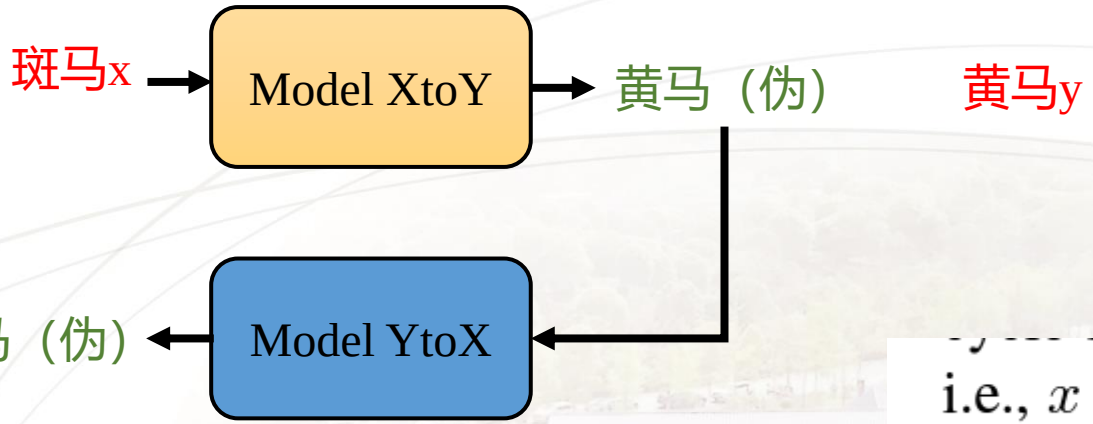
Figure 1: Given any two unordered image collections X and Y, our algorithm learns to automatically "translate" an image from one into the other and vice versa: *(left)* Monet paintings and landscape photos from Flickr; *(center)* zebras and horses from ImageNet; *(right)* summer and winter Yosemite photos from Flickr. Example application *(bottom)*: using a collection of paintings of famous artists, our method learns to render natural photographs into the respective styles.

作者：JY Zhu · 2017 · 被引用次数：24051 —

[Submitted on 30 Mar 2017 (v1), last revised 24 Aug 2020 (this version, v7)]

训练逻辑

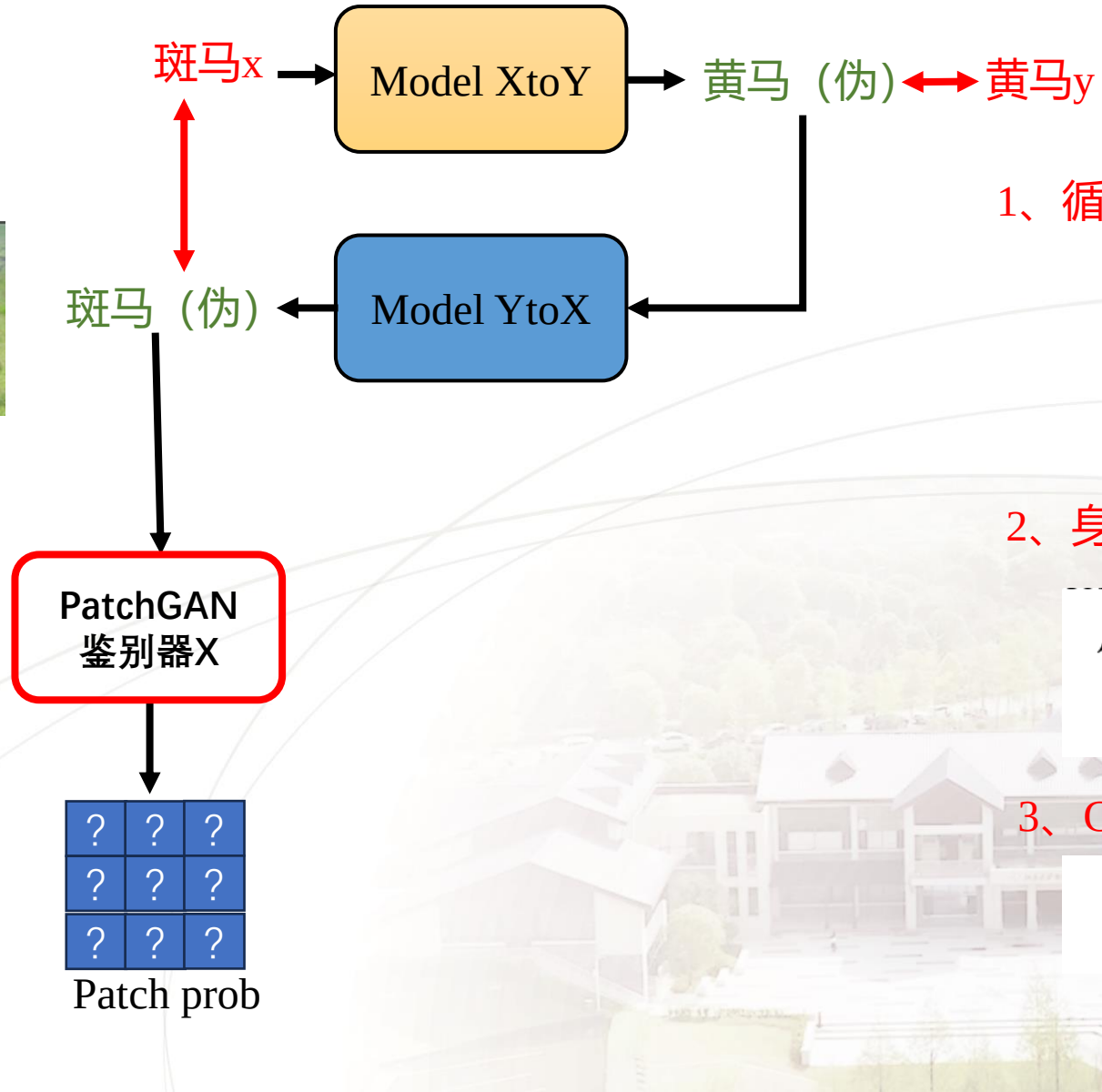
设斑马图像x为风格X，黄马图像y为风格Y



i.e., $x \rightarrow G(x) \rightarrow F(G(x)) \approx x.$

训练时，
初始化2个模型：modelXtoY, Model YtoX
For i in range(M):
    输入数据：x,y
    x’ = modelXtoY(x)
    y’ = modelYtoX(x’)

损失函数



1、循环一致性损失：

$$\mathcal{L}_{\text{cyc}}(G,F) = \mathbb{E}_{x\sim p_{\text{data}}(x)}[\|F(G(x)) - x\|_1] + \mathbb{E}_{y\sim p_{\text{data}}(y)}[\|G(F(y)) - y\|_1].$$

保证X-Y-X对称性

2、身份一致性损失：

$$\mathcal{L}_{id}(G_{X\to Y}, G_{Y\to X}) = \mathbb{E}_{y\sim P_Y(y)}[\|G_{X\to Y}(y) - y\|_1] + \mathbb{E}_{x\sim P_X(x)}[\|G_{Y\to X}(x) - x\|_1]. \quad (3)$$

保证X-Y一致性

3、GAN生成对抗损失：

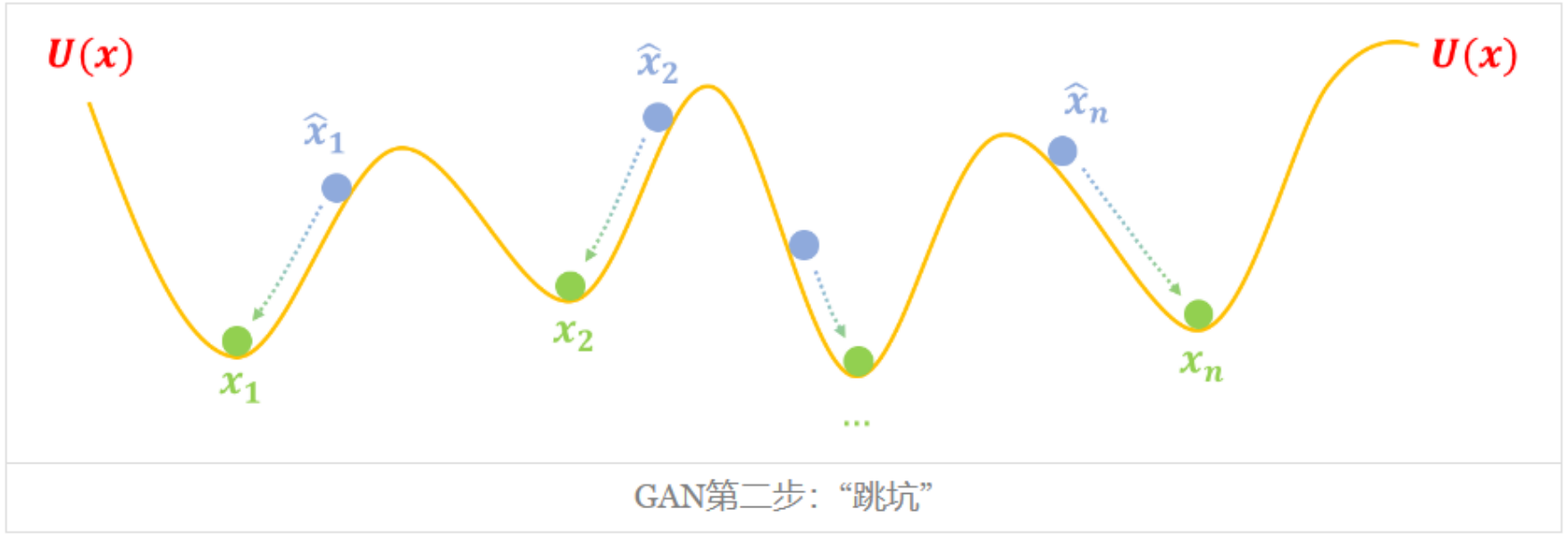
$$\mathcal{L}_{adv}(G_{X\to Y}, D_Y) = \mathbb{E}_{y\sim P_Y(y)}[\log D_Y(y)] + \mathbb{E}_{x\sim P_X(x)}[\log(1 - D_Y(G_{X\to Y}(x)))], \quad (1)$$

按理说有2个G loss,2个 D loss

3、GAN生成对抗损失：

$$\mathcal{L}_{adv}(G_{X\to Y}, D_Y) = \mathbb{E}_{y\sim P_Y(y)}[\log D_Y(y)] + \mathbb{E}_{x\sim P_X(x)}[\log(1 - D_Y(G_{X\to Y}(x)))], \quad (1)$$

能量视角的GAN



GAN第二步：“跳坑”

a.判别器负责判断样本属于能量函数中的高低位置。假样本获得高分数，真样本获得低分数。 Loss = $D(x)^2$ + $[1- D(x)]^2$

b.生成器G 负责让生成的样本靠近坑底，loss = D(G(x))，则假样本获得低分。

https://spaces.ac.cn/archives/6316

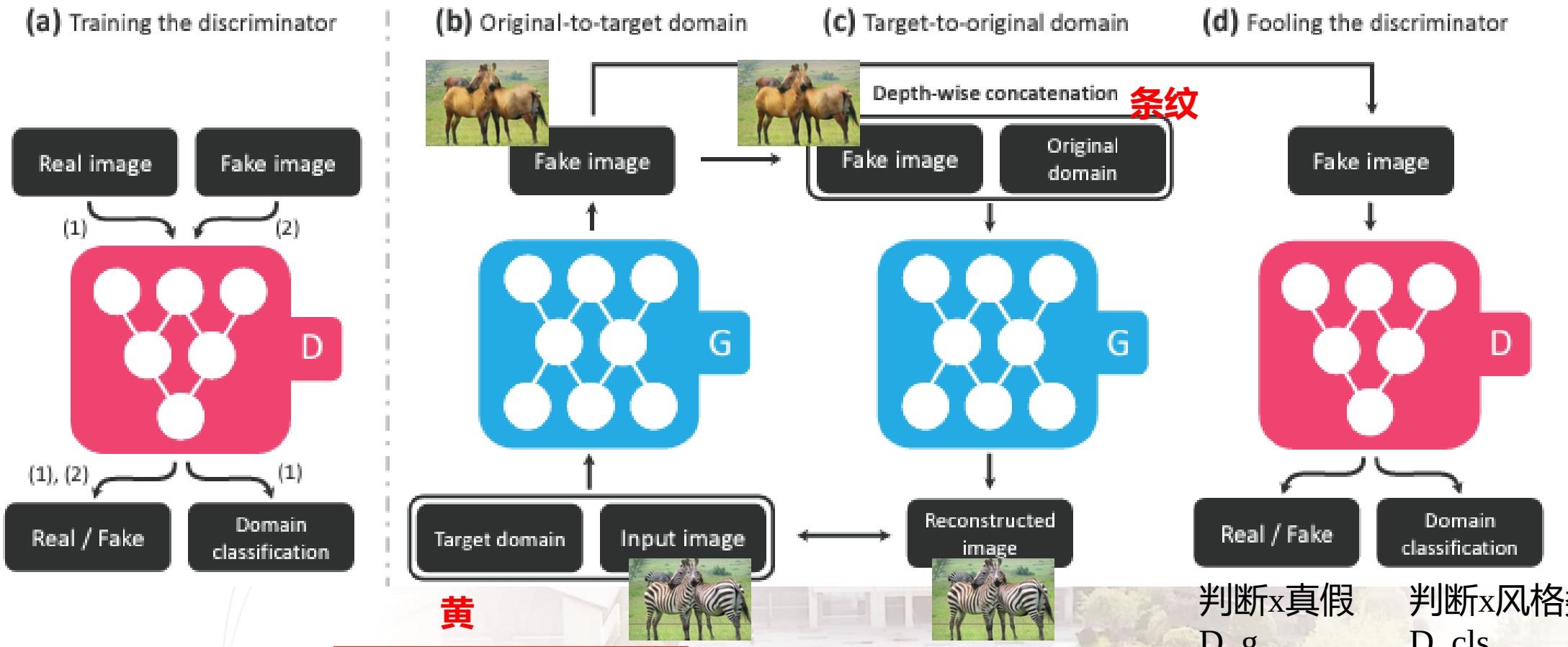
c.于是假样本的分布越来越靠近真样本。
GAN训练则可完成。

一、CycleGAN系列:其他语音领域CycleGAN

模型名	生成器	鉴别器	损失函数
Cyclegan vc	纯CNN	纯CNN，输出对整张图的判断概率，一个浮点数。	只对斑马（伪）计算了1 G loss 1 D loss
Cyclegan-vc2	加入6个残差模块在CNN中间	PatchGAN,输出多个区域的概率	对斑马（伪）、黄马（伪）计算。2 G loss , 2 D loss
Cyclegan-vc3	加入一个time-frequency adaptive normalization (TFAN) 模块来处理2D输入数据。		
Cycleganvc-Mask	对输入数据进行随机mask，强迫模型预测效果提升		

风格迁移的目标风格，是由G_xtoy()函数本身和训练数据x,y决定的。推理时，仅仅输入样本x。
因此，cyclegan无法灵活控制风格，应用限制较大。

《StarGAN: Unified Generative Adversarial Networks for Multi-Domain Image-to-Image Translation》



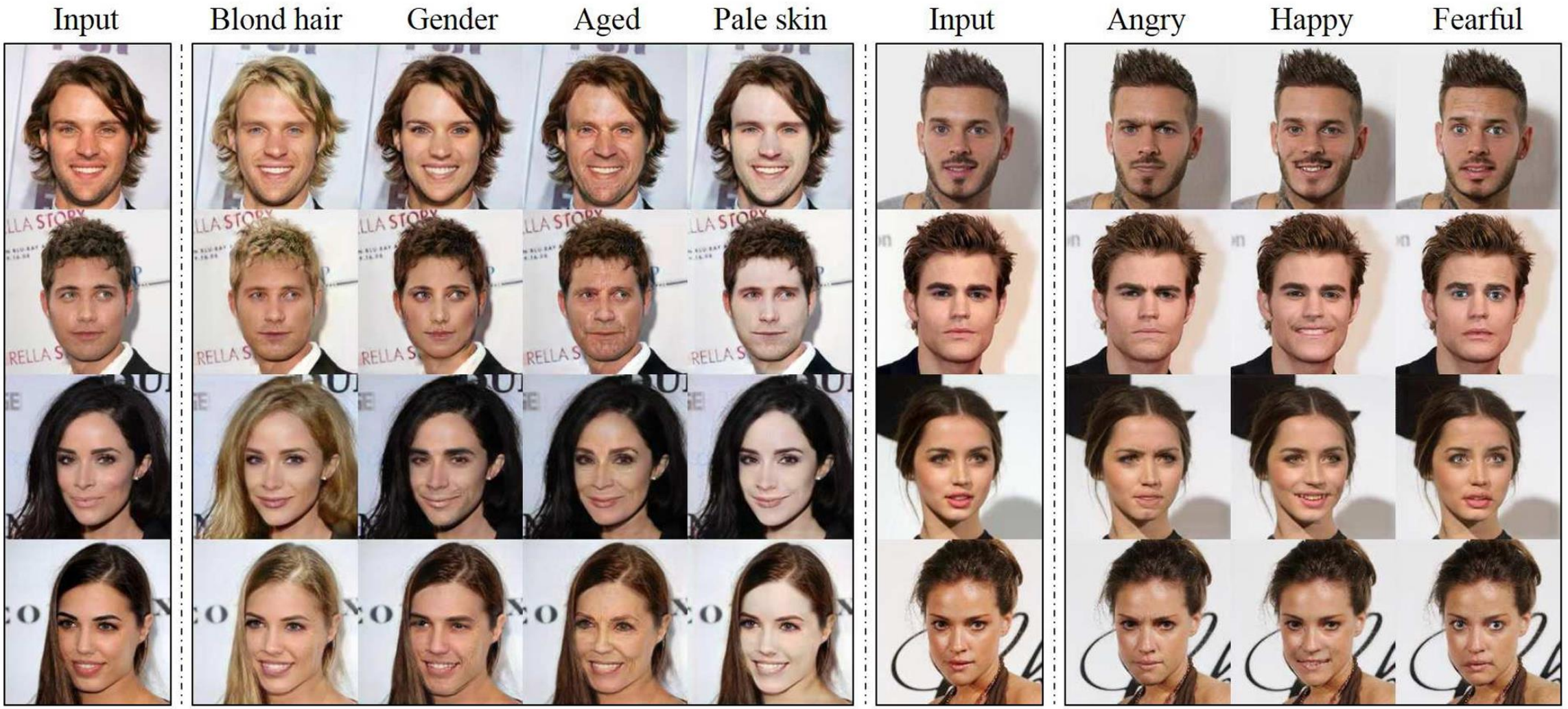
黄

判断x真假 D_g

判断x风格类别 D_cls

迁移公式：$G(x, c) \rightarrow y$

条件控制　　在deep learning 中， 如何将条件c变量跟x融合?

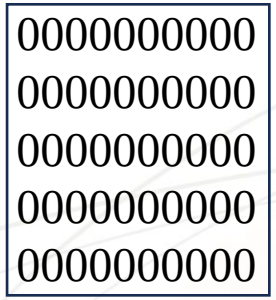


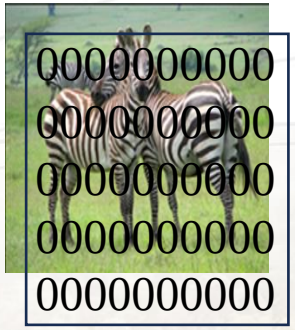
条件控制之一 label condition：

假设训练数据包含5种风格，分别映射为01234。
在训练的每次迭代时，随机采样一批数据x，和一批目标风格c，c为整数向量。

```
def forward(self, x, c):
    # Replicate spatially and concatenate domai
    # Note that this type of label conditioning
    # This is because instance normalization ig
    c = c.view(c.size(0), c.size(1), 1, 1)
    c = c.repeat(1, 1, x.size(2), x.size(3))
    x = torch.cat([x, c], dim=1)
    return self.main(x)
```

concatenate





[B,3,64,64

B

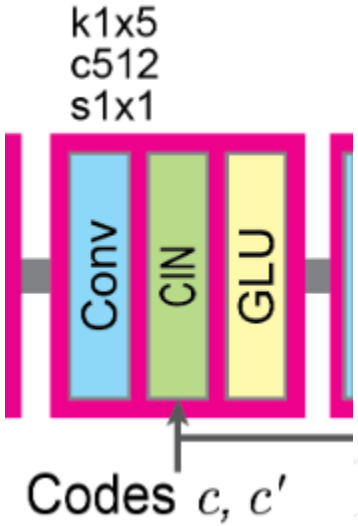
B,1,64,64

[B,4,64,64

缺陷：
1、x和c的拼接方式在GAN较难学习；一旦风格过多， 则整数数字也较大。与图像0-1的数值范围不匹配；
2、无法拓展新风格。

二、StarGAN系列：CIN

条件控制之二 condition IN：

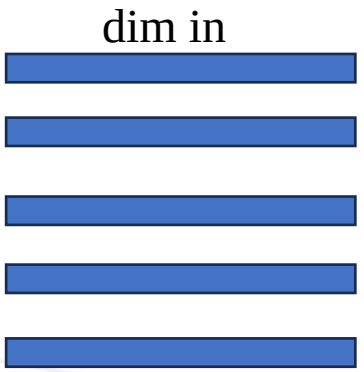


https://github.com/yunjey/stargan/blob/master/model.py

```python
class ConditionalInstanceNormalisation(nn.Module):
    """AdaIN Block."""

    def __init__(self, dim_in, dim_c):
        super(ConditionalInstanceNormalisation, self).__init__()

        self.device = torch.device("cuda:0" if torch.cuda.is_available() else "cpu")

        self.dim_in = dim_in
        self.gamma_t = nn.Linear(dim_c, dim_in)
        self.beta_t = nn.Linear(dim_c, dim_in)

    def forward(self, x, c_trg):
        u = torch.mean(x, dim=2, keepdim=True)
        var = torch.mean((x - u) * (x - u), dim=2, keepdim=True)
        std = torch.sqrt(var + 1e-8)

        gamma = self.gamma_t(c_trg.to(self.device))
        gamma = gamma.view(-1, self.dim_in, 1)
        beta = self.beta_t(c_trg.to(self.device))
        beta = beta.view(-1, self.dim_in, 1)

        h = (x - u) / std
        h = h * gamma + beta

        return h
```

初始化2个权重矩阵。Gamma和beta

X:[B,D,H,W]
Gamma :[B,D]



本质上就是将每种风格表示为dim_in的向量。设5种风格，dimin=128,则风格权重有5个128维度的向量。
但是究竟哪个向量表示哪种风格，需要模型自己学习。如何学习？用一个onehot条件向量对W去乘。 **相当于每次从W中挑选序号为c的向量。**

举例：
【1,0,0】 * W【3,128】 = [1,128]
→（B，C）@ (B,C,D)= (B,D)

（1）根据输入c， 从W中选择风格向量表示gamma和beta
（2）归一化x
（3）仿射乘g b

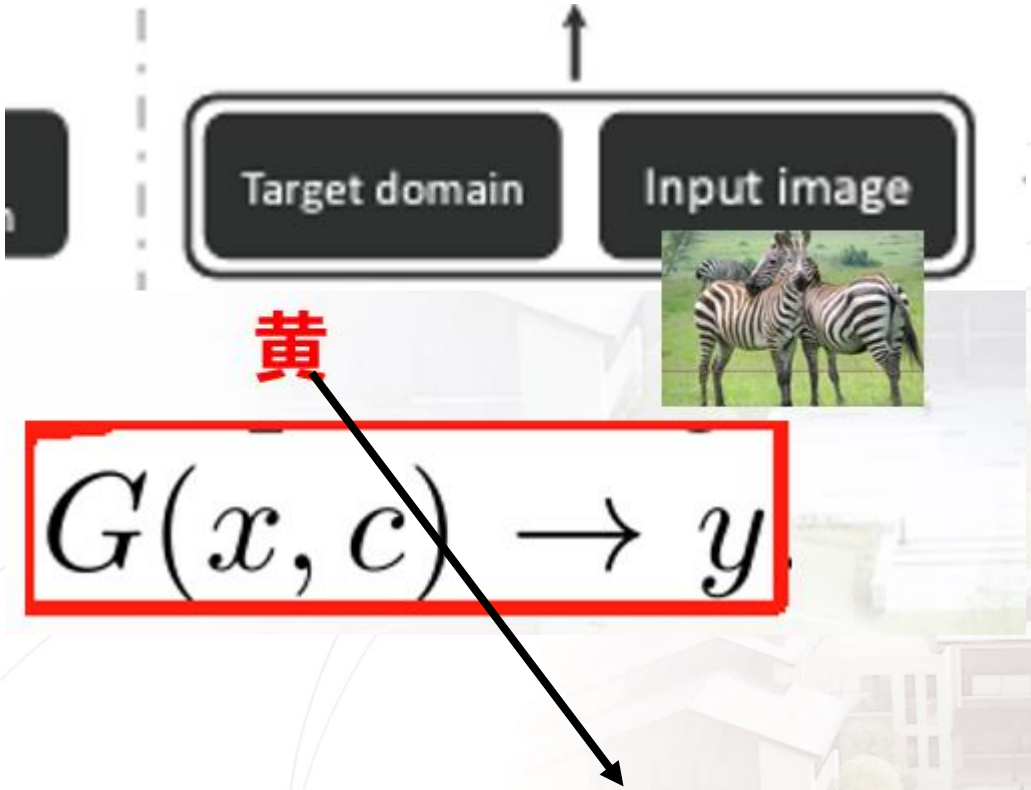
二、StarGAN系列：CIN

缺陷：

1、解决了stargan的问题1，但conIN的W初始化方式影响到不同风格的学习。

2、无法拓展新风格。

解决思路：

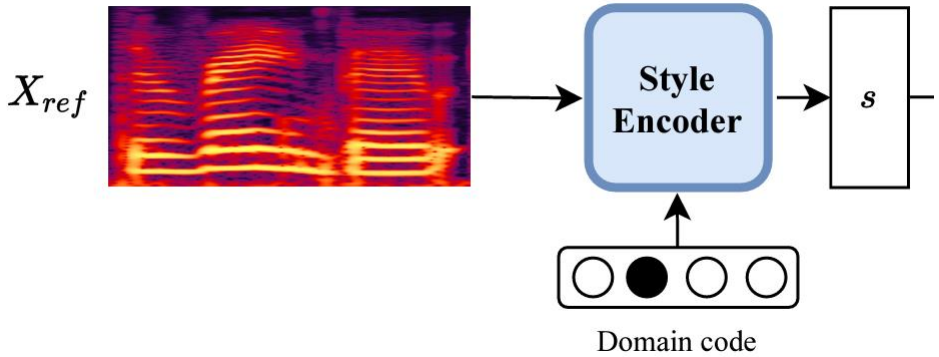


1、这个条件不应该无法拓展

2、应该与目标风格的单例数据有所联系。

二、StarGAN系列：AdaIN

（1）提取目标数据的风格向量s



假设这个encoder提取出来是【B，style_dim】

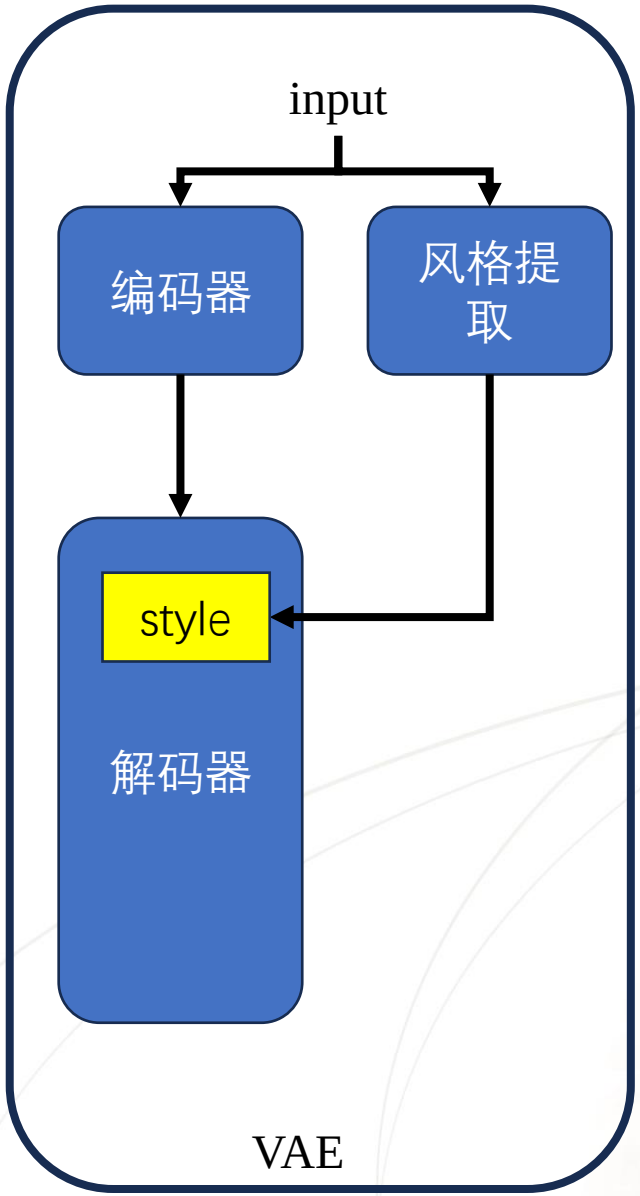
（2）利用s，直接和x做仿射计算

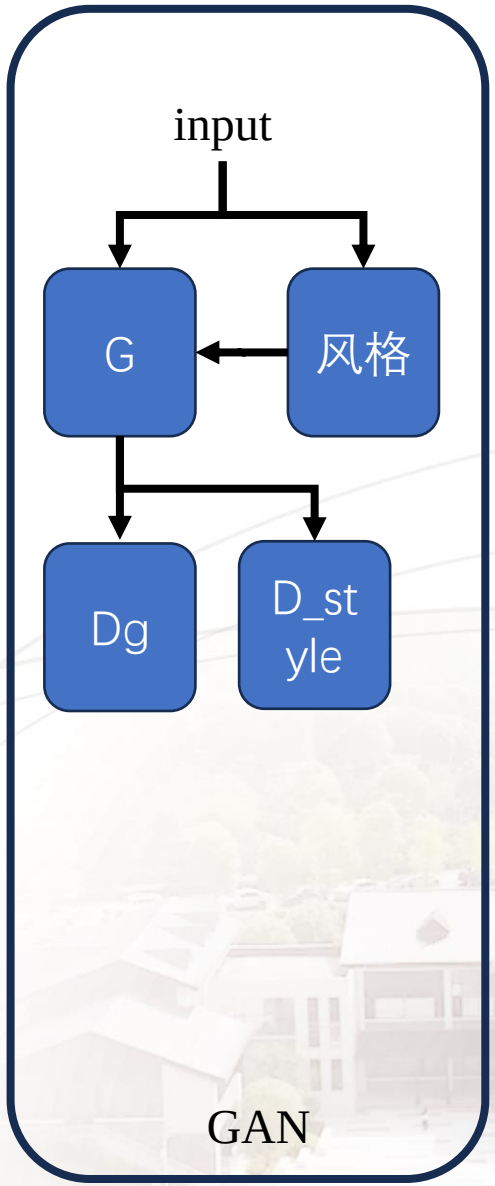
```
class AdaIN(nn.Module):
    def __init__(self, style_dim, num_features):
        super().__init__()
        self.norm = nn.InstanceNorm2d(num_features, affine=False)
        self.fc = nn.Linear(style_dim, num_features*2)

    def forward(self, x, s):
        h = self.fc(s)
        h = h.view(h.size(0), h.size(1), 1, 1)
        gamma, beta = torch.chunk(h, chunks=2, dim=1)
        return (1 + gamma) * self.norm(x) + beta
```

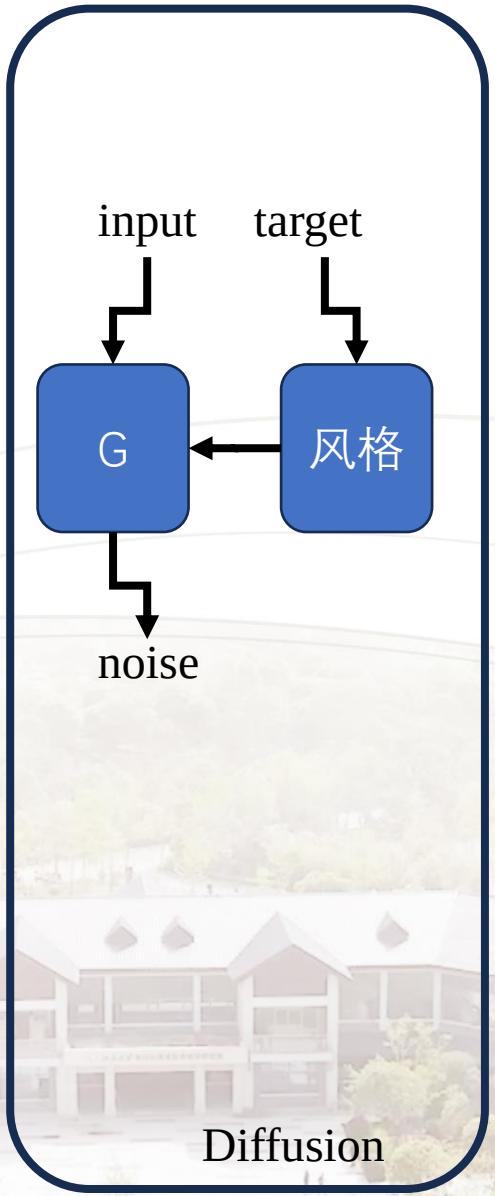
映射到【B,num *2 】,再拆成gamma,beta,于是他俩都是【B,num】,x是【B,num,H,W】则可以做乘加计算。

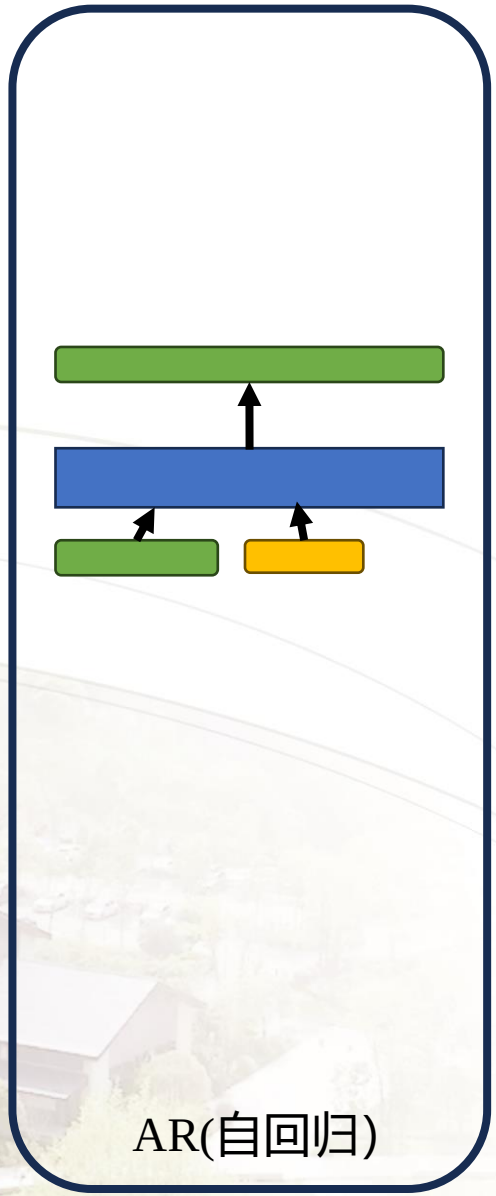
https://github.com/yl4579/StarGANv2-VC/blob/main/models.py

二、AdaIN的广泛应用





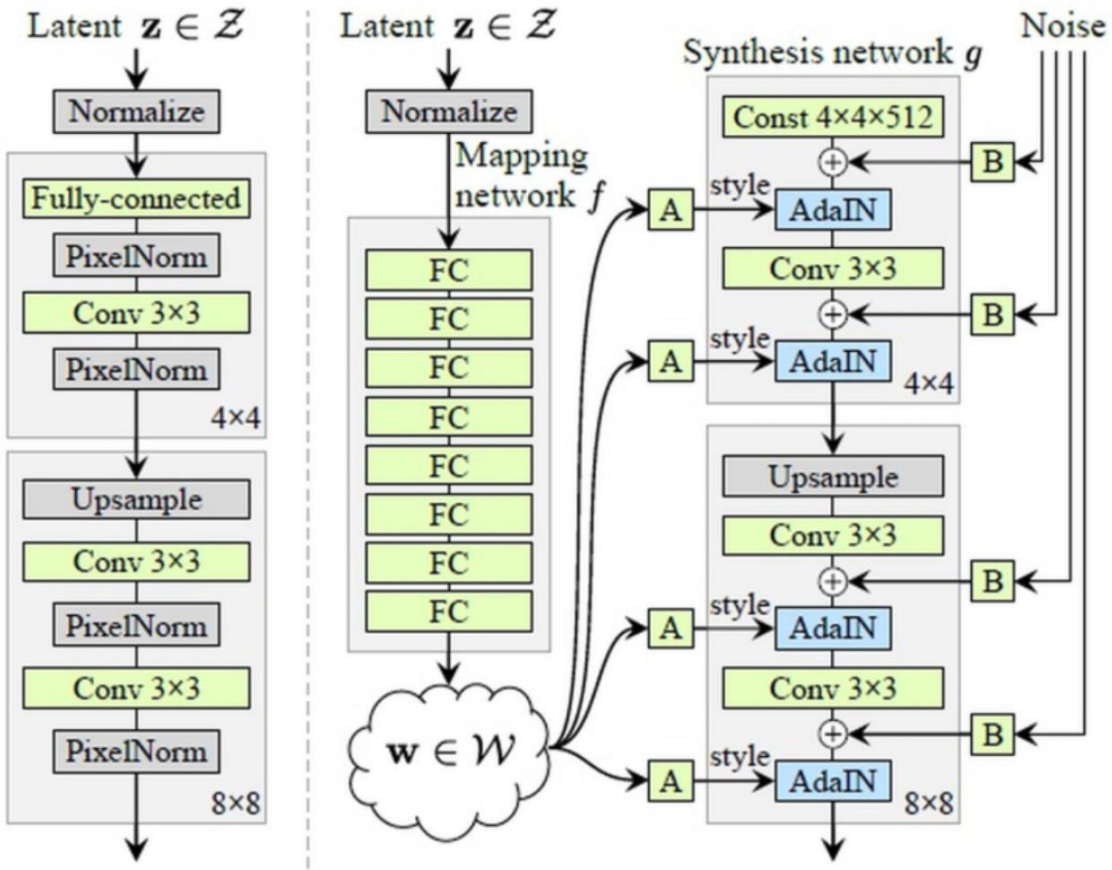




以上都是对数据x进行风格仿射计算！

二、AdaIN的经典应用:StyleGAN

按照我的习惯，我习惯从模型图开始入手理解，所以先来看一看stylegan的模型架构：



三、WadaIN：StyleGAN3

首次提出对神经网络权重w本身进行风格仿射计算。
且用卷积计算。而不是乘法和加法。

```
def forward(self, x, c_src, c_trg):

    batch_size, in_channel, t = x.size()

    #c = torch.cat([c_src, c_trg], dim = -1)

    s = self.style_linear(c_trg).view(batch_size, 1, in_channel, 1)
    beta = self.style_linear_beta(c_trg).view(batch_size, 1, in_channel, 1)
    # scale weights
    weight = self.scale * (self.weight * s + beta) # b out in ks

    # demodulate
    demod = torch.rsqrt(weight.pow(2).sum([2,3]) + 1e-8)
    demod_mean = torch.mean(weight.view(batch_size, self.dim_out, -1), dim = 2)
    weight = (weight - demod_mean.view(batch_size, self.dim_out, 1,1) )  * demod.view(batch_size, self.dim_out, 1,1)

    weight = weight.view(batch_size * self.dim_out, self.dim_in, self.kernel_size)

    x = x.view(1, batch_size * in_channel, t)

    out = F.conv1d(x, weight, padding = self.padding, groups = batch_size)

    _, _, new_t = out.size()

    out = out.view(batch_size, self.dim_out, new_t)
    out = self.glu(out)
```

（1）从c target 样本，提取风格向量s

（2）卷积核 风格化（scale modulate）

（3）卷积核的方差归零（demodulate）

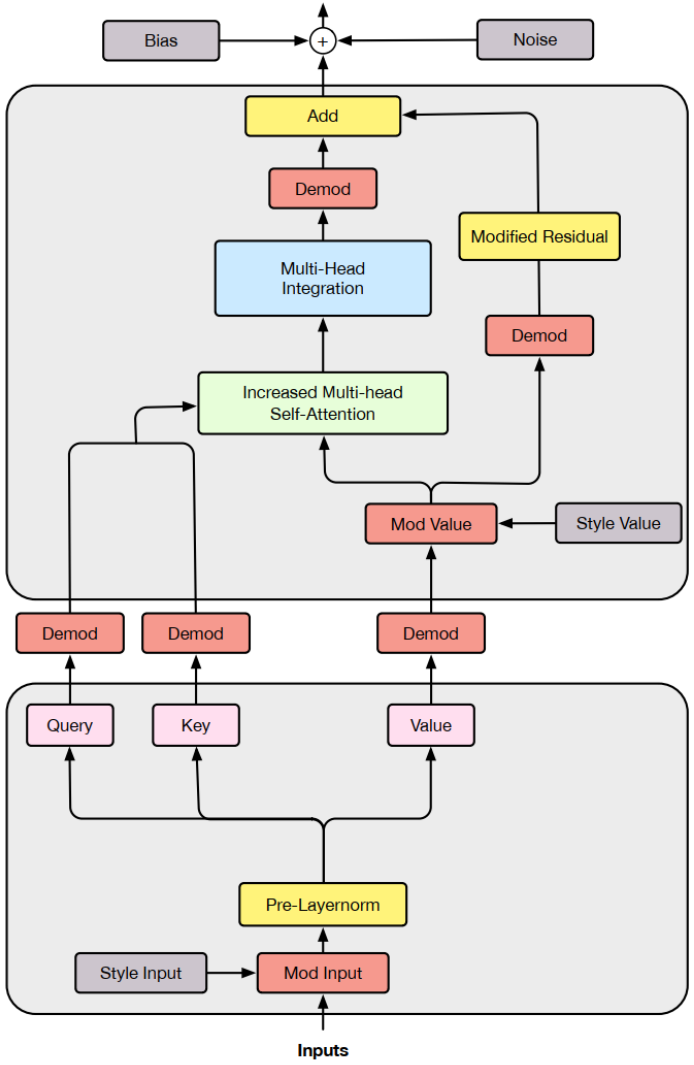
（4）卷积核对数据X进行卷积
图像数据就是2D conv
语音文本数据就是1D conv

https://github.com/MingjieChen/LowResourceVC/blob/master/stgan_adain/stylegan2_module.py

《TOWARDS LOW-RESOURCE STARGAN VOICE CONVERSION USING WEIGHT ADAPTIVE INSTANCE NORMALIZATION》

Styleformer: Transformer based Generative Adversarial Networks with Style Vector

Mod：将权重与风格s仿射计算
Demod：维护权重更新的稳定性



(b)

四、Styleformer


```
115
116         styles1 = styles[:, :hidden_dimension]
117         styles2 = styles[:, hidden_dimension:]
118
119
120         x = x * styles1.to(x.dtype).reshape(batch_size, 1, -1)
121         x = layernorm(x)
```

风格向量的cut
内容变量的归一化

```
q = q_weight.unsqueeze(0)
q = q * styles1.reshape(batch_size, 1, -1)
q_dcoefs = (q.square().sum(dim=[2]) + 1e-8).rsqrt()
```

Wq,Wk,Wv 三个权重矩阵的module , demod

```
q_value = torch.matmul(x, q_weight.t().to(x.dtype)) * q_dcoefs
```

计算得Q K V，自注意力计算。

```
skip = torch.matmul(skip, u_weight.t().to(x.dtype))
skip = skip * u_dcoefs.to(x.dtype).reshape(batch_size, 1, -1)

x = x + skip

return x
```

残差连接计算

https://github.com/Jeeseung-Park/Styleformer/blob/main/training/networks_Generator.py

四、Styleformer

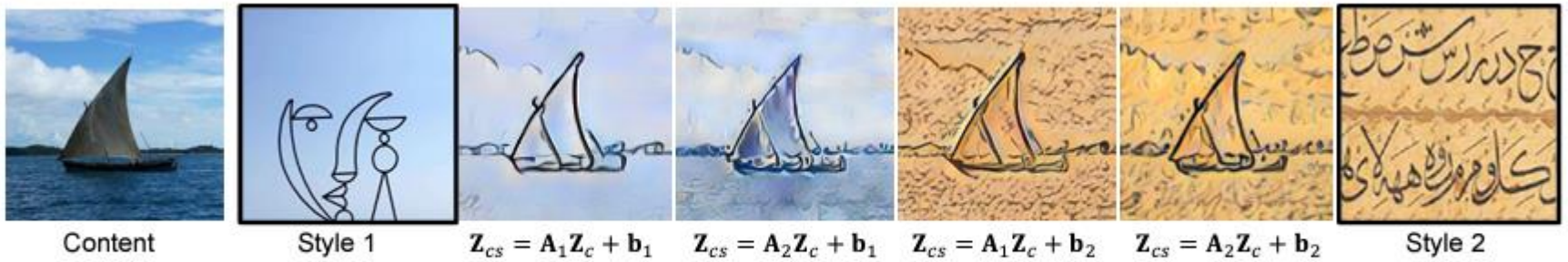
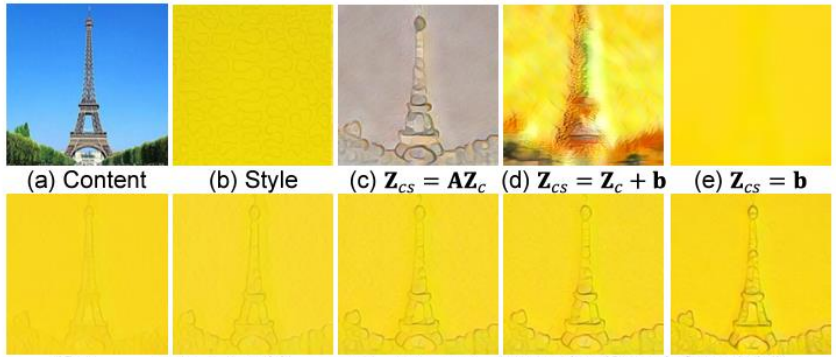


Figure 7. The exchange of the strokes and background of two style images. $\mathbf{Z}_{cs}$ denotes the stylized feature, $\mathbf{A}$ and $\mathbf{b}$ denote the affine transformation matrix and bias from style 1 or style 2 and $\mathbf{Z}_c$ denotes the content feature.



探讨在仿射计算中，Y = AZ + b 中 A，b的作用

结论：

1、参数A和b分别负责不同的风格信息：A主要包含细节风格（如笔触），而b主要控制整体风格（如背景）。

2、当移除b时，整体风格（例如背景颜色）会消失；当A被替换为单位矩阵时，输出图像中的细节笔触也会消失。

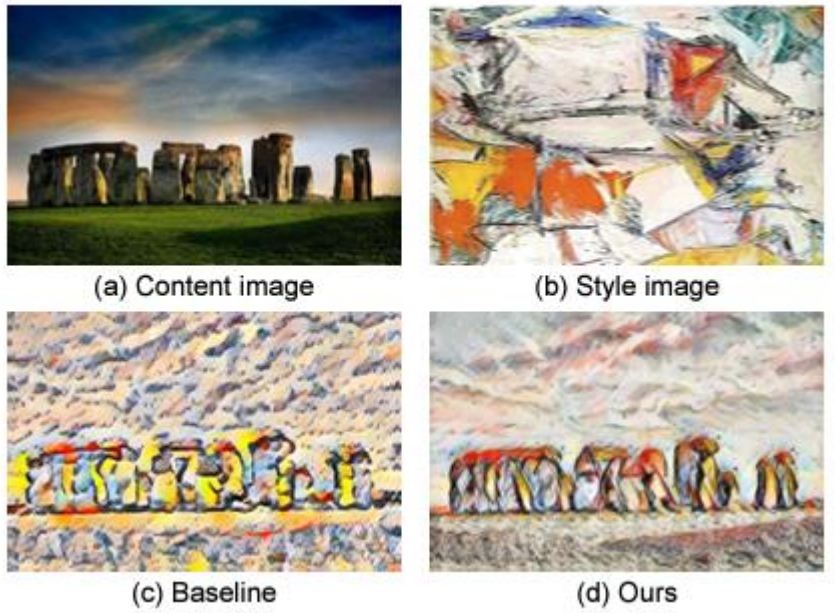
3、控制A的权重α时，随着α增大，图像边缘的笔触变得更强，且出现更多弯曲线条。

4、这种参数功能的解耦为多风格应用提供了更大的灵活性，允许交换不同风格图像的笔触和背景，实现更灵活的风格化效果。

四、Styleformer


```
[Casual]  I am quitting my job
[Formal]  I will be stepping down from my job.
-------------------------------------------------------------------
[Casual]  Jimmy is on crack and can't trust him
[Formal]  Jimmy is a crack addict I cannot trust him
-------------------------------------------------------------------
[Casual]  What do guys do to show that they like a gal?
[Formal]  What do guys do to demonstrate their affinity for women?
-------------------------------------------------------------------
[Casual]  i looooooooooooooooooooooove going to the movies.
[Formal]  I really like to go to the movies.
```

文本的说话风格迁移



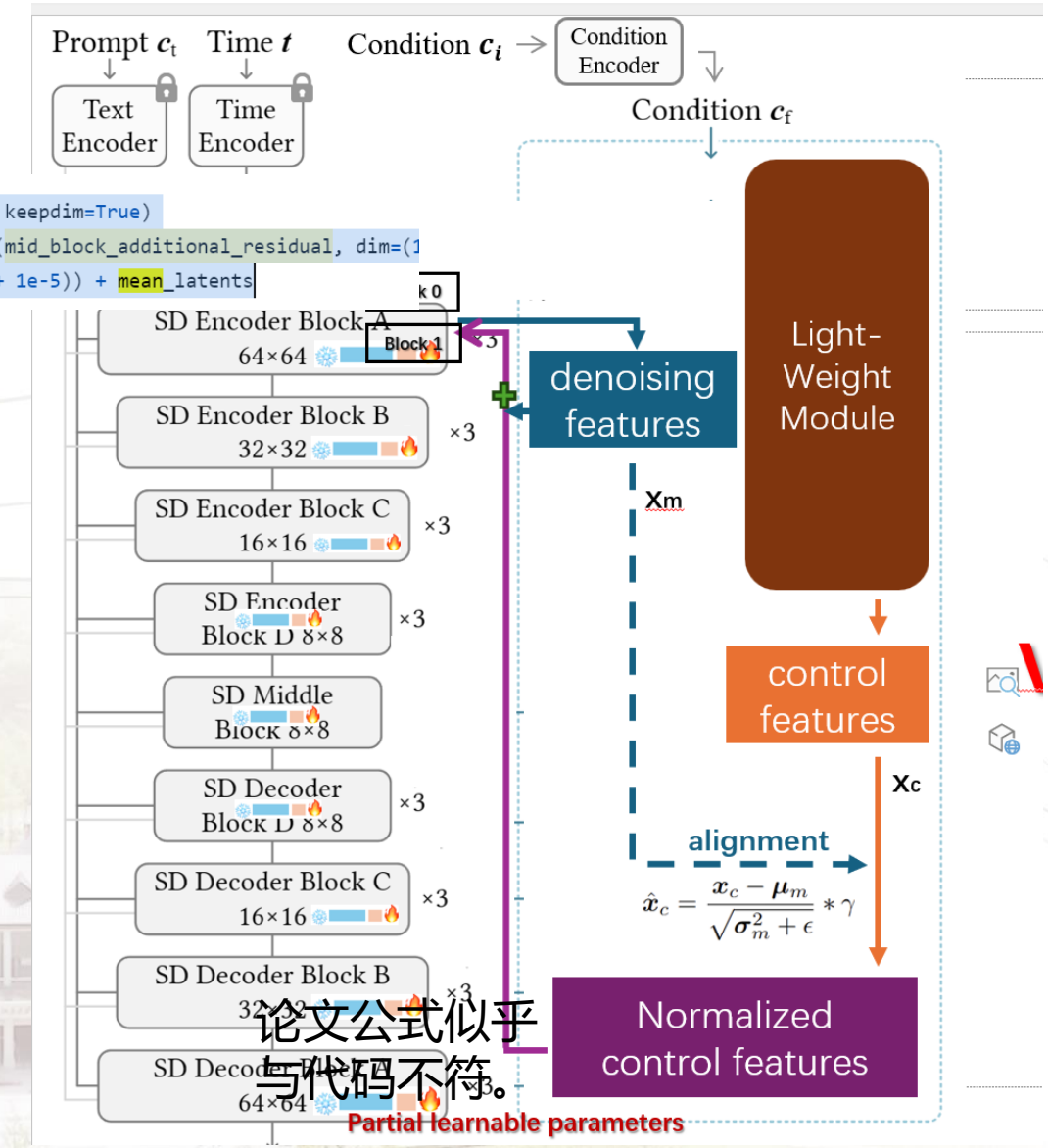
图像的风格迁移

五、ControlNeXt

Controlnet的风格迁移思想
一如既往。

```
mean_latents, std_latents = torch.mean(sample, dim=(1, 2, 3), keepdim=True), torch.std(sample, dim=(1, 2, 3), keepdim=True)
mean_control, std_control = torch.mean(mid_block_additional_residual, dim=(1, 2, 3), keepdim=True), torch.std(mid_block_additional_residual, dim=(1
mid_block_additional_residual = (mid_block_additional_residual - mean_control) * (std_latents / (std_control + 1e-5)) + mean_latents
```

Control变量来自于 Xc
Sample来自于 SD 模块,为Xm



论文公式似乎与代码不符。

将代码化为公式

3. **均值和标准差计算**：计算 `sample` 和 `conditional_controls` 的均值和标准差：

$$\mu_{\text{latents}} = \frac{1}{N}\sum_i \text{sample}_i$$

$$\sigma_{\text{latents}} = \sqrt{\frac{1}{N}\sum_i (\text{sample}_i - \mu_{\text{latents}})^2}$$

$$\mu_{\text{control}} = \frac{1}{M}\sum_i \text{conditional\_controls}_i$$

$$\sigma_{\text{control}} = \sqrt{\frac{1}{M}\sum_i (\text{conditional\_controls}_i - \mu_{\text{control}})^2}$$

Cross Normalization aligns the distributions of the denoising and control features, serving as a bridge to connect the diffusion and control branches. It accelerates the train-

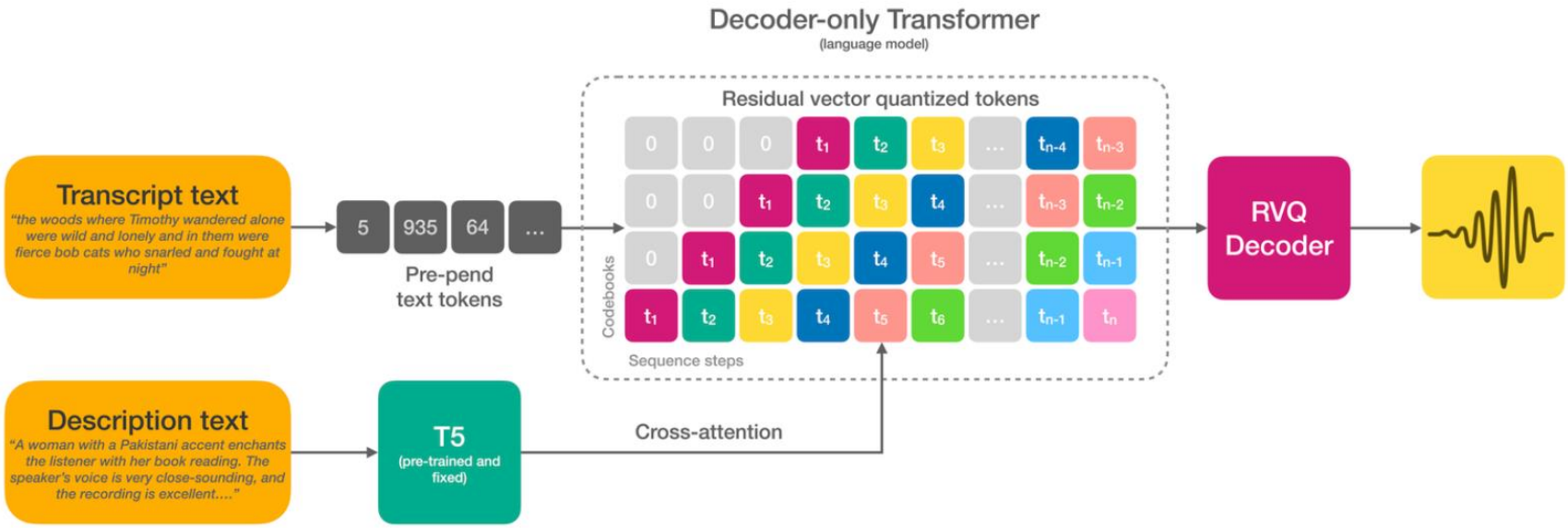
为了让额外添加的那组参数W_light能够获得强如SD模型本身的能力，就需要让输入给W_light的control的变量向SD的latents去靠近。

4. **归一化调整**：根据均值和标准差调整 `conditional_controls`：

$$\text{conditional\_controls} = (\text{conditional\_controls} - \mu_{\text{control}}) \cdot \frac{\sigma_{\text{latents}}}{\sigma_{\text{control}} + 1e-5} + \mu_{\text{latents}}$$

$$control = \frac{control - \mu_{control}}{\sigma_{control}} \cdot \sigma_{latents} + \mu_{latents}$$

六、LLM 风格零样本生成



六、总结

CycleGAN 系列

1、属于对偶学习+GAN。对偶学习的假设，有2个loss（循环一致性和身份一致性）。 只能进行2种风格之间的互相迁移。且不能添加风格迁移模块。

StarGAN 系列

训练原理基本同上，但可以添加风格迁移模块。有CIN ,AdaIN，WadaIN，发展出了不少论文和开源代码。

AdaIN系列

经典工作：语音的AdaIN-VC,StarGANv2-VC。图像的stylegan-1。特点是可以把数据样本本身当作风格控制去输入，于是灵活度很高。在大规模数据训练的情况下，也能模仿任意样本的风格。

WadaIN系列

语音的 WIN-VC, 图像的 stylegan-2和3

Styleformer

对自注意力矩阵进行风格迁移，且保证了图像的高质量

ControlNext