深入理解风格迁移三部曲

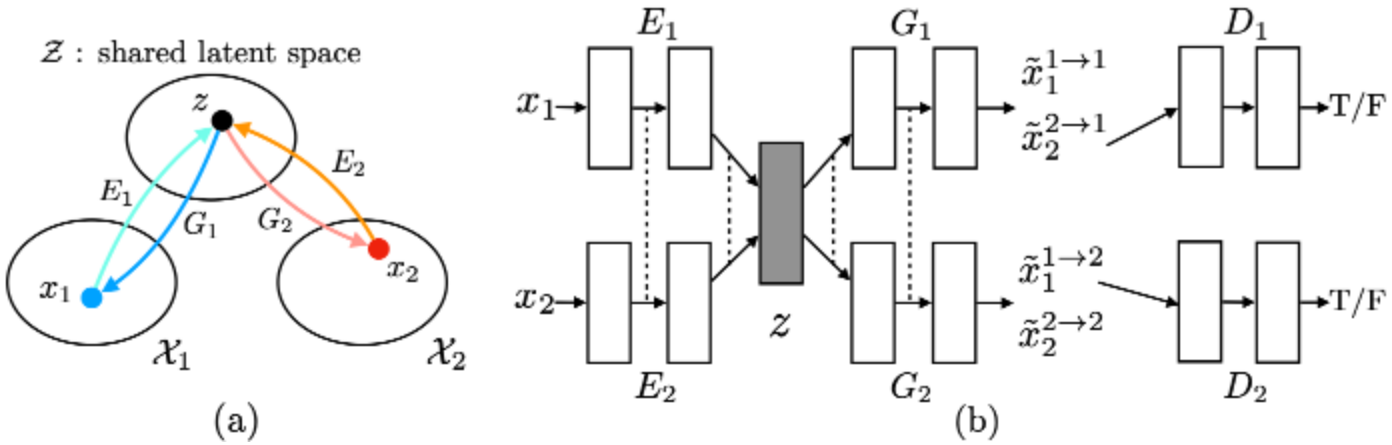
报告:陈扬

FUNIT

无监督的图像到图像转换方法学习利用图像的非结构化(UNlabel)数据集将给定类中的图像映射到不同类中的类似图像。在ICCV2019上,NVIDIA-Lab发表了Image-to-image最新的研究成果,基于少量类别学习的FUNIT.笔者在CVPR2020的投稿中正好也大量涉及到了image2image的实验,其中样本不均衡是对unpair-image2image任务来说同样是一个很大的问题,而few-shot Learning则是其中一种极端情况.于此同时,paper中令我影响深刻的是作者做了大量的Ablation Study ,充分的实验证明了算法的有效性以及鲁棒性.

前言

在我们展开深入理解FUNIT之前,我们来看一下他的前身UNIT,由NVIDIA-Lab在2017年提出,该文章首次提Image-Image Translation这个概念，将计算机视觉和计算机图形学的许多任务总结进去，分为一对多和多对一的两类转换任务，包括CV里的边缘检测，图像分割，语义标签以及CG里的mapping labels or sparse user inputs to realistic images.



该文章定义了χ_1和χ_2作为两个图像域.传统的supervised Image-to-image 通过对图像域进行采样,求其联合概率分布$P_{(\chi_1,\chi_2)}(x_1,x_2)$,通过Encoder-Decoder的思想,作者定义了两个E和G,希望使得z=E(X)在latent space上近可能的分布一致.意味着当我们同时对$Sample(\chi_1,\chi_2)$时,我们希望得出:

$$z = E_1^*(x_1) = E_2^*(x_2)$$

这样,我们得到了两个Domain下image的一致表示,再通过令$$G=D$$,从latent space中重构$\hat{x} = G(z)$,

因此,我们两个采样下的$\{x_1, x_2\}$经过$\{< E_1, G_1 >, < E_2, G_1 >, < E_1, G_2 >, < E_2, G_1 >\}$后得到了$\{\hat{x}_1^{1\to1}, \hat{x}_2^{2\to1}, \hat{x}_1^{1\to2}, \hat{x}_2^{2\to2}\}$,再把:

$$\hat{x}_1^{1\to1}, \hat{x}_2^{2\to1} \to D_1 \to T/F$$

$$\hat{x}_1^{1\to2}, \hat{x}_2^{2\to2} \to D_2 \to T/F$$

通过Adv_loss对抗学习跨域生成图片的效果.

可能细心的你以及发现了这是不是很类似VAE-GAN吗?是的.

作者通过联合训练4个网络$VAE_1, VAE_2, GAN_1, GAN_2$的三个$loss function$来训练整个网络:

$$\min_{E_1, E_2, G_1, G_2} \max_{D_1, D_2} \mathcal{L}_{\text{VAE}_1}(E_1, G_1) + \mathcal{L}_{\text{GAN}_1}(E_2, G_1, D_1) + \mathcal{L}_{\text{CC}_1}(E_1, G_1, E_2, G_2)$$
$$\mathcal{L}_{\text{VAE}_2}(E_2, G_2) + \mathcal{L}_{\text{GAN}_2}(E_1, G_2, D_2) + \mathcal{L}_{\text{CC}_2}(E_2, G_2, E_1, G_1)$$

训练好的网络,我们可以通过对latent sapce的latent variable重编码,进而把输入图像迁移到各个域中

VAE的目标是minimize source domain to latent space's KL diversity and latent space to destination domain's KL diversity(我觉得中文太拗口了,这句话实在是说不来)来最小化变分上界,VAE的定义如下:

$$\mathcal{L}_{\mathrm{VAE}_1}\left(E_1, G_1\right)=\lambda_1 \mathrm{KL}\left(q_1\left(z_1 | x_1\right) \| p_\eta(z)\right)-\lambda_2 \mathbb{E}_{z_1 \sim q_1\left(z_1 | x_1\right)}\left[\log p_{G_1}\left(x_1 | z_1\right)\right]$$
$$\mathcal{L}_{\mathrm{VAE}_2}\left(E_2, G_2\right)=\lambda_1 \mathrm{KL}\left(q_2\left(z_2 | x_2\right) \| p_\eta(z)\right)-\lambda_2 \mathbb{E}_{z_2 \sim q_2\left(z_2 | x_2\right)}\left[\log p_{G_2}\left(x_2 | z_2\right)\right]$$

对抗:GAN_LOSS被用于确保翻译图像类似图像在目标域.定义如下:

$$\mathcal{L}_{\mathrm{GAN}_1}\left(E_2, G_1, D_1\right) = \lambda_0 \mathbb{E}_{x_1 \sim P_{\mathcal{X}_1}}\left[\log D_1\left(x_1\right)\right] + \lambda_0 \mathbb{E}_{z_2 \sim q_2\left(z_2 \mid x_2\right)}\left[\log\left(1 - D_1\left(G_1\left(z_2\right)\right)\right)\right]$$
$$\mathcal{L}_{\mathrm{GAN}_2}\left(E_1, G_2, D_2\right) = \lambda_0 \mathbb{E}_{x_2 \sim P_{\mathcal{X}_2}}\left[\log D_2\left(x_2\right)\right] + \lambda_0 \mathbb{E}_{z_1 \sim q_1\left(z_1 \mid x_1\right)}\left[\log\left(1 - D_2\left(G_2\left(z_1\right)\right)\right)\right]$$

循环一致性:由于shared latent-space假设暗含了循环一致性约束，因此我们在提出的框架中实施循环一致性约束，以进一步规范不适定的无监督图像间转换问题。产生的信息处理流称为循环重建流,定义如下:

$$\mathcal{L}_{\mathrm{CC}_1}\left(E_1, G_1, E_2, G_2\right) = \lambda_3 \mathrm{KL}\left(q_1\left(z_1|x_1\right) \| p_\eta(z)\right) + \lambda_3 \mathrm{KL}\left(q_2|x_1^{1\to 2}\right) \| p_\eta(z)\big) - \lambda_4 \mathbb{E}_{z_2\sim q_2\left(z_2|x_1^{1\to 2}\right)}\left[\log p_{G_1}\left(x_1|z_2\right)\right]$$

$$\mathcal{L}_{\mathrm{CC}_2}\left(E_2, G_2, E_1, G_1\right) = \lambda_3 \mathrm{KL}\left(q_2\left(z_2|x_2\right) \| p_\eta(z)\right) + \lambda_3 \mathrm{KL}\left(q_1\left(z_1|x_2^{2\to 1}\right)\right) \| p_\eta(z)\big) - \lambda_4 \mathbb{E}_{z_1\sim q_1\left(z_1|x_2^{2}\to 1\right)}\left[\log p_{G_2}\left(x_2|z_1\right)\right]$$

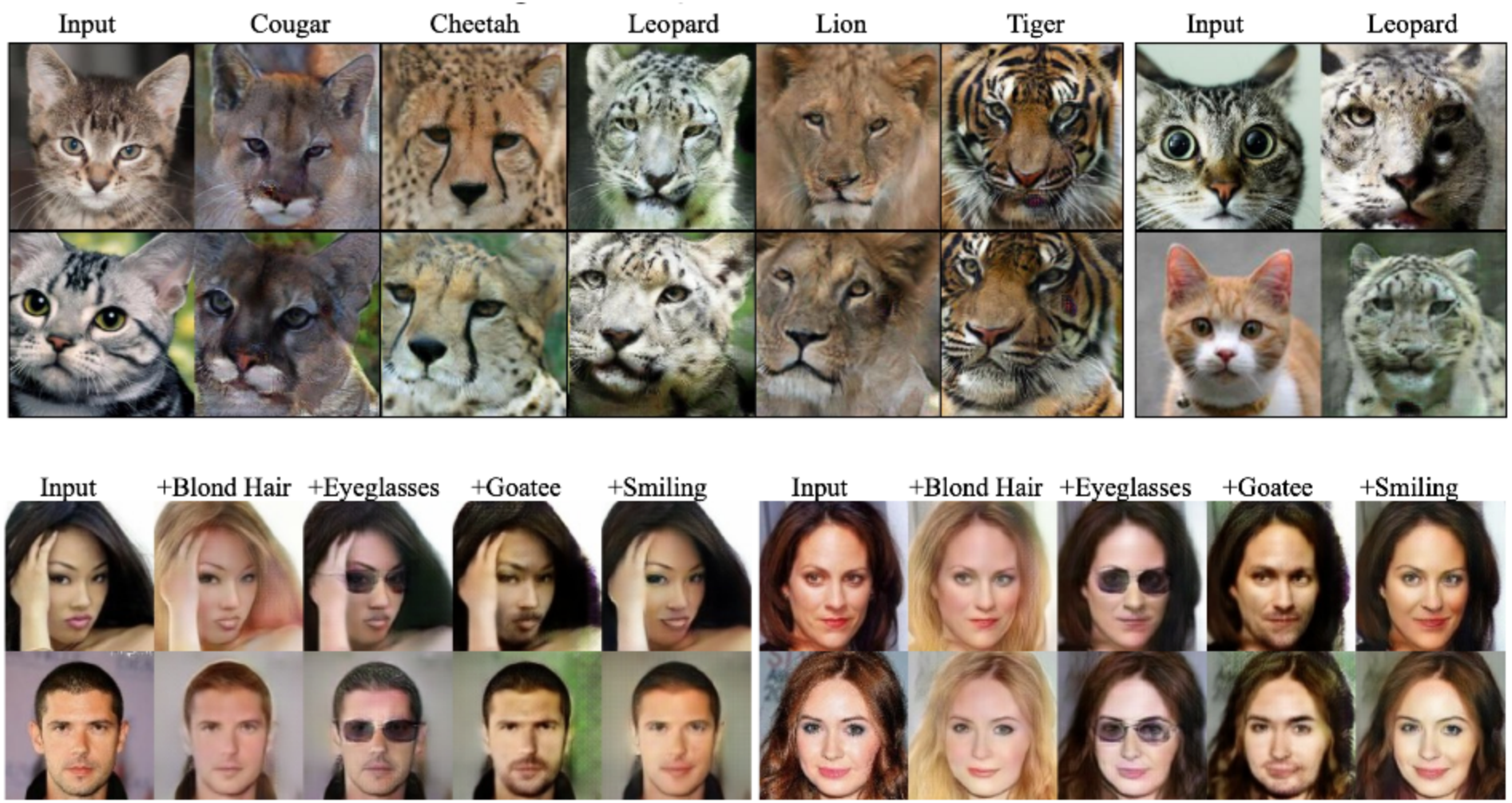
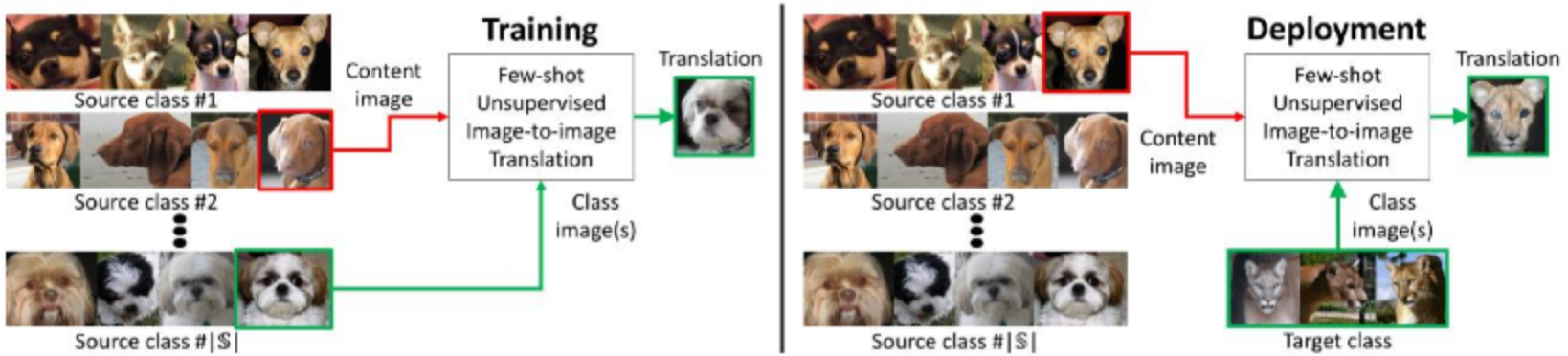


Figure 6: Attribute-based face translation results.

Few-Shot

虽然UNIT以及其变种(cycleGAN等等)已经表现得非常成功，现有的无监督图像到图像转换模型在两个方面受到限制。首先，如果在训练时只给出很少的图像，它们的样本效率低，产生差的转换输出。

其次，学习的模型仅限于在两个类之间转换图像。尽管新任务与原始任务之间存在相似性，但是用于一个转换任务的训练模型不能直接重用于新任务。



前提假设:为了训练 FUNIT，我们使用来自一组对象类（例如各种动物物种的图像）的图像，称为源类。我们不假设任何两个类别之间配对图像的存在.

使用源类图像来训练多级无监督图像到图像的转换模型。

在测试过程中，我们为模型提供了 一些来自新对象类的图像，称为目标类。该模型必须利用少 数目标图像将任何源类图像转换为目标类的同类图像。

框架由条件图像生成器 G 和多任务对抗判别器 D 组成。它采用一个图像作为输入，我们的生成 器 G 同时采用内容图像 x和一组类别图像K:$\{y_1, y_2 \dots y_k\}$作为输入,并通过生成器G产生输出图像:

$$\hat{x} = G(x, \{y_1, y_2, \dots y_k\})$$

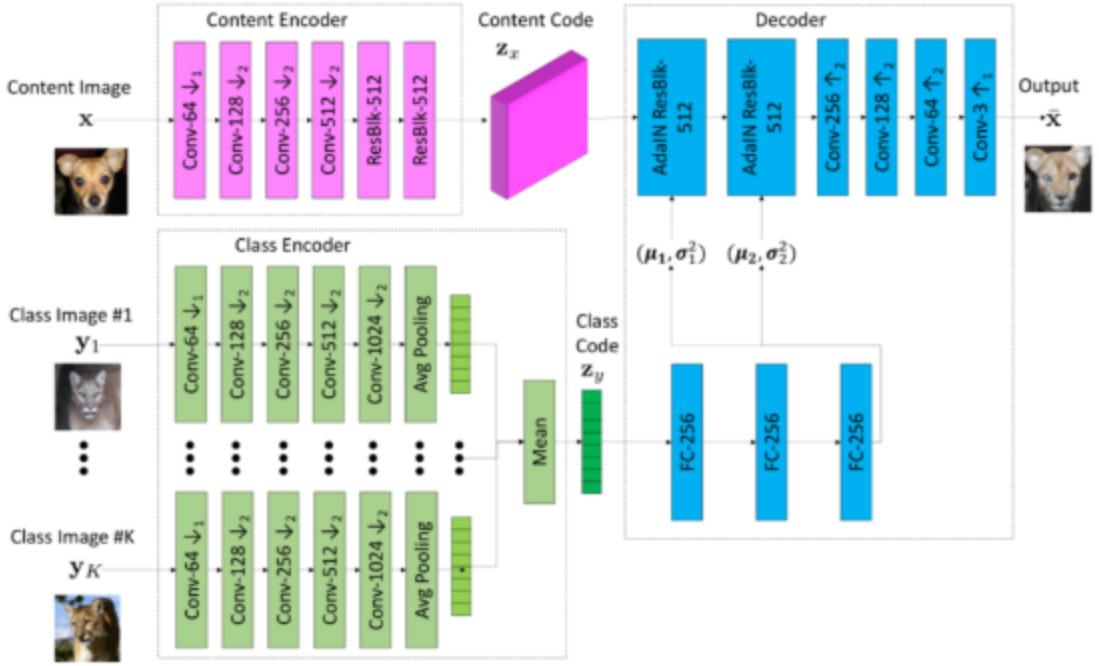
我们将 G 称为少样本图像转换器。G 将输入内容图像x映射到输出图像$\hat{x}$,使得 x 看起来像属于对象类c_y的图像，并且x和$\hat{x}$具有纹理结构上的相似性。设 S 和 T 分别表示源类集和目标类集。训练时,$c_x, c_y \in Sandc_x \neq c_y$ 在测试时，G 从未看到的目标类$c \in T$ 获取一些图像作为类图像，并将从任何源类采样的 图像映射到目标类 $c $ 的类似图像。

多任务对抗判别器

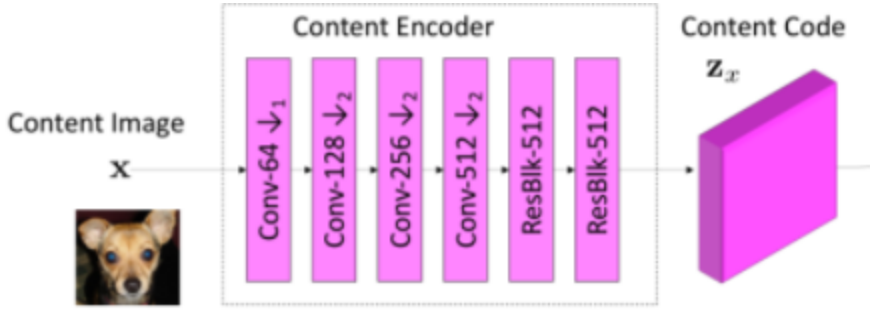
判别器 D 通过同时解决多个对抗分类任务来训练。 每个任务是二分类任务，确定输入图像是源类的实际图像还来自 G 生成的转换输出.这个对抗训练实际上是类似前面提到过的,不在赘述.

少样本图像转换器由三个子网络组成：内容编码器E_x(Content Encoder)，E_y类编码器(Class Encoder)和F_x解码器(Decoder).

$$\overline{\mathbf{x}} = F_x\left(\mathbf{z}_x, \mathbf{z}_y\right) = F_x\left(E_x(\mathbf{x}), E_y\left(\{\mathbf{y}_1, \ldots, \mathbf{y}_K\}\right)\right)$$



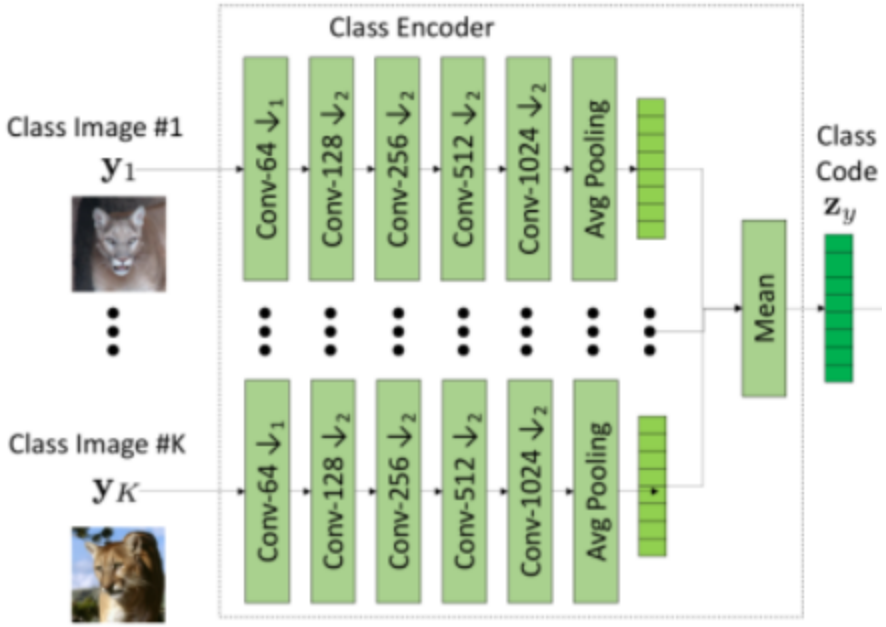
内容编码器由几个 2D 卷积层组成，后跟几个ResBlock。它将输入内容图像 x 映射到内容潜码z_x，其代表空间特征映射。



```
class ContentEncoder(nn.Module):
    def __init__(self, downs, n_res, input_dim, dim, norm, activ, pad_type):
        super(ContentEncoder, self).__init__()
        self.model = []
        self.model += [Conv2dBlock(input_dim, dim, 7, 1, 3,
                                   norm=norm,
                                   activation=activ,
                                   pad_type=pad_type)]
        for i in range(downs):
            self.model += [Conv2dBlock(dim, 2 * dim, 4, 2, 1,
                                       norm=norm,
                                       activation=activ,
                                       pad_type=pad_type)]
            dim *= 2
        self.model += [ResBlocks(n_res, dim,
                                 norm=norm,
                                 activation=activ,
                                 pad_type=pad_type)]
        self.model = nn.Sequential(*self.model)
        self.output_dim = dim

    def forward(self, x):
        return self.model(x)
```

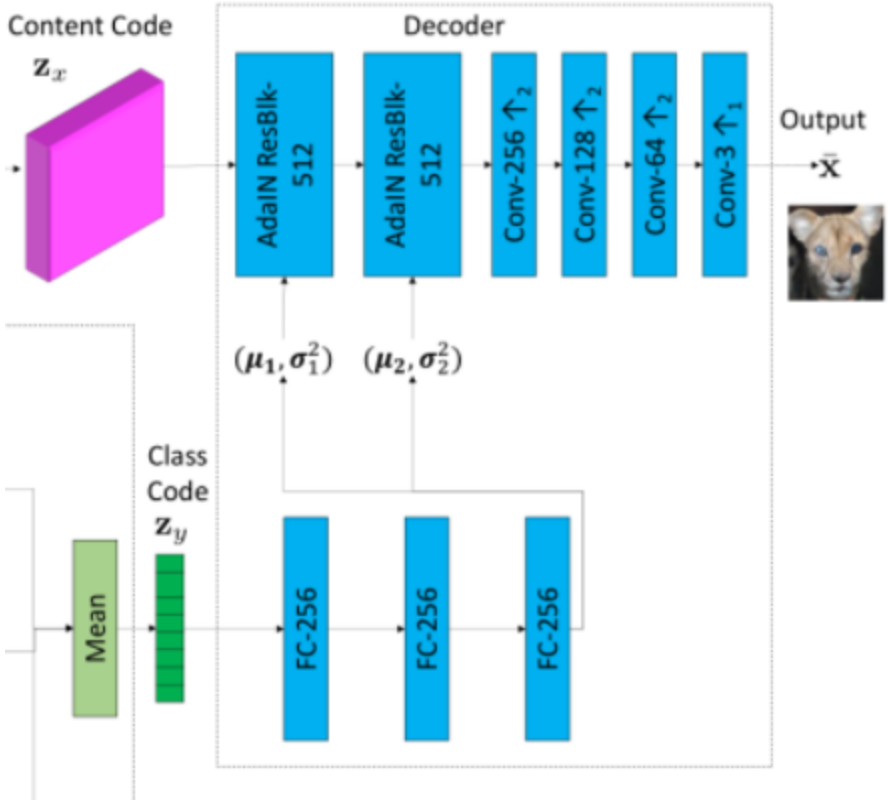
类编码器由几个 2D 卷积层组成，后面是样本轴平滑操作。具体地说，它首先映射 K个类别图像$\{y_1, y_2, \ldots y_k\}$映射到中间潜在向量,然后计算中间潜在向量的平均值，以获得最终的潜码z_y。



```
class ClassModelEncoder(nn.Module):
    def __init__(self, downs, ind_im, dim, latent_dim, norm, activ, pad_type):
        super(ClassModelEncoder, self).__init__()
        self.model = []
        self.model += [Conv2dBlock(ind_im, dim, 7, 1, 3,
                                   norm=norm,
                                   activation=activ,
                                   pad_type=pad_type)]
        for i in range(2):
            self.model += [Conv2dBlock(dim, 2 * dim, 4, 2, 1,
                                       norm=norm,
                                       activation=activ,
                                       pad_type=pad_type)]
            dim *= 2
        for i in range(downs - 2):
            self.model += [Conv2dBlock(dim, dim, 4, 2, 1,
                                       norm=norm,
                                       activation=activ,
                                       pad_type=pad_type)]
        self.model += [nn.AdaptiveAvgPool2d(1)]
        self.model += [nn.Conv2d(dim, latent_dim, 1, 1, 0)]
        self.model = nn.Sequential(*self.model)
        self.output_dim = dim

    def forward(self, x):
        return self.model(x)
```

解码器由几个自适应实例正规化（AdaIN)和残差块Resblock组成，后面跟着一些上 采样卷 积层。 AdaIN 残余 块是使用 AdaIN [18]作为正则化层的残余块。对于每个样本，AdaIN 首先将每个通道中样本的激活函数标准化为零均值和单位方差。然后它会缩放激活使用一组标量和偏置组成的学习仿射变换(通过两层全连接网络FC,使用自适应地计算仿射变换参数)。



```
class Decoder(nn.Module):
    def __init__(self, ups, n_res, dim, out_dim, res_norm, activ, pad_type):
        super(Decoder, self).__init__()

        self.model = []
        self.model += [ResBlocks(n_res, dim, res_norm,
                                 activ, pad_type=pad_type)]
        for i in range(ups):
            self.model += [nn.Upsample(scale_factor=2),
                           Conv2dBlock(dim, dim // 2, 5, 1, 2,
                                       norm='in',
                                       activation=activ,
                                       pad_type=pad_type)]
            dim //= 2
        self.model += [Conv2dBlock(dim, out_dim, 7, 1, 3,
                                   norm='none',
                                   activation='tanh',
                                   pad_type=pad_type)]
        self.model = nn.Sequential(*self.model)

    def forward(self, x):
        return self.model(x)

---
class MLP(nn.Module):
    def __init__(self, in_dim, out_dim, dim, n_blk, norm, activ):

        super(MLP, self).__init__()
        self.model = []
        self.model += [LinearBlock(in_dim, dim, norm=norm, activation=activ)]
        for i in range(n_blk - 2):
            self.model += [LinearBlock(dim, dim, norm=norm, activation=activ)]
        self.model += [LinearBlock(dim, out_dim,
                                   norm='none', activation='none')]
        self.model = nn.Sequential(*self.model)

    def forward(self, x):
        return self.model(x.view(x.size(0), -1))
```

生成器整体代码:

```
class FewShotGen(nn.Module):
    def __init__(self, hp):
        super(FewShotGen, self).__init__()
        nf = hp['nf']
        nf_mlp = hp['nf_mlp']
        down_class = hp['n_downs_class']
        down_content = hp['n_downs_content']
        n_mlp_blks = hp['n_mlp_blks']
        n_res_blks = hp['n_res_blks']
        latent_dim = hp['latent_dim']
        self.enc_class_model = ClassModelEncoder(down_class,
                                                 3,
                                                 nf,
                                                 latent_dim,
                                                 norm='none',
                                                 activ='relu',
                                                 pad_type='reflect')

        self.enc_content = ContentEncoder(down_content,
                                          n_res_blks,
                                          3,
                                          nf,
                                          'in',
                                          activ='relu',
                                          pad_type='reflect')

        self.dec = Decoder(down_content,
                           n_res_blks,
                           self.enc_content.output_dim,
                           3,
                           res_norm='adain',
                           activ='relu',
                           pad_type='reflect')

        self.mlp = MLP(latent_dim,
                       get_num_adain_params(self.dec),
                       nf_mlp,
                       n_mlp_blks,
                       norm='none',
                       activ='relu')

    def forward(self, one_image, model_set):
        # reconstruct an image
        content, model_codes = self.encode(one_image, model_set)
        model_code = torch.mean(model_codes, dim=0).unsqueeze(0)
        images_trans = self.decode(content, model_code)
        return images_trans

    def encode(self, one_image, model_set):
        # extract content code from the input image
        content = self.enc_content(one_image)
        # extract model code from the images in the model set
        class_codes = self.enc_class_model(model_set)
        class_code = torch.mean(class_codes, dim=0).unsqueeze(0)
        return content, class_code

    def decode(self, content, model_code):
        # decode content and style codes to an image
        adain_params = self.mlp(model_code)
        assign_adain_params(adain_params, self.dec)
        images = self.dec(content)
        return images
```

损失函数的设计

我们通过解决由下式给出的极小极大优化问题来训练所提 出的 FUNIT 框架：

$$\min_{D} \max_{G} \mathcal{L}_{\mathrm{GAN}}(D, G) + \lambda_{\mathrm{R}} \mathcal{L}_{\mathrm{R}}(G) + \lambda_{\mathrm{F}} \mathcal{L}_{\mathrm{FM}}(G)$$

其中L_{GAN}, L_R和L_F分别是 GAN 损失，内容图像重建损失和特征匹配损失。

GAN**对抗**损失仅使用类的相应二分类预测分数来计算损失。

$$\begin{aligned}\mathcal{L}_{\mathrm{GAN}}(G, D) =& E_{\mathrm{x}}\left[-\log D^{c_x}(\mathrm{x})\right]+ \\ & E_{\mathrm{x},\{\mathrm{y}_1,\ldots,\mathrm{y}_K\}}\left[\log\left(1-D^{c_y}(\overline{\mathrm{x}})\right)\right]\end{aligned}$$

L_R**内容重建**损失有助于 G 学习转换模型。 具体地，当对输入内容图像和输入类图像使用相同图像时（在这种情况下 K = 1），损失促使 G 生成与输入相同的输出图像(重构一致性,在cycleGAN在叫identity Loss).

$$\mathcal{L}_{\mathrm{R}}(G) = E_{\mathrm{x}}\left[\|\mathrm{x} - G(\mathrm{x}, \{\mathrm{x}\})\|_1^1\right]$$

L_F**特征匹配**损失使训练正常化。 我们首先通过从 D 重新移 动最后一个（预测）层来构造一个特征提取器，称为D_f 。 然后我们使用 D_f从转换输出 x 和 类图像 $\{y_1, y_2 \ldots y_k\}$ 中提取特征并最小化 :

$$\mathcal{L}_{\mathrm{F}}(G)=E_{\mathrm{x},\{\mathrm{y}_1,\ldots,\mathrm{y}_K\}}\left[D_f(\overline{\mathrm{x}}))-\sum_k \frac{D_f\left(\mathrm{y}_k\right)}{K}\|_1^1\right]$$

实验部分

我非常想重点讲一下实验部分,我看完这篇文章的实验部分做的太好了,给了我在今后的科研中一个非常好的榜样.

重点说一下评价指标啊,在image-to-image这个领域据我所知,肉眼观察法是最好的评价指标,但是他也会带来一些个人的主观性,我们看看作者是如何通过实验说服我的:

性能指标。作者使用几个标准进行评估。首先，作者测量转换是否类似于目标类的图像。其次，作者检查在转换期间是否保留了类不变内容。第三，作者量化输出图像的写实照片。最后，作者测量该模型是否可用于生成目标类的图像分布。

	Setting	Top1-all ↑	Top5-all ↑	Top1-test ↑	Top5-test ↑	DIPD ↓	IS-all ↑	IS-test ↑	mFID ↓
Animal Faces	CycleGAN-Unfair-20	28.97	47.88	38.32	71.82	1.615	10.48	7.43	197.13
	UNIT-Unfair-20	22.78	43.55	35.73	70.89	1.504	12.14	6.86	197.13
	MUNIT-Unfair-20	38.61	62.94	53.90	84.00	1.700	10.20	7.59	158.93
	StarGAN-Unfair-1	2.56	10.50	9.07	32.55	1.311	10.49	5.17	201.58
	StarGAN-Unfair-5	12.99	35.56	25.40	60.64	1.514	7.46	6.10	204.05
	StarGAN-Unfair-10	20.26	45.51	30.26	68.78	1.559	7.39	5.83	208.60
	StarGAN-Unfair-15	20.47	46.46	34.90	71.11	1.558	7.20	5.58	204.13
	StarGAN-Unfair-20	24.71	48.92	35.23	73.75	1.549	8.57	6.21	198.07
	StarGAN-Fair-1	0.56	3.46	4.41	20.03	1.368	7.83	3.71	228.74
	StarGAN-Fair-5	0.60	3.56	4.38	20.12	1.368	7.80	3.72	235.66
	StarGAN-Fair-10	0.60	3.40	4.30	20.00	1.368	7.84	3.71	241.77
	StarGAN-Fair-15	0.62	3.49	4.28	20.24	1.368	7.82	3.72	228.42
	StarGAN-Fair-20	0.62	3.45	4.41	20.00	1.368	7.83	3.72	228.57
	FUNIT-1	17.07	54.11	46.72	82.36	1.364	22.18	10.04	93.03
	FUNIT-5	33.29	78.19	68.68	96.05	1.320	22.56	13.33	70.24
	FUNIT-10	37.00	82.20	72.18	97.37	1.311	22.49	14.12	67.35
	FUNIT-15	38.83	83.57	73.45	97.77	1.308	22.41	14.55	66.58
	FUNIT-20	**39.10**	**84.39**	**73.69**	**97.96**	**1.307**	**22.54**	**14.82**	**66.14**
North American Birds	CycleGAN-Unfair-20	9.24	22.37	19.46	42.56	1.488	25.28	7.11	215.30
	UNIT-Unfair-20	7.01	18.31	16.66	37.14	1.417	28.28	7.57	203.83
	MUNIT-Unfair-20	23.12	41.41	38.76	62.71	1.656	24.76	9.66	198.55
	StarGAN-Unfair-1	0.92	3.83	3.98	13.73	1.491	14.80	4.10	266.26
	StarGAN-Unfair-5	2.54	8.94	8.82	23.98	1.574	13.84	4.21	270.12
	StarGAN-Unfair-10	4.26	13.28	12.03	32.02	1.571	15.03	4.09	278.94
	StarGAN-Unfair-15	3.70	11.74	12.90	31.62	1.509	18.61	5.25	252.80
	StarGAN-Unfair-20	5.38	16.02	13.95	33.96	1.544	18.94	5.24	260.04
	StarGAN-Fair-1	0.24	1.17	0.97	4.84	1.423	13.73	4.83	244.65
	StarGAN-Fair-5	0.22	1.07	1.00	4.86	1.423	13.72	4.82	244.40
	StarGAN-Fair-10	0.24	1.13	1.03	4.90	1.423	13.72	4.83	244.55
	StarGAN-Fair-15	0.23	1.05	1.04	4.90	1.423	13.72	4.81	244.80
	StarGAN-Fair-20	0.23	1.08	1.00	4.86	1.423	13.75	4.82	244.71
	FUNIT-1	11.17	34.38	30.86	60.19	1.342	67.17	17.16	113.53
	FUNIT-5	20.24	51.61	45.40	75.75	1.296	74.81	22.37	99.72
	FUNIT-10	22.45	54.89	48.24	77.66	1.289	75.40	23.60	98.75
	FUNIT-15	23.18	55.63	49.01	78.70	1.287	**76.44**	23.86	98.16
	FUNIT-20	**23.50**	**56.37**	**49.81**	**78.89**	**1.286**	76.42	**24.00**	**97.94**

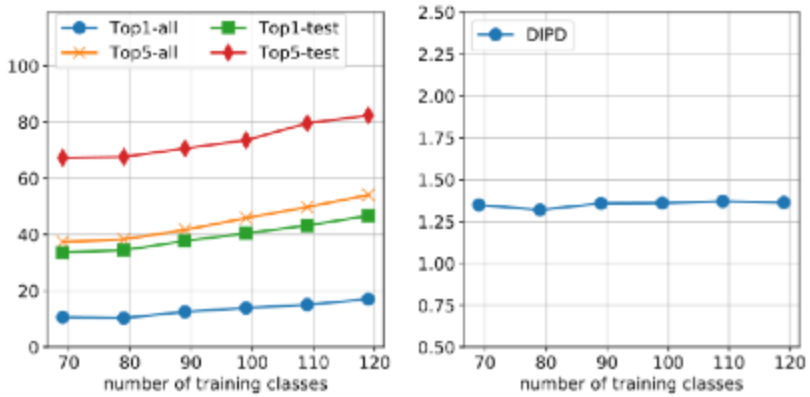
Table 1. Performance comparison with the fair and unfair baselines. ↑ means larger numbers are better, ↓ means smaller numbers are better.

量化对比:User performance score(肉眼观察法):

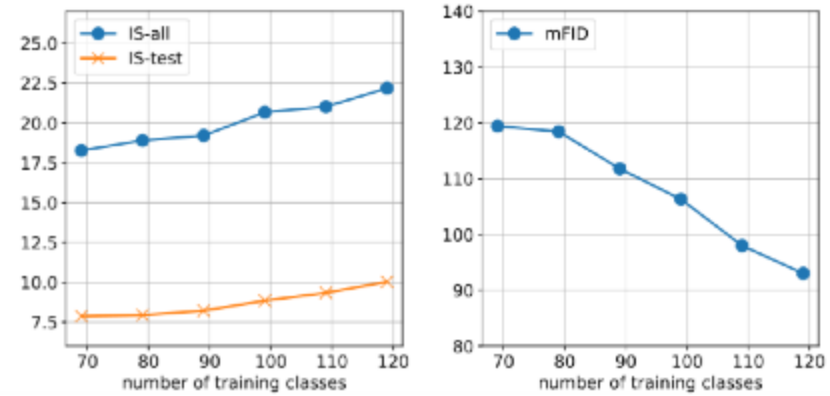
Setting	Animal	Birds
FUNIT-5 vs. StarGAN-Fair-5	86.08	82.56
FUNIT-5 vs. StarGAN-Unfair-20	86.00	84.48
FUNIT-5 vs. CycleGAN-Unfair-20	71.68	77.76
FUNIT-5 vs. UNIT-Unfair-20	77.84	77.96
FUNIT-5 vs. MUNIT-Unfair-20	83.56	79.64

少样本分类准确度:

# of generated samples N	Animal Face		North American Birds	
	S&H [15]	FUNIT	S&H [15]	FUNIT
0	38.76		30.38	
10	40.51	**42.05**	31.77	**33.41**
50	40.24	**42.22**	31.66	**33.64**
100	40.76	**42.14**	32.12	**34.39**



Inception Score(我很困惑这个指标真的有用吗?)和FID(这个确实还有点用):



Ablation Study

we analyze impact of the content image reconstruction loss weight on the Animal Faces dataset.The table shows that λR = 0.1 provides a good trade-off, and we used it as the default value throughout the paper. Interestingly, a very small weight value λR = 0.01 results in degrading performance on both
content preservation and translation accuracy.

Setting	Top1-all ↑	Top5-all ↑	Top1-test ↑	Top5-test ↑	DIPD ↓	IS-all ↑	IS-test ↑	mFID ↓
$\lambda_R = 0.01$	16.02	52.30	45.52	81.68	1.370	21.80	9.73	94.98
$\lambda_R = 0.1$	**17.07**	**54.11**	**46.72**	**82.36**	1.364	22.18	**10.04**	**93.03**
$\lambda_R = 1$	16.60	52.05	45.62	81.77	1.346	**22.21**	9.81	94.23
$\lambda_R = 10$	13.04	44.32	39.06	75.81	**1.298**	20.48	8.90	108.71

Table 4. Parameter sensitivity analysis on the content image reconstruction loss weight, λ_R. ↑ means larger numbers are better, ↓ means smaller numbers are better. The value of 0.1 provides a good trade-off between content preservation and translation accuracy, which is used as the default value throughout the paper. We use the `FUNIT-1` model for this experiment.

Method	# of generated Samples	Split 1	2	3	4	5	Average Accuracy
Baseline	0	38.81 ± 0.01	41.99 ± 0.03	39.13 ± 0.01	37.05 ± 0.02	36.82 ± 0.01	38.76
FUNIT	10	41.20 ± 0.41	46.25 ± 0.27	42.65 ± 0.41	40.75 ± 0.20	39.39 ± 0.31	42.05
	50	41.24 ± 0.16	46.27 ± 0.07	43.15 ± 0.06	41.01 ± 0.19	39.43 ± 0.09	42.22
	100	41.01 ± 0.18	46.72 ± 0.05	42.89 ± 0.09	40.73 ± 0.20	39.33 ± 0.04	42.14
S&H [15]	10	39.87 ± 0.47	42.69 ± 0.34	41.42 ± 0.39	39.95 ± 0.58	38.64 ± 0.42	40.51
	50	39.93 ± 0.15	42.62 ± 0.28	40.89 ± 0.09	39.31 ± 0.17	38.44 ± 0.13	40.24
	100	40.05 ± 0.31	41.72 ± 0.19	41.29 ± 0.16	41.33 ± 0.21	39.39 ± 0.16	40.76

Table 6. One-shot accuracies on the 5 splits of the Animal Faces dataset when using generated images and 1 real image. The average accuracy over 5 independent runs is reported per split (different set of generated images is sampled each time).

Method	# of generated Samples	Split 1	2	3	4	5	Average Accuracy
Baseline	0	30.71 ± 0.02	29.04 ± 0.01	31.93 ± 0.01	29.59 ± 0.01	30.64 ± 0.02	30.38
FUNIT	10	32.94 ± 0.49	33.29 ± 0.25	35.15 ± 0.22	31.20 ± 0.20	34.48 ± 0.58	33.41
	50	32.92 ± 0.34	33.78 ± 0.25	35.04 ± 0.10	31.80 ± 0.14	34.66 ± 0.17	33.64
	100	33.83 ± 0.16	33.99 ± 0.12	36.05 ± 0.14	32.01 ± 0.10	36.09 ± 0.19	34.39
S&H [15]	10	30.55 ± 0.11	31.96 ± 0.30	34.18 ± 0.19	30.65 ± 0.09	31.49 ± 0.24	31.77
	50	31.39 ± 0.07	30.59 ± 0.11	33.60 ± 0.05	30.92 ± 0.20	31.81 ± 0.18	31.66
	100	30.83 ± 0.10	32.03 ± 0.09	34.39 ± 0.17	31.12 ± 0.10	32.23 ± 0.15	32.12

Table 7. One-shot accuracies on the 5 splits of the North Amercian Birds dataset when using generated images and 1 real image. The average accuracy over 5 independent runs is reported per split (different set of generated images is sampled each time).

LatentSpaceInterpolation

weuset-SNE tovisualizetheclasscodeinatwodimensionalspace.Itcanbeseenthat imagesfromsimilarclassesaregroupedtogetherintheclassembeddingspace.

Figure 11. 2-D representation of the class code using t-SNE for 5000 images across 50 source classes. Please zoom-in for details.

THNKS
Q&A