Pix2pix 与 CycleGAN 的代码复现

21051923 邵屹杰

- 任务要求：精读给定的 Pix2pix 和一个 CycleGAN 代码，运行后，分析实验结果，写出对代码的理解。

一、 Pix2pix

- 具体代码理解

```
def resize (input_image, real_image, height, width):
  input_image = tf.image.resize(input_image, [height, width],
                                method=tf.image.ResizeMethod.NEAREST_NE
IGHBOR)
  real_image = tf.image.resize(real_image, [height, width],
                               method=tf.image.ResizeMethod.NEAREST_NEI
GHBOR)

  return input_image, real_image
```

使用最近邻插值方法调整图片大小

```
def random_crop (input_image, real_image):
  stacked_image = tf.stack([input_image, real_image], axis=0)
  cropped_image = tf.image.random_crop(
      stacked_image, size= [2, IMG_HEIGHT, IMG_WIDTH, 3])

  return cropped_image[0], cropped_image[1]
```

函数用于数据增强中的随机裁剪操作，通过在训练过程中随机裁剪图像，可以增加数据的多样性，提高模型的鲁棒性和泛化能力。

```
@tf.function()
def random_jitter(input_image, real_image):
  input_image, real_image = resize(input_image, real_image, 286, 286)
  input_image, real_image = random_crop(input_image, real_image)

  if tf.random.uniform(()) > 0.5:
    input_image = tf.image.flip_left_right(input_image)
    real_image = tf.image.flip_left_right(real_image)

  return input_image, real_image
```

该函数可用于数据增强中的随机扭曲操作，通过调整尺寸、随机裁剪和随机镜像等操作，增加训练数据的多样性，提高模型的泛化能力。由于使用了@tf.function()装饰器，函数可以直接在 TensorFlow 图模式下执行，从而加速运行速度。

```
def downsample(filters, size, apply_batchnorm=True):
```

```
  initializer = tf.random_normal_initializer(0., 0.02)

  result = tf.keras.Sequential()
  result.add(
      tf.keras.layers.Conv2D(filters, size, strides=2, padding='same',
                                        kernel_initializer=initializer,
use_bias=False))

  if apply_batchnorm:
    result.add(tf.keras.layers.BatchNormalization())

  result.add(tf.keras.layers.LeakyReLU())

  return result
```

该函数用于构建卷积层、批归一化层和 LeakyReLU 激活函数的组合，用于下采样操作，即编码器。

```
def upsample(filters, size, apply_dropout=False):
  initializer = tf.random_normal_initializer(0., 0.02)

  result = tf.keras.Sequential()
  result.add(
    tf.keras.layers.Conv2DTranspose(filters, size, strides=2,
                                    padding='same',
                                    kernel_initializer=initializer,
                                    use_bias=False))

  result.add(tf.keras.layers.BatchNormalization())

  if apply_dropout:
      result.add(tf.keras.layers.Dropout(0.5))

  result.add(tf.keras.layers.ReLU())

  return result
```

该函数用于构建反卷积层、批归一化层、Dropout 层和 ReLU 激活函数的组合，用于上采样操作，即解码器。

```
def Generator():
…
```

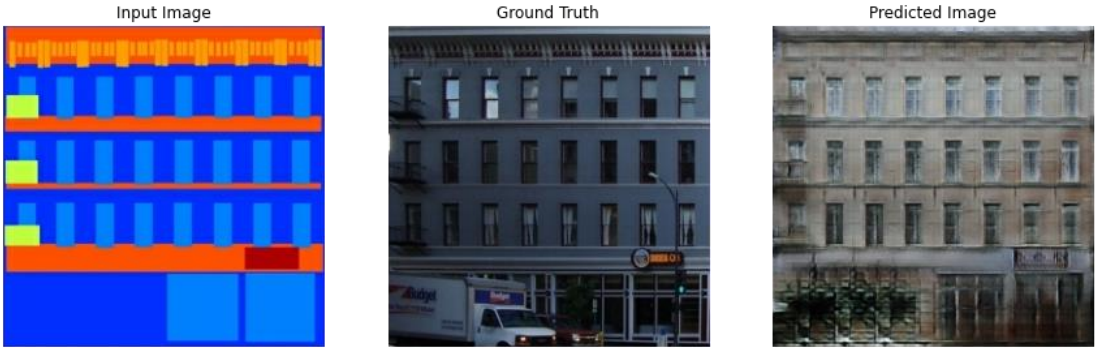
该生成器模型通过一系列下采样和上采样操作，逐渐从输入图像生成目标大小的合成图像。跳跃连接用于传递低层级的细节信息，有助于生成更加真实和细致的图像。

```
def Discriminator():
```

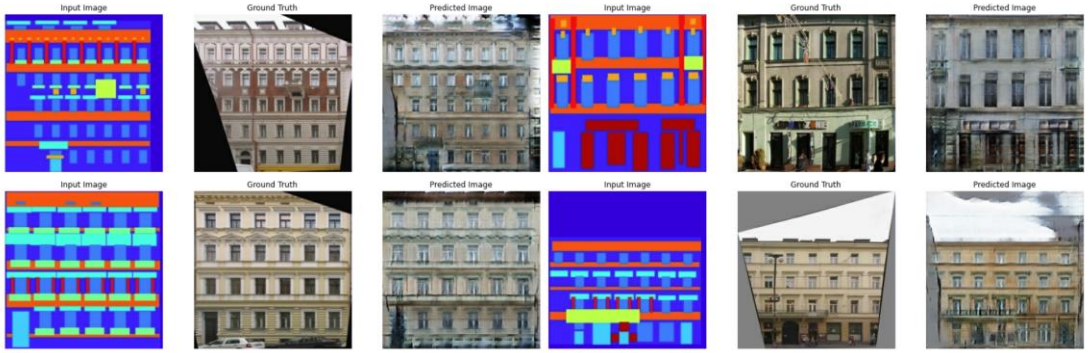
…
定义了一个判别器模型，用于区分生成器生成的合成图像和真实图像。

- 参数设置与结果展示

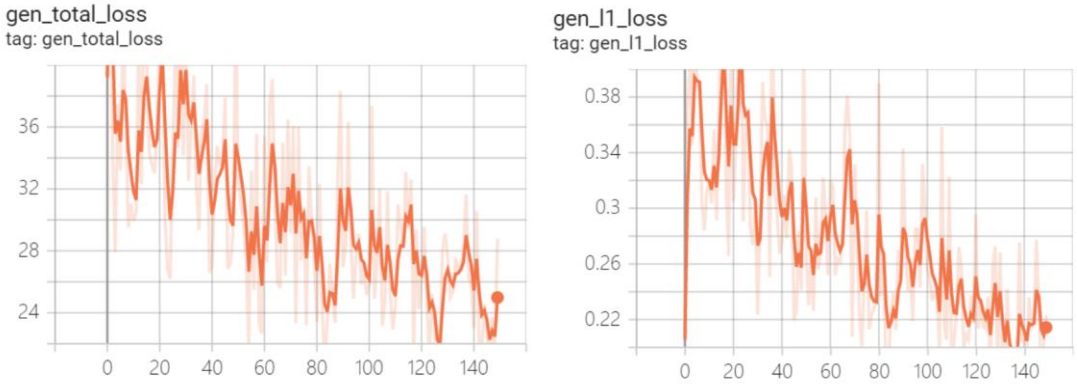
  租用服务器 v100，训练 40k 步，得到以下结果（选取测试集中的四张图片）：



图表 1 ： pix2pix（40k）训练集



图表 2 ： pix2pix（40k）测试集



图表 3 ： pix2pix 相关 loss

- 算法理解

Pix2Pix 的优点在于它能够生成高质量、细节丰富的输出图像，并且可以在各种图像翻译任务上进行有效训练。它能够捕捉输入图像和目标图像之间的复杂映射关系，并生成逼真的输出结果。

Pix2Pix 的训练过程中的对抗损失可以推动生成器生成更逼真的输出图像，而内容损失可以确保生成的图像与目标图像在视觉上相似。这种对抗性训练策略使得生成器能够逐渐学习到生成高质量图像的技巧。

此外，Pix2Pix 还具有一些改进和扩展的版本，如条件 Pix2Pix、CycleGAN 等。条件

Pix2Pix 允许根据额外的条件信息（如标签、语义分割图等）进行图像翻译，而 CycleGAN 进一步扩展了 Pix2Pix 的能力，可以在没有配对训练数据的情况下进行图像翻译。

总的来说，Pix2Pix 是一种强大的图像翻译模型，它能够在许多应用领域中产生出色的结果。通过适当的数据集准备、网络设计和训练过程调优，可以实现各种复杂的图像翻译任务，并生成具有高度逼真性和准确性的输出图像。

二、 CycleGAN

- 具体代码理解

```
dataset, metadata = tfds.load('cycle_gan/horse2zebra', with_info=True,
as_supervised=True)

train_horses, train_zebras = dataset['trainA'], dataset['trainB']
test_horses, test_zebras = dataset['testA'], dataset['testB']
```

tfds.load 函数用于从 TensorFlow Datasets 中加载指定名称的数据集。
with_info=True 参数表示加载数据集时同时返回有关数据集的元数据信息。
as_supervised=True 参数表示加载数据集时返回带有监督标签的形式，即每个样本都包括输入和目标。
A 和 B 分别表示马和斑马的图像。

```
BUFFER_SIZE = 1000
def random_crop(image):…
```

BUFFER_SIZE 指缓冲区大小。
random_crop 函数进行随机裁剪，返回裁剪后的图像；随机操作可以增加数据的多样性，有助于模型的训练和泛化能力。

```
def random_jitter(image):
  image = tf.image.resize(image, [286, 286],
                          method=tf.image.ResizeMethod.NEAREST_NEIGHBOR
)

  image = random_crop(image)
  image = tf.image.random_flip_left_right(image)
  return image
```

函数 random_jitter，用于对输入图像进行随机扰动（jitter）操作，通过扩充、随机裁剪、随机翻转实现；上述函数用于预处理数据。

```
OUTPUT_CHANNELS = 3

generator_g = pix2pix.unet_generator (OUTPUT_CHANNELS,
norm_type='instancenorm')
generator_f = pix2pix.unet_generator (OUTPUT_CHANNELS,
norm_type='instancenorm')
```

```
discriminator_x = pix2pix.discriminator(norm_type='instancenorm',
target=False)
discriminator_y = pix2pix.discriminator(norm_type='instancenorm',
target=False)
```

创建了 CycleGAN 模型中的生成器网络和判别器网络。

norm_type='instancenorm' 表示使用 Instance Normalization 来进行归一化操作。target=False 表示判别器不会对目标图像进行判别，而是对生成器所产生的图像进行判别。

```
to_zebra = generator_g(sample_horse)
to_horse = generator_f(sample_zebra)
plt.figure(figsize=(8, 8))
contrast = 8

imgs = [sample_horse, to_zebra, sample_zebra, to_horse]
title = ['Horse', 'To Zebra', 'Zebra', 'To Horse']

for i in range(len(imgs)):
  plt.subplot(2, 2, i+1)
  plt.title(title[i])
  if i % 2 == 0:
    plt.imshow(imgs[i][0] * 0.5 + 0.5)
  else:
    plt.imshow(imgs[i][0] * 0.5 * contrast + 0.5)
plt.show()
```

这段代码用于生成并展示转换后的图像，通过生成器网络将马的图像转换为斑马的图像，以及将斑马的图像转换为马的图像。

```
def discriminator_loss(real, generated):
  real_loss = loss_obj(tf.ones_like(real), real)
  generated_loss = loss_obj(tf.zeros_like(generated), generated)
  total_disc_loss = real_loss + generated_loss
  return total_disc_loss * 0.5

def generator_loss(generated):
  return loss_obj(tf.ones_like(generated), generated)
```

这段代码定义了判别器损失函数（discriminator_loss）和生成器损失函数（generator_loss）。

```
def calc_cycle_loss(real_image, cycled_image):
  loss1 = tf.reduce_mean(tf.abs(real_image - cycled_image))
  return LAMBDA * loss1
```

循环一致性损失函数用于鼓励生成器和反生成器能够将输入图像映射回原始域，并且保

持图像内容的一致性。通过最小化循环一致性损失，可以帮助模型学习更好的图像转换映射，并提高生成图像与真实图像之间的相似性。

```
generator_g_optimizer = tf.keras.optimizers.Adam(2e-4, beta_1=0.5)
generator_f_optimizer = tf.keras.optimizers.Adam(2e-4, beta_1=0.5)

discriminator_x_optimizer = tf.keras.optimizers.Adam(2e-4, beta_1=0.5)
discriminator_y_optimizer = tf.keras.optimizers.Adam(2e-4, beta_1=0.5)
```

这些优化器将被用于在训练过程中分别更新生成器 G、生成器 F、判别器 X 和判别器 Y 的参数，以最小化定义的损失函数。

```
def train_step(real_x, real_y):
  with tf.GradientTape(persistent=True) as tape:
…
```

根据梯度和优化器的配置，使用 apply_gradients 方法将梯度应用于生成器和判别器的可训练变量，从而更新模型参数。
使用了 persistent=True，这意味着在同一个梯度带上可以多次计算梯度。
此函数用于执行一次训练步骤，其中包括生成器和判别器的前向传播、损失计算和梯度更新。通过调用此函数进行多次迭代，可以完成模型的训练过程。

```
for inp in test_horses.take(5):
  generate_images(generator_g, inp)
```

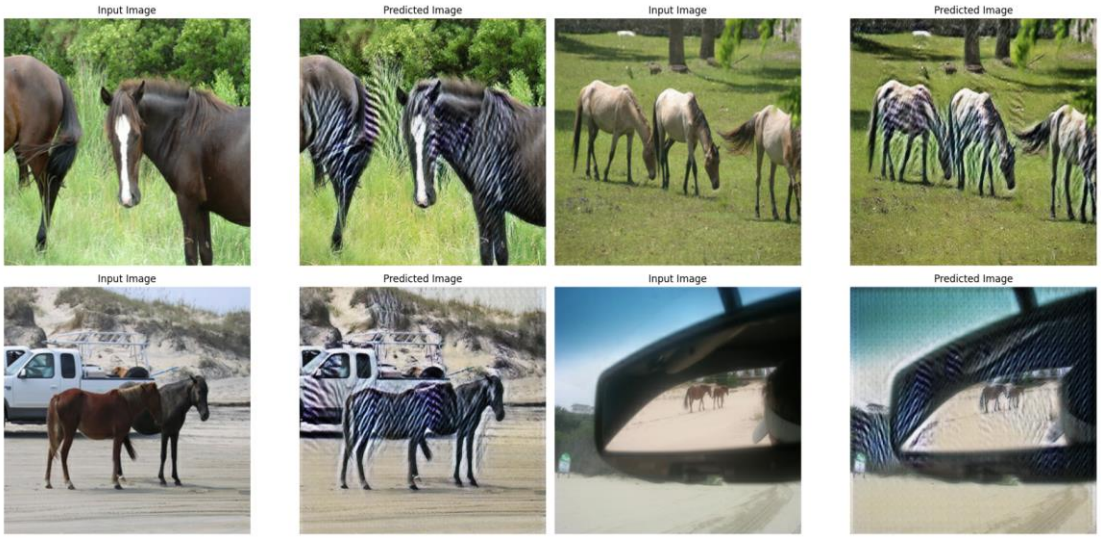
将训练好的生成器模型应用于测试数据集上的五张图像，并生成并展示生成器生成的图像。

- 参数设置与运行过程、结果展示

由于在自己笔记本上训练过于缓慢，选择使用 Colab 进行训练，但由于显卡与时间限制等，最终只测试了 EPOCHS = 5 与 EPOCHS = 10 两种情况，各选取了四份对比图，结果如下图所示：

发现由于 EPOCHS 较小，远不及论文中的 200，所以效果不太好。但可以看出，参数为 10 的结果相对于参数为 5 的结果有了比较明显的进步，斑马斑纹的特征更加真实。

图表 4 ： 运行截图（Colab 运行 cyclegan）



图表 5 ： epoch=5 结果图

图表 6 ： epoch=10 结果图

- 算法理解

该算法的核心思想是通过两个生成器和两个判别器的对抗训练，实现两个领域之间的图像转换。

对于调整参数，可以从以下几个方面进行考虑：

1. 网络结构：生成器和判别器的网络结构对于模型的性能起着重要的作用。可以尝试调整网络的层数、通道数和激活函数等参数，以探索更好的网络结构。
2. 损失函数权重：生成器和判别器损失函数中的权重参数（如循环一致性损失的权重）可以根据具体任务的需求进行调整。可以尝试增加或减小循环一致性损失和身份一致性损失的权重，以平衡不同损失的重要性。
3. 训练参数：调整训练参数如批量大小、缓冲区大小等，可以影响模型的训练效果和速度。可以尝试不同的参数配置，以找到最佳的训练设置。
4. 迭代次数：可以尝试增加或减小迭代次数，以观察模型在不同训练轮次下的表现，应该在 200 次附近为较优的参数值。

总体而言，对于 CycleGAN 算法的理解和参数调整，需要综合考虑网络结构、损失函数、

优化器、数据预处理和训练参数等多个方面。通过尝试不同的参数设置和观察生成结果的质量，可以逐步优化和改进模型的性能。