Mismatching Images: Keeping a Check on the Generator

Abhijit Singh Jowhari

Introduction

In the provided code snippet for the Stage1 model of a StackGAN, there is a variable called `mismatched_images`. This document provides an explanation of what this variable means and its role in the training process.

Code Snippet

Here is the code snippet that includes the variable `mismatched_images`:

```
class Stage1Model(tf.keras.Model):

    def __init__(self):
        super(Stage1Model, self).__init__()
        self.stage1_generator = Stage1Generator()
        self.stage1_discriminator = Stage1Discriminator()

    def train(self, train_ds, batch_size=64, num_epochs=600, z_dim=100, c_dim=128,
              stage1_generator_lr=0.0004, stage1_discriminator_lr=0.0004):

        self.generator_optimizer = tf.keras.optimizers.Adam(learning_rate=stage1_generator_lr,
                                                                                  beta_1=0.5,
                                                                                  beta_2=0.999)
        self.discriminator_optimizer = tf.keras.optimizers.Adam(learning_rate=stage1_discriminator_lr,
```

```
                                                                                beta_1=0.5, beta_2=0.999)

        for epoch in range(num_epochs):
            print("Epoch %d/%d:\n [" % (epoch + 1, num_epochs), end="")
            start_time = time.time()

            if epoch % 100 == 0:
                current_generator_lr = self.generator_optimizer.learning_rate.numpy()
                self.generator_optimizer.learning_rate.assign(current_generator_lr / 2)

                current_discriminator_lr = self.discriminator_optimizer.learning_rate.numpy()
                self.discriminator_optimizer.learning_rate.assign(current_discriminator_lr / 2)

            generator_loss_log = []
            discriminator_loss_log = []

            steps_per_epoch = 125
            batch_iter = iter(train_ds)

            for i in range(steps_per_epoch):
                if i % 50 == 0:
                     print("-", end="")

                image_batch, embedding_batch = next(batch_iter)
                z_noise = tf.random.normal((batch_size, z_dim))

                mismatched_images = tf.roll(image_batch, shift=1, axis=0)

                real_labels = tf.random.uniform(shape=(batch_size,), minval=0.9, maxval
```

```
                =1.0)
            fake_labels = tf.random.uniform(shape
                =(batch_size,), minval=0.0, maxval
                =0.1)
            mismatched_labels = tf.random.uniform(
                shape=(batch_size,), minval=0.0,
                maxval=0.1)
```

Listing 1: Code snippet for the Stage1 model of a StackGAN

Role and Purpose of `mismatched_images`

The variable `mismatched_images` typically represents images that are intentionally paired with incorrect (or mismatched) text embeddings. Here are the key points to understand:

- **Training Diversity**: During the training of GANs, especially in conditional GANs like StackGANs where the generation is conditioned on text embeddings, it is crucial to expose the discriminator to both correctly matched and mismatched pairs. This helps the discriminator learn to differentiate not just between real and fake images, but also between correct and incorrect matches between images and their corresponding embeddings.
- **Discriminator Training**: The discriminator in a StackGAN is trained to not only distinguish between real and generated images but also to verify whether an image matches the given text description (embedding). Introducing `mismatched_images` allows the discriminator to learn what incorrect matches look like, thereby improving its ability to enforce that generated images are well-aligned with their textual descriptions.

Implementation

In the provided code snippet, `mismatched_images` is created by rolling (shifting) the `image_batch` by one position along the batch dimension. This means each image is now paired with a text embedding from a different image in the batch.

```
mismatched_images = tf.roll(image_batch, shift=1, axis=0)
```

- `image_batch`: This is the batch of real images.
- `tf.roll`: This function rolls (shifts) the elements of the `image_batch`. By shifting the batch by 1 along the batch axis (`axis=0`), each image is moved to the position of the next image, creating a new set of image-embedding pairs that are intentionally mismatched.

Example

Consider a batch of images and their corresponding text embeddings:

- **Batch of Images**: [Image1, Image2, Image3, Image4]
- **Corresponding Embeddings**: [Embedding1, Embedding2, Embedding3, Embedding4]

After rolling:

- **Mismatched Images**: [Image4, Image1, Image2, Image3]

These `mismatched_images` are then paired with their original embeddings:

- **Mismatched Pairs**:
  - (Image4, Embedding1)
  - (Image1, Embedding2)
  - (Image2, Embedding3)
  - (Image3, Embedding4)

These pairs are used to train the discriminator to recognize mismatched image-text pairs as "incorrect."

Why It Matters

Training with `mismatched_images` improves the model's ability to generate images that are more accurately aligned with the given textual descriptions because the discriminator is more effective at enforcing correct image-text alignment.

In summary, `mismatched_images` in this context are used to provide negative examples to the discriminator, helping it learn to distinguish between images that correctly and incorrectly match their associated text descriptions.