LandscapeGAN

January 4, 2023

[ ]:

```
import numpy as np
```

[ ]:

```
from tensorflow import keras

dataset = keras.utils.image_dataset_from_directory(
    "Dataset",
    label_mode=None,
    image_size=(128,128),
    batch_size=8,
    smart_resize=True
)
```

```
Found 4319 files belonging to 1 classes.
```

[ ]:

```
dataset = dataset.map(lambda x: x/255.)
```

[ ]:

```
import matplotlib.pyplot as plt
for x in dataset:
    plt.axis("off")
    plt.imshow((x.numpy() * 255).astype("int32")[0])
    break
```

[ ]:

```
from tensorflow.keras import layers

discriminator = keras.Sequential(
    [
        keras.Input(shape=(128, 128, 3)),
        layers.Conv2D(128, kernel_size=4, strides=2, padding="same"),
        layers.LeakyReLU(alpha=0.2),
        layers.Conv2D(256, kernel_size=4, strides=2, padding="same"),
        layers.LeakyReLU(alpha=0.2),
        layers.Conv2D(256, kernel_size=4, strides=2, padding="same"),
        layers.LeakyReLU(alpha=0.2),
        layers.Flatten(),
        layers.Dropout(0.3),
        layers.Dense(1, activation="sigmoid"),
    ],
    name="discriminator",
)
```

[ ]:

```
discriminator.summary()
```

```
Model: "discriminator"

_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_8 (Conv2D)           (None, 64, 64, 128)       6272

 leaky_re_lu_14 (LeakyReLU)  (None, 64, 64, 128)       0

 conv2d_9 (Conv2D)           (None, 32, 32, 256)       524544

 leaky_re_lu_15 (LeakyReLU)  (None, 32, 32, 256)       0

 conv2d_10 (Conv2D)          (None, 16, 16, 256)       1048832

 leaky_re_lu_16 (LeakyReLU)  (None, 16, 16, 256)       0

 flatten_2 (Flatten)         (None, 65536)             0

 dropout_2 (Dropout)         (None, 65536)             0

 dense_4 (Dense)             (None, 1)                 65537

=================================================================
Total params: 1,645,185
Trainable params: 1,645,185
Non-trainable params: 0
_________________________________________________________________
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_8 (Conv2D)           (None, 64, 64, 128)       6272

 leaky_re_lu_14 (LeakyReLU)  (None, 64, 64, 128)       0

 conv2d_9 (Conv2D)           (None, 32, 32, 256)       524544

 leaky_re_lu_15 (LeakyReLU)  (None, 32, 32, 256)       0

 conv2d_10 (Conv2D)          (None, 16, 16, 256)       1048832

 leaky_re_lu_16 (LeakyReLU)  (None, 16, 16, 256)       0

 flatten_2 (Flatten)         (None, 65536)             0

 dropout_2 (Dropout)         (None, 65536)             0

 dense_4 (Dense)             (None, 1)                 65537

=================================================================
Total params: 1,645,185
Trainable params: 1,645,185
Non-trainable params: 0
```

```
_________________________________________________________________
```

[ ]:

```python
latent_dim = 128

generator = keras.Sequential(
    [
        keras.Input(shape=(latent_dim,)),
        layers.Dense(8 * 8 * 128),
        layers.Reshape((8, 8, 128)),
        layers.Conv2DTranspose(128, kernel_size=4, strides=2, padding="same"),
        layers.LeakyReLU(alpha=0.2),
        layers.Conv2DTranspose(256, kernel_size=4, strides=2, padding="same"),
        layers.LeakyReLU(alpha=0.2),
        layers.Conv2DTranspose(512, kernel_size=4, strides=2, padding="same"),
        layers.LeakyReLU(alpha=0.2),
        layers.Conv2DTranspose(1024, kernel_size=4, strides=2, padding="same"),
        layers.LeakyReLU(alpha=0.2),
        layers.Conv2D(3, kernel_size=5, padding="same", activation="tanh"),
    ],
    name="generator",
)
```

[ ]:

```python
generator.summary()
```

```
Model: "generator"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_5 (Dense)             (None, 8192)              1056768

 reshape_2 (Reshape)         (None, 8, 8, 128)         0

 conv2d_transpose_8 (Conv2DT  (None, 16, 16, 128)      262272
 ranspose)

 leaky_re_lu_17 (LeakyReLU)  (None, 16, 16, 128)       0

 conv2d_transpose_9 (Conv2DT  (None, 32, 32, 256)      524544
 ranspose)

 leaky_re_lu_18 (LeakyReLU)  (None, 32, 32, 256)       0

 conv2d_transpose_10 (Conv2D  (None, 64, 64, 512)      2097664
 Transpose)

 leaky_re_lu_19 (LeakyReLU)  (None, 64, 64, 512)       0

 conv2d_transpose_11 (Conv2D  (None, 128, 128, 1024)   8389632
 Transpose)

 leaky_re_lu_20 (LeakyReLU)  (None, 128, 128, 1024)    0

 conv2d_11 (Conv2D)          (None, 128, 128, 3)       76803

=================================================================
Total params: 12,407,683
Trainable params: 12,407,683
Non-trainable params: 0
_________________________________________________________________
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_5 (Dense)             (None, 8192)              1056768

 reshape_2 (Reshape)         (None, 8, 8, 128)         0

 conv2d_transpose_8 (Conv2DT  (None, 16, 16, 128)      262272
 ranspose)

 leaky_re_lu_17 (LeakyReLU)  (None, 16, 16, 128)       0

 conv2d_transpose_9 (Conv2DT  (None, 32, 32, 256)      524544
 ranspose)

 leaky_re_lu_18 (LeakyReLU)  (None, 32, 32, 256)       0

 conv2d_transpose_10 (Conv2D  (None, 64, 64, 512)      2097664
 Transpose)

 leaky_re_lu_19 (LeakyReLU)  (None, 64, 64, 512)       0

 conv2d_transpose_11 (Conv2D  (None, 128, 128, 1024)   8389632
 Transpose)

 leaky_re_lu_20 (LeakyReLU)  (None, 128, 128, 1024)    0

 conv2d_11 (Conv2D)          (None, 128, 128, 3)       76803

=================================================================
Total params: 12,407,683
Trainable params: 12,407,683
Non-trainable params: 0
_________________________________________________________________
```

[ ]:

```
import tensorflow as tf
class GAN(keras.Model):
    def __init__(self, discriminator, generator, latent_dim):
        super().__init__()
        self.discriminator = discriminator
        self.generator = generator
        self.latent_dim = latent_dim
        self.d_loss_metric = keras.metrics.Mean(name="d_loss")
        self.g_loss_metric = keras.metrics.Mean(name="g_loss")

    def compile(self, d_optimizer, g_optimizer, loss_fn):
        super(GAN, self).compile()
        self.d_optimizer = d_optimizer
        self.g_optimizer = g_optimizer
        self.loss_fn = loss_fn

    @property
    def metrics(self):
        return [self.d_loss_metric, self.g_loss_metric]

    def train_step(self, real_images):
        batch_size = tf.shape(real_images)[0]
        random_latent_vectors = tf.random.normal(
            shape=(batch_size, self.latent_dim))
        generated_images = self.generator(random_latent_vectors)
        combined_images = tf.concat([generated_images, real_images], axis=0)
        labels = tf.concat(
            [tf.ones((batch_size, 1)), tf.zeros((batch_size, 1))],
            axis=0
        )
        labels += 0.1 * tf.random.uniform(tf.shape(labels))

        with tf.GradientTape() as tape:
            predictions = self.discriminator(combined_images)
            d_loss = self.loss_fn(labels, predictions)
        grads = tape.gradient(d_loss, self.discriminator.trainable_weights)
        self.d_optimizer.apply_gradients(
            zip(grads, self.discriminator.trainable_weights)
        )

        random_latent_vectors = tf.random.normal(
            shape=(batch_size, self.latent_dim))

        misleading_labels = tf.zeros((batch_size, 1))

        with tf.GradientTape() as tape:
            predictions = self.discriminator(
                self.generator(random_latent_vectors))
            g_loss = self.loss_fn(misleading_labels, predictions)
        grads = tape.gradient(g_loss, self.generator.trainable_weights)
        self.g_optimizer.apply_gradients(
            zip(grads, self.generator.trainable_weights))

        self.d_loss_metric.update_state(d_loss)
        self.g_loss_metric.update_state(g_loss)
        return {"d_loss": self.d_loss_metric.result(),
                "g_loss": self.g_loss_metric.result()}
```

[ ]:
```
class GANMonitor(keras.callbacks.Callback):
    def __init__(self, num_img=3, latent_dim=128):
        self.num_img = num_img
        self.latent_dim = latent_dim

    def on_epoch_end(self, epoch, logs=None):
        random_latent_vectors = tf.random.normal(shape=(self.num_img, self.
 ↪latent_dim))
        generated_images = self.model.generator(random_latent_vectors)
        generated_images *= 255
        generated_images.numpy()
        for i in range(self.num_img):
            img = keras.utils.array_to_img(generated_images[i])
            img.save(f"Generated Images\generated_img_{epoch:03d}_{i}.png")
```

[ ]:
```
epochs = 50

gan = GAN(discriminator=discriminator, generator=generator,␣
 ↪latent_dim=latent_dim)
gan.compile(
    d_optimizer=keras.optimizers.Adam(learning_rate=0.000001),
    g_optimizer=keras.optimizers.Adam(learning_rate=0.00001),
    loss_fn=keras.losses.BinaryCrossentropy(),
)

gan.fit(
    dataset, epochs=epochs, callbacks=[GANMonitor(num_img=10,␣
 ↪latent_dim=latent_dim)]
)
```

```
Epoch 1/50
540/540 [==============================] - 90s 163ms/step - d_loss: 0.6882 -
g_loss: 0.6741
Epoch 2/50
540/540 [==============================] - 87s 161ms/step - d_loss: 0.6946 -
g_loss: 0.6680
Epoch 3/50
```

```
540/540 [==============================] - 87s 162ms/step - d_loss: 0.6947 -
g_loss: 0.7112
Epoch 4/50
540/540 [==============================] - 88s 162ms/step - d_loss: 0.7164 -
g_loss: 0.6676
Epoch 5/50
540/540 [==============================] - 87s 162ms/step - d_loss: 0.7135 -
g_loss: 0.6794
Epoch 6/50
540/540 [==============================] - 87s 162ms/step - d_loss: 0.7287 -
g_loss: 0.6495
Epoch 7/50
540/540 [==============================] - 87s 162ms/step - d_loss: 0.7296 -
g_loss: 0.6665
Epoch 8/50
540/540 [==============================] - 88s 162ms/step - d_loss: 0.7315 -
g_loss: 0.6772
Epoch 9/50
540/540 [==============================] - 87s 162ms/step - d_loss: 0.7302 -
g_loss: 0.6957
Epoch 10/50
540/540 [==============================] - 87s 161ms/step - d_loss: 0.7303 -
g_loss: 0.6954
Epoch 11/50
540/540 [==============================] - 88s 162ms/step - d_loss: 0.7308 -
g_loss: 0.6967
Epoch 12/50
540/540 [==============================] - 88s 162ms/step - d_loss: 0.7278 -
g_loss: 0.7026
Epoch 13/50
540/540 [==============================] - 88s 162ms/step - d_loss: 0.7252 -
g_loss: 0.7039
Epoch 14/50
540/540 [==============================] - 88s 162ms/step - d_loss: 0.7228 -
g_loss: 0.7097
Epoch 15/50
540/540 [==============================] - 87s 162ms/step - d_loss: 0.7210 -
g_loss: 0.7079
Epoch 16/50
540/540 [==============================] - 88s 162ms/step - d_loss: 0.7196 -
g_loss: 0.7085
Epoch 17/50
540/540 [==============================] - 88s 163ms/step - d_loss: 0.7176 -
g_loss: 0.7081
Epoch 18/50
540/540 [==============================] - 88s 162ms/step - d_loss: 0.7157 -
g_loss: 0.7088
Epoch 19/50
```

```
540/540 [==============================] - 87s 162ms/step - d_loss: 0.7145 -
g_loss: 0.7102
Epoch 20/50
540/540 [==============================] - 88s 162ms/step - d_loss: 0.7133 -
g_loss: 0.7177
Epoch 21/50
540/540 [==============================] - 87s 161ms/step - d_loss: 0.7122 -
g_loss: 0.7127
Epoch 22/50
540/540 [==============================] - 87s 161ms/step - d_loss: 0.7121 -
g_loss: 0.7150
Epoch 23/50
540/540 [==============================] - 86s 160ms/step - d_loss: 0.7119 -
g_loss: 0.7162
Epoch 24/50
540/540 [==============================] - 85s 158ms/step - d_loss: 0.7105 -
g_loss: 0.7187
Epoch 25/50
540/540 [==============================] - 87s 161ms/step - d_loss: 0.7077 -
g_loss: 0.7272
Epoch 26/50
540/540 [==============================] - 85s 158ms/step - d_loss: 0.7094 -
g_loss: 0.7213
Epoch 27/50
540/540 [==============================] - 86s 159ms/step - d_loss: 0.7076 -
g_loss: 0.7169
Epoch 28/50
540/540 [==============================] - 86s 159ms/step - d_loss: 0.7069 -
g_loss: 0.7250
Epoch 29/50
540/540 [==============================] - 85s 158ms/step - d_loss: 0.7061 -
g_loss: 0.7213
Epoch 30/50
540/540 [==============================] - 85s 158ms/step - d_loss: 0.7057 -
g_loss: 0.7225
Epoch 31/50
540/540 [==============================] - 86s 159ms/step - d_loss: 0.7043 -
g_loss: 0.7292
Epoch 32/50
540/540 [==============================] - 86s 159ms/step - d_loss: 0.7036 -
g_loss: 0.7300
Epoch 33/50
540/540 [==============================] - 86s 159ms/step - d_loss: 0.7049 -
g_loss: 0.7254
Epoch 34/50
540/540 [==============================] - 86s 160ms/step - d_loss: 0.7028 -
g_loss: 0.7343
Epoch 35/50
```

```
540/540 [==============================] - 88s 163ms/step - d_loss: 0.7015 -
g_loss: 0.7276
Epoch 36/50
540/540 [==============================] - 86s 159ms/step - d_loss: 0.7021 -
g_loss: 0.7263
Epoch 37/50
540/540 [==============================] - 86s 160ms/step - d_loss: 0.7000 -
g_loss: 0.7383
Epoch 38/50
540/540 [==============================] - 85s 158ms/step - d_loss: 0.7010 -
g_loss: 0.7312
Epoch 39/50
540/540 [==============================] - 86s 158ms/step - d_loss: 0.7005 -
g_loss: 0.7273
Epoch 40/50
540/540 [==============================] - 86s 158ms/step - d_loss: 0.6997 -
g_loss: 0.7387
Epoch 41/50
540/540 [==============================] - 86s 159ms/step - d_loss: 0.7011 -
g_loss: 0.7282
Epoch 42/50
540/540 [==============================] - 87s 160ms/step - d_loss: 0.7009 -
g_loss: 0.7290
Epoch 43/50
540/540 [==============================] - 88s 163ms/step - d_loss: 0.6991 -
g_loss: 0.7322
Epoch 44/50
540/540 [==============================] - 88s 163ms/step - d_loss: 0.6992 -
g_loss: 0.7353
Epoch 45/50
540/540 [==============================] - 87s 162ms/step - d_loss: 0.6978 -
g_loss: 0.7364
Epoch 46/50
540/540 [==============================] - 87s 161ms/step - d_loss: 0.6977 -
g_loss: 0.7319
Epoch 47/50
540/540 [==============================] - 88s 162ms/step - d_loss: 0.7008 -
g_loss: 0.7366
Epoch 48/50
540/540 [==============================] - 88s 164ms/step - d_loss: 0.6989 -
g_loss: 0.7372
Epoch 49/50
540/540 [==============================] - 86s 159ms/step - d_loss: 0.6978 -
g_loss: 0.7343
Epoch 50/50
540/540 [==============================] - 86s 159ms/step - d_loss: 0.6976 -
g_loss: 0.7275
```

[ ]:

```
<keras.callbacks.History at 0x176b83c8940>
```

[ ]:

```
generator.save('savedmodels\GeneratorTanh04/01')
```

```
WARNING:tensorflow:Compiled the loaded model, but the compiled metrics have yet
to be built. `model.compile_metrics` will be empty until you train or evaluate
the model.
```

```
WARNING:absl:Found untraced functions such as _jit_compiled_convolution_op,
_jit_compiled_convolution_op, _jit_compiled_convolution_op,
_jit_compiled_convolution_op, _jit_compiled_convolution_op while saving (showing
5 of 5). These functions will not be directly callable after loading.
```

```
INFO:tensorflow:Assets written to: savedmodels\GeneratorTanh04/01\assets
```

```
INFO:tensorflow:Assets written to: savedmodels\GeneratorTanh04/01\assets
```

[ ]:

```
import tensorflow as tf
new_model = tf.keras.models.load_model('savedmodels\GeneratorTanh04/01')
```

```
WARNING:tensorflow:No training configuration found in save file, so the model
was *not* compiled. Compile it manually.
```

```
WARNING:tensorflow:No training configuration found in save file, so the model
was *not* compiled. Compile it manually.
```

[ ]:

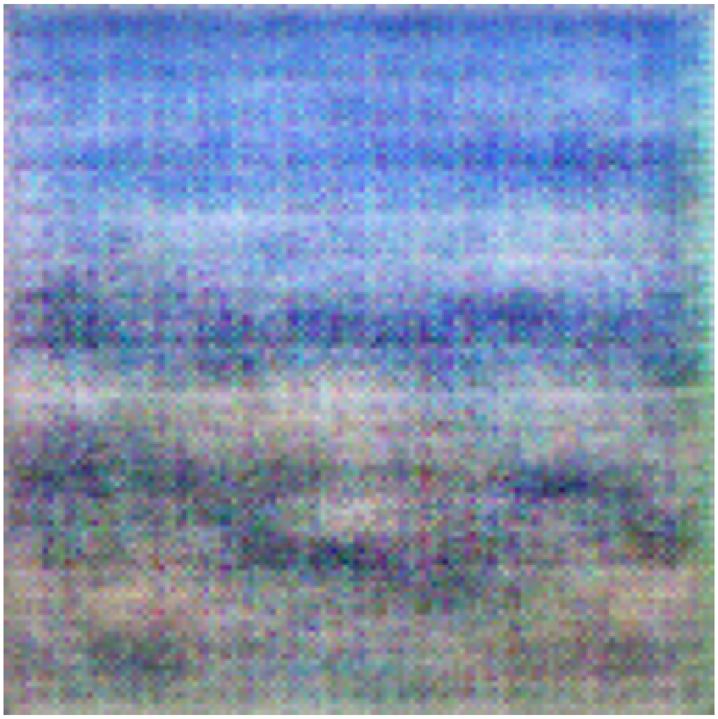
```
num = 1
random_vector = tf.random.normal(shape=(num, 256))
images = new_model(random_vector)
images *= 255
tf.shape(images[0])
```

[ ]:

```
<tf.Tensor: shape=(3,), dtype=int32, numpy=array([256, 256,   3])>
```

[ ]:

```
for x in range(num):
    img = keras.utils.array_to_img(images[x])
    plt.axis("off")
    plt.imshow(img)
```



[ ]:
```
new_model.summary()
```

[ ]:
```
import tensorflow as tf

from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras import mixed_precision

policy = mixed_precision.Policy('float32')
mixed_precision.set_global_policy(policy)
mixed_precision.set_global_policy('float32')

print('Compute dtype: %s' % policy.compute_dtype)
print('Variable dtype: %s' % policy.variable_dtype)
```

```
Compute dtype: float32
Variable dtype: float32
```

[ ]:
```
num = 1
random_vector = tf.random.normal(shape=(num, 128))
images = new_model(random_vector)
images *= 255
print(tf.shape(images[0]))
```

```
for x in range(num):
    img = keras.utils.array_to_img(images[x])
    plt.axis("off")
    plt.imshow(img)
```

```
tf.Tensor([128 128   3], shape=(3,), dtype=int32)
```