```
import tensorflow as tf
```

```
tf.__version__
```

```
'2.9.2'
```

```
# To generate GIFs
!pip install imageio
!pip install git+https://github.com/tensorflow/docs
```

```
g in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
ement already satisfied: imageio in /usr/local/lib/python3.8/dist-packages (2.9.0)
ement already satisfied: pillow in /usr/local/lib/python3.8/dist-packages (from imageio) (7.1.2)
ement already satisfied: numpy in /usr/local/lib/python3.8/dist-packages (from imageio) (1.21.6)
g in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
ting git+https://github.com/tensorflow/docs
ing https://github.com/tensorflow/docs to /tmp/pip-req-build-420i4y1u
ing command git clone --filter=blob:none --quiet https://github.com/tensorflow/docs /tmp/pip-req-build-420i4y1u
lved https://github.com/tensorflow/docs to commit badea29fa47ed6595244641bfee5a3d6717701f4
aring metadata (setup.py) ... done
ement already satisfied: astor in /usr/local/lib/python3.8/dist-packages (from tensorflow-docs==0.0.0.dev0) (0.8.1)
ement already satisfied: absl-py in /usr/local/lib/python3.8/dist-packages (from tensorflow-docs==0.0.0.dev0) (1.3.0)
ement already satisfied: jinja2 in /usr/local/lib/python3.8/dist-packages (from tensorflow-docs==0.0.0.dev0) (2.11.3)
ement already satisfied: nbformat in /usr/local/lib/python3.8/dist-packages (from tensorflow-docs==0.0.0.dev0) (5.7.1)
ement already satisfied: protobuf<3.20,>=3.12.0 in /usr/local/lib/python3.8/dist-packages (from tensorflow-docs==0.0.0.dev0) (3.19.6)
ement already satisfied: pyyaml in /usr/local/lib/python3.8/dist-packages (from tensorflow-docs==0.0.0.dev0) (6.0)
ement already satisfied: MarkupSafe>=0.23 in /usr/local/lib/python3.8/dist-packages (from jinja2->tensorflow-docs==0.0.0.dev0) (2.0.1)
ement already satisfied: jsonschema>=2.6 in /usr/local/lib/python3.8/dist-packages (from nbformat->tensorflow-docs==0.0.0.dev0) (4.3.3)
ement already satisfied: jupyter-core in /usr/local/lib/python3.8/dist-packages (from nbformat->tensorflow-docs==0.0.0.dev0) (5.1.3)
ement already satisfied: traitlets>=5.1 in /usr/local/lib/python3.8/dist-packages (from nbformat->tensorflow-docs==0.0.0.dev0) (5.7.1)
ement already satisfied: fastjsonschema in /usr/local/lib/python3.8/dist-packages (from nbformat->tensorflow-docs==0.0.0.dev0) (2.16.2)
ement already satisfied: pyrsistent!=0.17.0,!=0.17.1,!=0.17.2,>=0.14.0 in /usr/local/lib/python3.8/dist-packages (from jsonschema>=2.6->
ement already satisfied: importlib-resources>=1.4.0 in /usr/local/lib/python3.8/dist-packages (from jsonschema>=2.6->nbformat->tensorflo
ement already satisfied: attrs>=17.4.0 in /usr/local/lib/python3.8/dist-packages (from jsonschema>=2.6->nbformat->tensorflow-docs==0.0.0
ement already satisfied: platformdirs>=2.5 in /usr/local/lib/python3.8/dist-packages (from jupyter-core->nbformat->tensorflow-docs==0.0.
ement already satisfied: zipp>=3.1.0 in /usr/local/lib/python3.8/dist-packages (from importlib-resources>=1.4.0->jsonschema>=2.6->nbform
ng wheels for collected packages: tensorflow-docs
ding wheel for tensorflow-docs (setup.py) ... done
ted wheel for tensorflow-docs: filename=tensorflow_docs-0.0.0.dev0-py3-none-any.whl size=184468 sha256=ef4556ed2524c45f454b5e6ccfee521a2
ed in directory: /tmp/pip-ephem-wheel-cache-fid87qjr/wheels/3b/ee/a2/ab4d36a9a4af495bcb936f3e849d4b497b65fa40548a68d6c3
sfully built tensorflow-docs
ling collected packages: tensorflow-docs
sfully installed tensorflow-docs-0.0.0.dev0
```

```
import glob
import imageio
import matplotlib.pyplot as plt
import numpy as np
import os
import PIL
from tensorflow.keras import layers
import time
from PIL import Image
import pandas as pd
import keras.utils as image
from tqdm import tqdm
import cv2

from IPython import display
```

```
#mounting google drive
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
GENERATE_SQUARE = 28
IMAGE_CHANNELS = 1

# Configuration
DATA_PATH = '/content/drive/MyDrive/archive (1)/Alzheimer_s Dataset/train/NonDemented'
EPOCHS = 100
BATCH_SIZE = 256
BUFFER_SIZE = 60000
```

```
print(f"Will generate {GENERATE_SQUARE}px square images.")
```

```
Will generate 28px square images.
```

```
training_data = []
faces_path = os.path.join(DATA_PATH)
for filename in tqdm(os.listdir(faces_path)):
    path = os.path.join(faces_path,filename)
    image = Image.open(path).resize((GENERATE_SQUARE,GENERATE_SQUARE),Image.ANTIALIAS)
    training_data.append(np.asarray(image))
training_data = np.reshape(training_data,(-1,GENERATE_SQUARE,GENERATE_SQUARE,IMAGE_CHANNELS))
training_data = training_data.astype(np.float32)
training_data = training_data / 127.5 - 1.
```

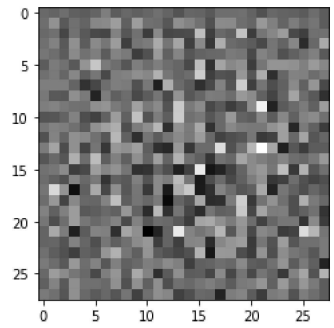
```
100%|██████████| 2560/2560 [01:10<00:00, 36.39it/s]
```

```
train_dataset = tf.data.Dataset.from_tensor_slices(training_data).shuffle(BUFFER_SIZE).batch(BATCH_SIZE)
```

```
def make_generator_model():
    model = tf.keras.Sequential()
    model.add(layers.Dense(7*7*256, use_bias=False, input_shape=(100,)))
    model.add(layers.BatchNormalization())
    model.add(layers.LeakyReLU())

    model.add(layers.Reshape((7, 7, 256)))
    assert model.output_shape == (None, 7, 7, 256)  # Note: None is the batch size

    model.add(layers.Conv2DTranspose(128, (5, 5), strides=(1, 1), padding='same', use_bias=False))
    assert model.output_shape == (None, 7, 7, 128)
    model.add(layers.BatchNormalization())
    model.add(layers.LeakyReLU())

    model.add(layers.Conv2DTranspose(64, (5, 5), strides=(2, 2), padding='same', use_bias=False))
    assert model.output_shape == (None, 14, 14, 64)
    model.add(layers.BatchNormalization())
    model.add(layers.LeakyReLU())

    model.add(layers.Conv2DTranspose(1, (5, 5), strides=(2, 2), padding='same', use_bias=False, activation='tanh'))
    assert model.output_shape == (None, 28, 28, 1)

    return model
```

```
generator = make_generator_model()

noise = tf.random.normal([1, 100])
generated_image = generator(noise, training=False)

plt.imshow(generated_image[0, :, :, 0], cmap='gray')
```

```
<matplotlib.image.AxesImage at 0x7f1eaae17af0>
```



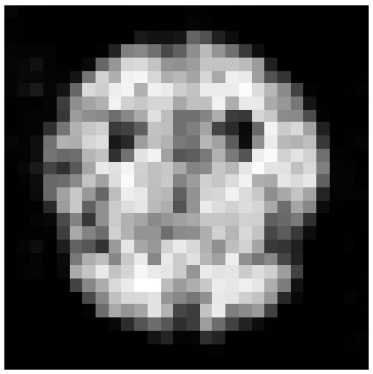

```
def make_discriminator_model():
    model = tf.keras.Sequential()
    model.add(layers.Conv2D(64, (5, 5), strides=(2, 2), padding='same',
                                     input_shape=[28, 28, 1]))
    model.add(layers.LeakyReLU())
    model.add(layers.Dropout(0.3))

    model.add(layers.Conv2D(128, (5, 5), strides=(2, 2), padding='same'))
```

```
    model.add(layers.LeakyReLU())
    model.add(layers.Dropout(0.3))

    model.add(layers.Flatten())
    model.add(layers.Dense(1))

    return model

discriminator = make_discriminator_model()
decision = discriminator(generated_image)
print (decision)
```

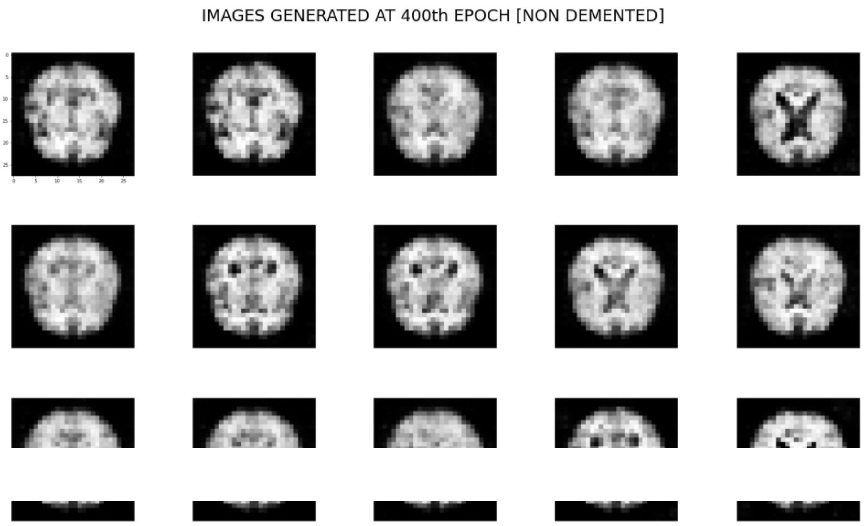
```
 tf.Tensor([[-0.00094588]], shape=(1, 1), dtype=float32)
```

```
# This method returns a helper function to compute cross entropy loss
cross_entropy = tf.keras.losses.BinaryCrossentropy(from_logits=True)


def discriminator_loss(real_output, fake_output):
    real_loss = cross_entropy(tf.ones_like(real_output), real_output)
    fake_loss = cross_entropy(tf.zeros_like(fake_output), fake_output)
    total_loss = real_loss + fake_loss
    return total_loss


def generator_loss(fake_output):
    return cross_entropy(tf.ones_like(fake_output), fake_output)


generator_optimizer = tf.keras.optimizers.Adam(1e-4)
discriminator_optimizer = tf.keras.optimizers.Adam(1e-4)


checkpoint_dir = './training_checkpoints'
checkpoint_prefix = os.path.join(checkpoint_dir, "ckpt")
checkpoint = tf.train.Checkpoint(generator_optimizer=generator_optimizer,
                                 discriminator_optimizer=discriminator_optimizer,
                                 generator=generator,
                                 discriminator=discriminator)


noise_dim = 100
num_examples_to_generate = 16

# You will reuse this seed overtime (so it's easier)
# to visualize progress in the animated GIF)
seed = tf.random.normal([num_examples_to_generate, noise_dim])


# Notice the use of `tf.function`
# This annotation causes the function to be "compiled".
@tf.function
def train_step(images):
    noise = tf.random.normal([BATCH_SIZE, noise_dim])

    with tf.GradientTape() as gen_tape, tf.GradientTape() as disc_tape:
      generated_images = generator(noise, training=True)

      real_output = discriminator(images, training=True)
      fake_output = discriminator(generated_images, training=True)

      gen_loss = generator_loss(fake_output)
      disc_loss = discriminator_loss(real_output, fake_output)

    gradients_of_generator = gen_tape.gradient(gen_loss, generator.trainable_variables)
    gradients_of_discriminator = disc_tape.gradient(disc_loss, discriminator.trainable_variables)

    generator_optimizer.apply_gradients(zip(gradients_of_generator, generator.trainable_variables))
    discriminator_optimizer.apply_gradients(zip(gradients_of_discriminator, discriminator.trainable_variables))


def train(dataset, epochs):
  for epoch in range(epochs):
    start = time.time()

    for image_batch in dataset:
      train_step(image_batch)

    # Produce images for the GIF as you go
```

```
      display.clear_output(wait=True)
      generate_and_save_images(generator,
                               epoch + 1,
                               seed)

      # Save the model every 15 epochs
      if (epoch + 1) % 15 == 0:
        checkpoint.save(file_prefix = checkpoint_prefix)

      print ('Time for epoch {} is {} sec'.format(epoch + 1, time.time()-start))

    # Generate after the final epoch
    display.clear_output(wait=True)
    generate_and_save_images(generator,
                             epochs,
                             seed)


def generate_and_save_images(model, epoch, test_input):
  # Notice `training` is set to False.
  # This is so all layers run in inference mode (batchnorm).
  predictions = model(test_input, training=False)

  fig = plt.figure(figsize=(5, 5))

  for i in range(predictions.shape[0]):
      plt.imshow(predictions[i, :, :, 0] * 127.5 + 127.5, cmap='gray')
      plt.savefig('/content/drive/MyDrive/archive (1)/Alzheimer_s Dataset/GAN/train/NonDemented/image_at_epoch_{:04d}-{}.png'.format(epoch+40
      plt.axis('off')

  plt.show()


train(train_dataset, EPOCHS)
```



```
fig, ax = plt.subplots(figsize=(14, 8))
fig.suptitle('IMAGES GENERATED AT 400th EPOCH [NON DEMENTED]',fontsize=18, y=1.03)

for i in range(0,15):
  plt.subplot(3, 5, i+1)
  plt.axis('off')
  img=cv2.imread('/content/drive/MyDrive/archive (1)/Alzheimer_s Dataset/GanFinal/NonDemented/image_at_epoch_0500-{}.png'.format(i))
  plt.imshow(img, cmap='gray')


plt.tight_layout()
```



✓ 2s completed at 7:31 PM