Tensorflow with GPU

This notebook provides an introduction to computing on a GPU in Colab. In this notebook you will connect to a GPU, and then run some basic TensorFlow operations on both the CPU and a GPU, observing the speedup provided by using the GPU.

Enabling and testing the GPU

First, you'll need to enable GPUs for the notebook:

- Navigate to Edit→Notebook Settings
- select GPU from the Hardware Accelerator drop-down

Next, we'll confirm that we can connect to the GPU with tensorflow:

```
%tensorflow_version 2.x
import tensorflow as tf
device_name = tf.test.gpu_device_name()
if device_name != '/device:GPU:0':
  raise SystemError('GPU device not found')
print('Found GPU at: {}'.format(device_name))
```

```
Found GPU at: /device:GPU:0
```

Variational AutoEncoder

credit: https://keras.io/examples/generative/vae/

```
%tensorflow_version 2.x
import tensorflow as tf
import timeit

device_name = tf.test.gpu_device_name()
if device_name != '/device:GPU:0':
  print(
      '\n\nThis error most likely means that this notebook is not '
      'configured to use a GPU.  Change this in Notebook Settings via the '
```

```
    'command palette (cmd/ctrl-shift-P) or the Edit menu.\n\n')
  raise SystemError('GPU device not found')

def cpu():
  with tf.device('/cpu:0'):
    random_image_cpu = tf.random.normal((100, 100, 100, 3))
    net_cpu = tf.keras.layers.Conv2D(32, 7)(random_image_cpu)
    return tf.math.reduce_sum(net_cpu)

def gpu():
  with tf.device('/device:GPU:0'):
    random_image_gpu = tf.random.normal((100, 100, 100, 3))
    net_gpu = tf.keras.layers.Conv2D(32, 7)(random_image_gpu)
    return tf.math.reduce_sum(net_gpu)

# We run each op once to warm up; see: https://stackoverflow.com/a/45067900
cpu()
gpu()

# Run the op several times.
print('Time (s) to convolve 32x7x7x3 filter over random 100x100x100x3 images '
      '(batch x height x width x channel). Sum of ten runs.')
print('CPU (s):')
cpu_time = timeit.timeit('cpu()', number=10, setup="from __main__ import cpu")
print(cpu_time)
print('GPU (s):')
gpu_time = timeit.timeit('gpu()', number=10, setup="from __main__ import gpu")
print(gpu_time)
print('GPU speedup over CPU: {}x'.format(int(cpu_time/gpu_time)))
```

```
Time (s) to convolve 32x7x7x3 filter over random 100x100x100x3 images (batch x
CPU (s):
3.3689634109999957
GPU (s):
0.10654259700000068
GPU speedup over CPU: 31x
```

Double-click (or enter) to edit

```
import numpy as np
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers


class Sampling(layers.Layer):
    """Uses (z_mean, z_log_var) to sample z, the vector encoding a digit."""

    def call(self, inputs):
        z_mean, z_log_var = inputs
        batch = tf.shape(z_mean)[0]
        dim = tf.shape(z_mean)[1]
        epsilon = tf.keras.backend.random_normal(shape=(batch, dim))
        return z_mean + tf.exp(0.5 * z_log_var) * epsilon
```

```
latent_dim = 2

encoder_inputs = keras.Input(shape=(28, 28, 1))
x = layers.Conv2D(32, 3, activation="relu", strides=2, padding="same")(encoder_in
x = layers.Conv2D(64, 3, activation="relu", strides=2, padding="same")(x)
x = layers.Flatten()(x)
x = layers.Dense(16, activation="relu")(x)
z_mean = layers.Dense(latent_dim, name="z_mean")(x)
z_log_var = layers.Dense(latent_dim, name="z_log_var")(x)
z = Sampling()([z_mean, z_log_var])
encoder = keras.Model(encoder_inputs, [z_mean, z_log_var, z], name="encoder")
encoder.summary()
```

```
Model: "encoder"
__________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================
input_1 (InputLayer)            [(None, 28, 28, 1)]  0
__________________________________________________________________________________
conv2d_22 (Conv2D)              (None, 14, 14, 32)   320         input_1[0][0]
__________________________________________________________________________________
conv2d_23 (Conv2D)              (None, 7, 7, 64)     18496       conv2d_22[0]
__________________________________________________________________________________
flatten (Flatten)               (None, 3136)         0           conv2d_23[0]
__________________________________________________________________________________
dense (Dense)                   (None, 16)           50192       flatten[0][0]
__________________________________________________________________________________
z_mean (Dense)                  (None, 2)            34          dense[0][0]
__________________________________________________________________________________
z_log_var (Dense)               (None, 2)            34          dense[0][0]
__________________________________________________________________________________
sampling (Sampling)             (None, 2)            0           z_mean[0][0]
                                                                 z_log_var[0]
==================================================================================
Total params: 69,076
Trainable params: 69,076
Non-trainable params: 0
__________________________________________________________________________________
```

```
latent_inputs = keras.Input(shape=(latent_dim,))
x = layers.Dense(7 * 7 * 64, activation="relu")(latent_inputs)
x = layers.Reshape((7, 7, 64))(x)
x = layers.Conv2DTranspose(64, 3, activation="relu", strides=2, padding="same")(x
x = layers.Conv2DTranspose(32, 3, activation="relu", strides=2, padding="same")(x
decoder_outputs = layers.Conv2DTranspose(1, 3, activation="sigmoid", padding="sam
decoder = keras.Model(latent_inputs, decoder_outputs, name="decoder")
decoder.summary()
```

```
Model: "decoder"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_2 (InputLayer)         [(None, 2)]               0
_________________________________________________________________
dense_1 (Dense)              (None, 3136)              9408
_________________________________________________________________
reshape (Reshape)            (None, 7, 7, 64)          0
_________________________________________________________________
conv2d_transpose (Conv2DTran (None, 14, 14, 64)        36928
_________________________________________________________________
conv2d_transpose_1 (Conv2DTr (None, 28, 28, 32)        18464
_________________________________________________________________
conv2d_transpose_2 (Conv2DTr (None, 28, 28, 1)         289
=================================================================
Total params: 65,089
Trainable params: 65,089
Non-trainable params: 0
_________________________________________________________________
```

```
class VAE(keras.Model):
    def __init__(self, encoder, decoder, **kwargs):
        super(VAE, self).__init__(**kwargs)
        self.encoder = encoder
        self.decoder = decoder

    def train_step(self, data):
        if isinstance(data, tuple):
            data = data[0]
        with tf.GradientTape() as tape:
            z_mean, z_log_var, z = encoder(data)
            reconstruction = decoder(z)
            reconstruction_loss = tf.reduce_mean(
                keras.losses.binary_crossentropy(data, reconstruction)
            )
            reconstruction_loss *= 28 * 28
            kl_loss = 1 + z_log_var - tf.square(z_mean) - tf.exp(z_log_var)
            kl_loss = tf.reduce_mean(kl_loss)
            kl_loss *= -0.5
            total_loss = reconstruction_loss + kl_loss
        grads = tape.gradient(total_loss, self.trainable_weights)
        self.optimizer.apply_gradients(zip(grads, self.trainable_weights))
        return {
            "loss": total_loss,
            "reconstruction_loss": reconstruction_loss,
            "kl_loss": kl_loss,
        }


(x_train, _), (x_test, _) = keras.datasets.mnist.load_data()
mnist_digits = np.concatenate([x_train, x_test], axis=0)
mnist_digits = np.expand_dims(mnist_digits, -1).astype("float32") / 255

vae = VAE(encoder, decoder)
vae.compile(optimizer=keras.optimizers.Adam())
vae.fit(mnist_digits, epochs=30, batch_size=128)
```

```
Epoch 2/30
547/547 [==============================] - 4s 7ms/step - loss: 185.5423 - re
Epoch 3/30
547/547 [==============================] - 4s 7ms/step - loss: 169.6178 - re
Epoch 4/30
547/547 [==============================] - 4s 7ms/step - loss: 163.1009 - re
Epoch 5/30
547/547 [==============================] - 4s 7ms/step - loss: 160.2842 - re
Epoch 6/30
```

```
547/547 [==============================] - 4s 7ms/step - loss: 158.4207 - re
Epoch 7/30
547/547 [==============================] - 4s 8ms/step - loss: 157.0623 - re
Epoch 8/30
547/547 [==============================] - 4s 8ms/step - loss: 155.9911 - re
Epoch 9/30
547/547 [==============================] - 4s 8ms/step - loss: 155.1492 - re
Epoch 10/30
547/547 [==============================] - 4s 8ms/step - loss: 154.3663 - re
Epoch 11/30
547/547 [==============================] - 4s 8ms/step - loss: 153.7860 - re
Epoch 12/30
547/547 [==============================] - 4s 8ms/step - loss: 153.1879 - re
Epoch 13/30
547/547 [==============================] - 4s 8ms/step - loss: 152.8215 - re
Epoch 14/30
547/547 [==============================] - 4s 8ms/step - loss: 152.4045 - re
Epoch 15/30
547/547 [==============================] - 4s 8ms/step - loss: 152.0620 - re
Epoch 16/30
547/547 [==============================] - 4s 8ms/step - loss: 151.6716 - re
Epoch 17/30
547/547 [==============================] - 4s 8ms/step - loss: 151.3534 - re
Epoch 18/30
547/547 [==============================] - 4s 8ms/step - loss: 151.2538 - re
Epoch 19/30
547/547 [==============================] - 4s 8ms/step - loss: 150.8842 - re
Epoch 20/30
547/547 [==============================] - 4s 8ms/step - loss: 150.6137 - re
Epoch 21/30
547/547 [==============================] - 4s 8ms/step - loss: 150.3531 - re
Epoch 22/30
547/547 [==============================] - 4s 8ms/step - loss: 150.2528 - re
Epoch 23/30
547/547 [==============================] - 4s 8ms/step - loss: 150.0549 - re
Epoch 24/30
547/547 [==============================] - 4s 8ms/step - loss: 149.8594 - re
Epoch 25/30
547/547 [==============================] - 4s 8ms/step - loss: 149.6435 - re
Epoch 26/30
547/547 [==============================] - 4s 8ms/step - loss: 149.5845 - re
Epoch 27/30
547/547 [==============================] - 4s 8ms/step - loss: 149.5089 - re
Epoch 28/30
547/547 [==============================] - 4s 8ms/step - loss: 149.3294 - re
Epoch 29/30
547/547 [==============================] - 4s 8ms/step - loss: 149.2504 - re
Epoch 30/30
547/547 [==============================] - 4s 8ms/step - loss: 149.0676 - re
<tensorflow.python.keras.callbacks.History at 0x7ff3a918f2b0>
```

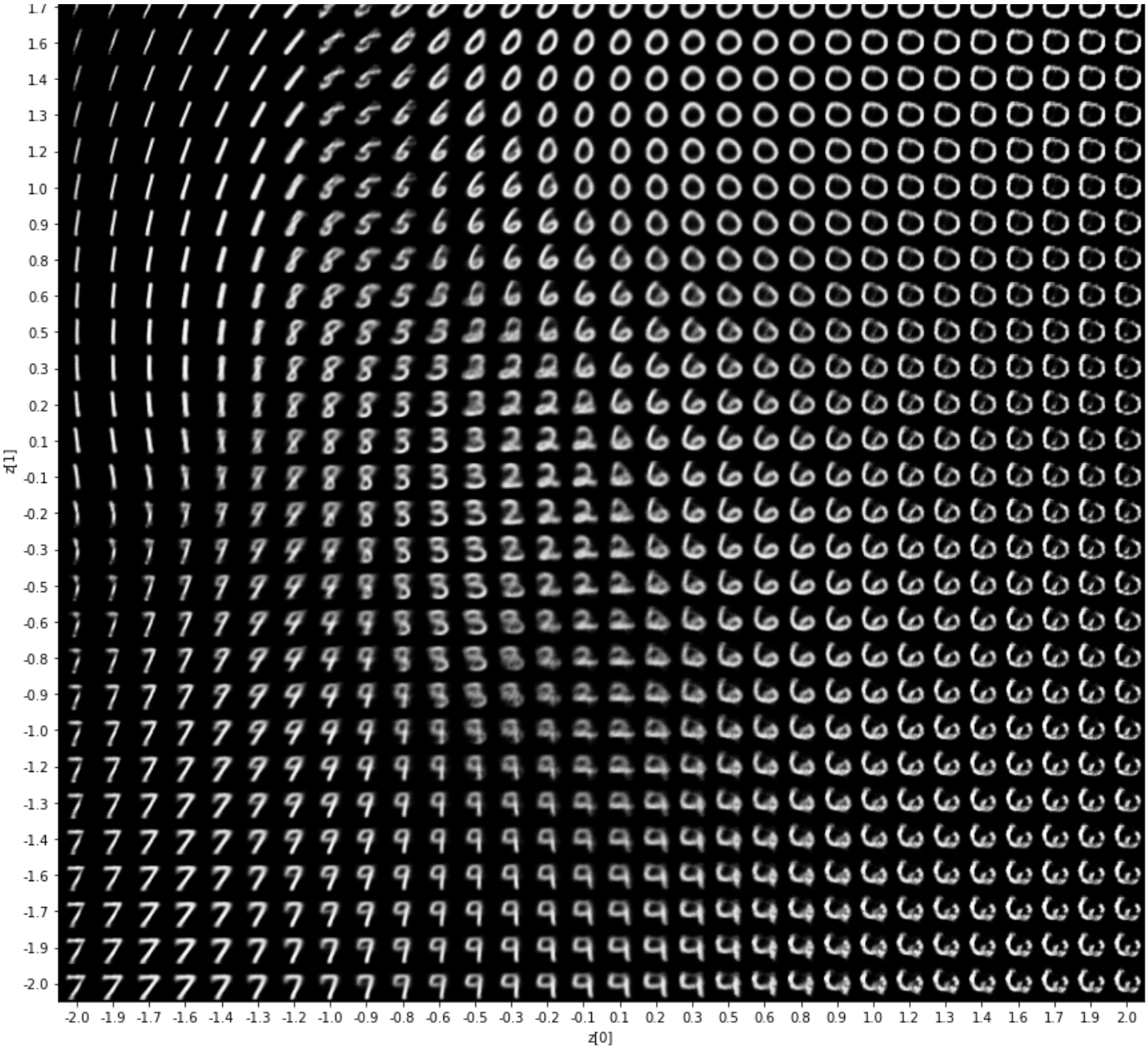
```
import matplotlib.pyplot as plt


def plot_latent(encoder, decoder):
    # display a n*n 2D manifold of digits
    n = 30
    digit_size = 28
    scale = 2.0
    figsize = 15
    figure = np.zeros((digit_size * n, digit_size * n))
    # linearly spaced coordinates corresponding to the 2D plot
    # of digit classes in the latent space
    grid_x = np.linspace(-scale, scale, n)
    grid_y = np.linspace(-scale, scale, n)[::-1]

    for i, yi in enumerate(grid_y):
        for j, xi in enumerate(grid_x):
            z_sample = np.array([[xi, yi]])
            x_decoded = decoder.predict(z_sample)
            digit = x_decoded[0].reshape(digit_size, digit_size)
            figure[
                i * digit_size : (i + 1) * digit_size,
                j * digit_size : (j + 1) * digit_size,
            ] = digit

    plt.figure(figsize=(figsize, figsize))
    start_range = digit_size // 2
    end_range = n * digit_size + start_range + 1
    pixel_range = np.arange(start_range, end_range, digit_size)
    sample_range_x = np.round(grid_x, 1)
    sample_range_y = np.round(grid_y, 1)
    plt.xticks(pixel_range, sample_range_x)
    plt.yticks(pixel_range, sample_range_y)
    plt.xlabel("z[0]")
    plt.ylabel("z[1]")
    plt.imshow(figure, cmap="Greys_r")
    plt.show()


plot_latent(encoder, decoder)
```

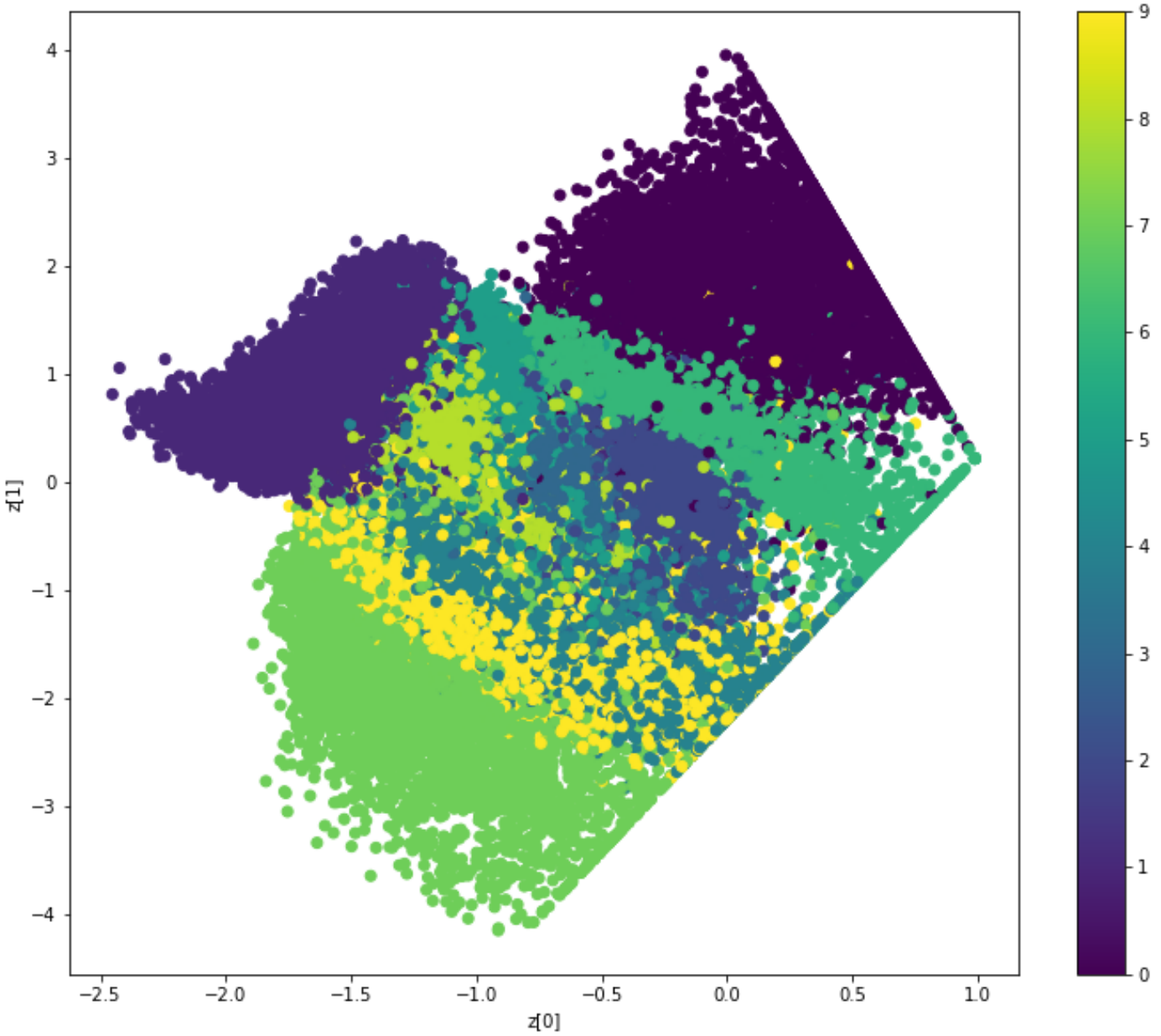



```
def plot_label_clusters(encoder, decoder, data, labels):
```

```
    # display a 2D plot of the digit classes in the latent space
    z_mean, _, _ = encoder.predict(data)
    plt.figure(figsize=(12, 10))
    plt.scatter(z_mean[:, 0], z_mean[:, 1], c=labels)
    plt.colorbar()
    plt.xlabel("z[0]")
    plt.ylabel("z[1]")
    plt.show()


(x_train, y_train), _ = keras.datasets.mnist.load_data()
x_train = np.expand_dims(x_train, -1).astype("float32") / 255

plot_label_clusters(encoder, decoder, x_train, y_train)
```


z[1]
z[0]
4
3
2
1
0
−1
−2
−3
−4
−2.5
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
9
8
7
6
5
4
3
2
1
0