Demystifying TensorFlow

Instructor: Dr. Santosh Chapaneri ¶

Hands-on Workshop at RAIT (Mumbai), Jan 2023

TensorFlow API

In [ ]:

```
import tensorflow as tf
tf.__version__
```

Out[ ]:

```
'2.9.2'
```

In [ ]:

```
# Making a constant tensor A, that does not change
A = tf.constant([[3, 2],
                 [5, 2]])

# Making another tensor B
B = tf.constant([[9, 5],
                 [1, 3]])

# Using TF1
# Init session, run session, close session

# Using TF2, sessions not needed
print(A)
print(A.numpy())
print(B.numpy())
```

```
tf.Tensor(
[[3 2]
 [5 2]], shape=(2, 2), dtype=int32)
[[3 2]
 [5 2]]
[[9 5]
 [1 3]]
```

In [ ]:

```
# Making a Variable tensor VA, which can change
VA = tf.Variable([[3, 2],
                  [5, 2],
                  [10, 3]])

print(VA.numpy())
```

```
[[ 3  2]
 [ 5  2]
 [10  3]]
```

In [ ]:

```
# Concatenate columns
AB_concatenated_c = tf.concat(values = [A, B], axis = 1) # horizontal stacking
print(AB_concatenated_c.numpy()) # 2 x 4

# Concatenate rows
AB_concatenated_r = tf.concat(values = [A, B], axis = 0) # vertical stacking
print(AB_concatenated_r.numpy()) # 4 x 2
```

```
[[3 2 9 5]
 [5 2 1 3]]
[[3 2]
 [5 2]
 [9 5]
 [1 3]]
```

In [ ]:

```
# Making a tensor filled with zeros; shape=[rows, columns]
tensor_z = tf.zeros(shape=[3, 4], dtype=tf.int32)
print(tensor_z.numpy())

# Making a tensor filled with ones with data type of float32
tensor_o = tf.ones(shape=[5, 3], dtype=tf.float32)
print(tensor_o.numpy())

# Create a 3 x 2 x 2 tensor, all values should be 4
# 3 matrices, each of size 2 x 2
tensor_4 = 4 * tf.ones(shape=[3, 2, 2], dtype=tf.float32)
print(tensor_4.numpy())
```

```
[[0 0 0 0]
 [0 0 0 0]
 [0 0 0 0]]
[[1. 1. 1.]
 [1. 1. 1.]
 [1. 1. 1.]
 [1. 1. 1.]
 [1. 1. 1.]]
[[[4. 4.]
  [4. 4.]]

 [[4. 4.]
  [4. 4.]]

 [[4. 4.]
  [4. 4.]]]
```

In [ ]:

```
# Reshape
t1 = tf.constant([[3, 2],
                  [5, 2],
                  [9, 5],
                  [1, 3]]) #

# Reshaping the tensor into a shape of: shape = [rows, columns]
t2 = tf.reshape(tensor = t1, shape = [1, 8])

print(t1.numpy())
print(t2.numpy())
```

```
[[3 2]
 [5 2]
 [9 5]
 [1 3]]
[[3 2 5 2 9 5 1 3]]
```

In [ ]:

```
# Type cast
t1 = tf.constant([[3.1, 2.8],
                  [5.2, 2.3],
                  [9.7, 5.5],
                  [1.1, 3.4]],
                  dtype=tf.float32)

t2 = tf.cast(t1, tf.int32)

print(t1.numpy())
print(t2.numpy())
```

```
[[3.1 2.8]
 [5.2 2.3]
 [9.7 5.5]
 [1.1 3.4]]
[[3 2]
 [5 2]
 [9 5]
 [1 3]]
```

Tensor Operations

In [ ]:

```
a = tf.constant(2.0)
b = tf.constant(3.0)
c = tf.constant(5.0)

add = tf.add(a, b)
sub = tf.subtract(a, b)
mul = tf.multiply(a, b)
div = tf.divide(a, b)
# All without using the session
print(f'add = {add.numpy()}')
print(f'sub = {sub.numpy()}')
print(f'mul = {mul.numpy()}')
print(f'div = {div.numpy()}')
```

```
add = 5.0
sub = -1.0
mul = 6.0
div = 0.6666666865348816
```

In [ ]:

```
# "Reduce" operations

a = tf.constant(2.0) # 1 x 1 scalar, rank-1 tensor
b = tf.constant(3.0)
c = tf.constant(5.0)

mymean = tf.reduce_mean([a, b, c]) # 2, 3, 5 => mean = 10/3 = 3.33
mysum = tf.reduce_sum([a, b, c]) # 2, 3, 5 => sum = 10

print(f'mean = {mymean.numpy()}')
print(f'sum = {mysum.numpy()}')
```

```
mean = 3.3333332538604736
sum = 10.0
```

In [ ]:

```
# "Reduce" operations

a = tf.constant([[10.0, 20.0], [30.0, 40.0]]) # 2 x 2, rank-2 tensor
b = tf.constant([[20.0, 30.0], [40.0, 50.0]])
c = tf.constant([[30.0, 40.0], [50.0, 60.0]])

mysum = tf.reduce_sum([a, b, c]) # axis not specified, all elements considered, scalar
mymean = tf.reduce_mean([a, b, c])
print(f'sum = \n {mysum.numpy()}')  # 420
print(f'mean = \n {mymean.numpy()}') # 35 = 420/12
```

```
sum =
 420.0
mean =
 35.0
```

In [ ]:

```
mysum_0 = tf.reduce_sum([a, b, c], axis=0)
mymean_0 = tf.reduce_mean([a, b, c], axis=0) # 0: column, 1: row

print(f'sum_0 = \n {mysum_0.numpy()}')
print(f'mean_0 = \n {mymean_0.numpy()}') #
```

```
sum_0 =
 [[ 60.  90.]
 [120. 150.]]
mean_0 =
 [[20. 30.]
 [40. 50.]]
```

In [ ]:

```
mysum_1 = tf.reduce_sum([a, b, c], axis=1)
mymean_1 = tf.reduce_mean([a, b, c], axis=1)

print(f'sum_1 = \n {mysum_1.numpy()}')
print(f'mean_1 = \n {mymean_1.numpy()}')
```

```
sum_1 =
 [[ 40.  60.]
 [ 60.  80.]
 [ 80. 100.]]
mean_1 =
 [[20. 30.]
 [30. 40.]
 [40. 50.]]
```

Linear Algebra Operations

- Transpose tensor with `tf.transpose`
- Matrix Multiplication with `tf.matmul`
- Element-wise multiplication with `tf.multiply`
- Identity Matrix with `tf.eye`
- Determinant with `tf.linalg.det`
- Dot Product with `tf.tensordot`

In [ ]:

```
A = tf.constant([[3, 7],
                 [1, 9]])

AT = tf.transpose(A)

print(f'Transposed matrix is \n {AT.numpy()}')
```

```
Transposed matrix is
 [[3 1]
 [7 9]]
```

In [ ]:

```
A = tf.constant([[3, 7],
                 [1, 9]]) # 2 x 2

v = tf.constant([[5],
                 [2]]) # 2 x 1

# Matrix multiplication of A and v
Av = tf.matmul(A, v) # 2 x 1

# Av = tf.matmul(v, A) , will this work? Error

print(f'Matrix multiplication is \n {Av.numpy()}')
```

```
Matrix multiplication is
 [[29]
 [23]]
```

In [ ]:

```
# Element-wise multiplication - Hadamard Product

Av2 = tf.multiply(A, v) # 2 x 2

print(f'Element-wise multiplication is \n {Av2.numpy()}')
```

```
Element-wise multiplication is
 [[15 35]
 [ 2 18]]
```

In [ ]:

```
# Determinant

A = tf.constant([[3, 7],
                 [1, 9]]) # 2 x 2, integers

A = tf.cast(A, tf.float32)

det_A = tf.linalg.det(A)

print(f'Determinant is \n {det_A.numpy()}')
```

```
Determinant is
 20.000001907348633
```

In [ ]:

```
# Dot product

A = tf.constant([[32, 83, 5],
                 [17, 23, 10],
                 [75, 39, 52]]) # 3 x 3

B = tf.constant([[28, 57, 20],
                 [91, 10, 95],
                 [37, 13, 45]])

dot_AB = tf.tensordot(A, B, axes=1) # 3 x 3, matrix mult along axes = 1, matmul

print(f'Dot product is \n {dot_AB.numpy()}')
```

```
Dot product is
 [[8634 2719 8750]
 [2939 1329 2975]
 [7573 5341 7545]]
```

TF2 supports automatic differentiation

y = x^3 = f(x)

dy/dx = 3 x^2; f'(x)

d2y/dx2 = 6x; f''(x)

f'''(x)

ML and DL algorithms are optimization based; maximize or minimize objective function

In general, f(x) can be complex; difficult to write the derivatives manually

TF Gradient Tape

`tf.GradientTape` allows us to track TensorFlow computations and calculate gradients w.r.t. some given variables.

In [ ]:

```
x = tf.constant(4.0)

# By default, GradientTape doesn't track constants,
# so we must instruct it to with 'watch'
with tf.GradientTape() as tape:
    tape.watch(x)
    y = x**3 # f(x), objective function

grad = tape.gradient(y, x) # dy/dx = 3 x^2 = 3 x 4^2 = 48
print(f'Gradient of y wrt x at x = {x} is {grad.numpy()}')
```

```
Gradient of y wrt x at x = 4.0 is 48.0
```

In [ ]:

```
# Variables are automatically watched

x = tf.Variable(4.0, trainable=True)
with tf.GradientTape() as tape:
    y = x**3 # f(x)
    # No need to watch the variable

grad = tape.gradient(y, x) # dy/dx
print(f'Gradient of y wrt x at x = {x.numpy()} is {grad.numpy()}')
```

```
Gradient of y wrt x at x = 4.0 is 48.0
```

In [ ]:

```
# Higher order derivatives

x = tf.Variable(4.0, trainable=True)
# dy/dx, and dy2/dx2

with tf.GradientTape() as tape1:
    with tf.GradientTape() as tape2:
        y = x ** 3 # f(x)
    order_1 = tape2.gradient(y, x) # dy/dx = 3 x^2 = 3 x 16 = 48

order_2 = tape1.gradient(order_1, x) # dy2/dx2 = 6x = 24
print(f'1st order gradient of y wrt x at x = {x.numpy()} is {order_1.numpy()}')
print(f'2nd order gradient of y wrt x at x = {x.numpy()} is {order_2.numpy()}')
```

```
1st order gradient of y wrt x at x = 4.0 is 48.0
2nd order gradient of y wrt x at x = 4.0 is 24.0
```

In [ ]:

```
# Persistent: store the information internally, do not release on execution

a = tf.Variable(5.0, trainable=True)
b = tf.Variable(3.0, trainable=True)

with tf.GradientTape(persistent=True) as tape: # reuse the same tape
    y1 = a ** 2
    y2 = b ** 3

# reusing the same tape for multiple operations
grad_a = tape.gradient(y1, a) # dy1/da = 10
grad_b = tape.gradient(y2, b) # dy2/db = 3 x 9 = 27
print(f'Gradient of y1 wrt a at a = {a.numpy()} is {grad_a.numpy()}')
print(f'Gradient of y2 wrt b at b = {b.numpy()} is {grad_b.numpy()}')
```

```
Gradient of y1 wrt a at a = 5.0 is 10.0
Gradient of y2 wrt b at b = 3.0 is 27.0
```

Usage of GradientTape

- Linear Regression
- y = mx + c = Wx + b
- Goal: to find W and b
- Minimize loss between true y and pred y

In [ ]:

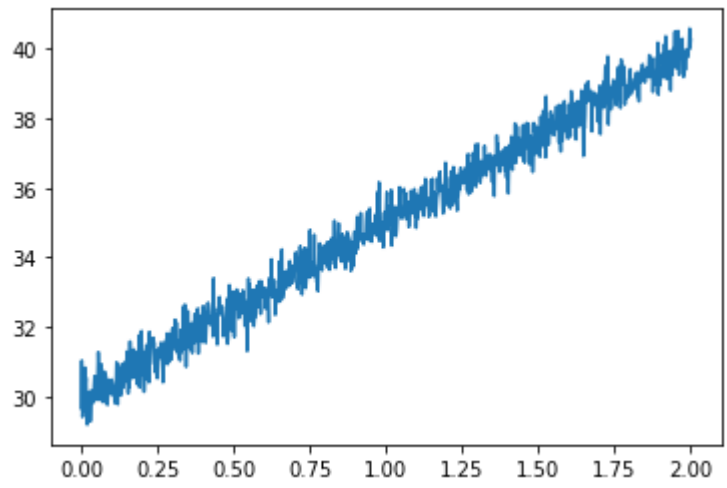
```
import numpy as np
x = tf.constant(value = np.linspace(0, 2, 1000), dtype = tf.float32)
y = 5 * x + 30 # True y
y = y + tf.random.normal(tf.shape(y), 0, 0.4, tf.float32)
# pred y: y = W x + b
```

In [ ]:

```
import matplotlib.pyplot as plt
plt.plot(x, y)
```

Out[ ]:

```
[<matplotlib.lines.Line2D at 0x7fe5c0538e50>]
```



In [ ]:

```
# create variables for weight and bias
W = tf.Variable(initial_value=0, trainable=True, name="weight", dtype=tf.float32)
b = tf.Variable(initial_value=0, trainable=True, name="bias", dtype=tf.float32 )
```

In [ ]:

```
# Gradient function to calculate gradients and loss
def mygrad(x, y, W, b):
  with tf.GradientTape() as tape:
    tape.watch(W)
    tape.watch(b)
    y_pred = W * x + b # Predicted output
    loss = tf.reduce_sum(input_tensor=( W * x + b - y)**2) # total loss

  grad = tape.gradient(loss, [W, b]) # grad will be a tuple: (dloss/dW, dloss/db)
  return grad, loss
```

In [ ]:

```
STEPS = 100 # how many times to iterate
LEARNING_RATE = .0001

for step in range(STEPS):
  # Calculate gradients and loss
  (d_W, d_b), loss = mygrad(x, y, W, b)

  # update weights - gradient descent
  W = W - d_W * LEARNING_RATE
  b = b - d_b * LEARNING_RATE

  # print STEP number and loss
  print("STEP: {} Loss: {}".format(step, loss))

# Print the Final Loss, Weights and bias
print("STEP: {} Loss: {}".format(STEPS, loss))
print("W:{}".format(round(float(W), 4)))
print("b:{}".format(round(float(b), 4)))
```

```
STEP: 0 Loss: 1233199.25
STEP: 1 Loss: 428897.6875
STEP: 2 Loss: 170989.5
STEP: 3 Loss: 86915.90625
STEP: 4 Loss: 58230.796875
STEP: 5 Loss: 47272.04296875
STEP: 6 Loss: 42059.53515625
STEP: 7 Loss: 38778.0625
STEP: 8 Loss: 36208.46875
STEP: 9 Loss: 33957.26171875
STEP: 10 Loss: 31893.79296875
STEP: 11 Loss: 29971.3046875
STEP: 12 Loss: 28170.033203125
STEP: 13 Loss: 26479.09375
STEP: 14 Loss: 24890.6875
STEP: 15 Loss: 23398.26171875
STEP: 16 Loss: 21995.916015625
STEP: 17 Loss: 20678.1796875
STEP: 18 Loss: 19439.939453125
STEP: 19 Loss: 18276.396484375
STEP: 20 Loss: 17183.0390625
STEP: 21 Loss: 16155.63671875
STEP: 22 Loss: 15190.208984375
STEP: 23 Loss: 14283.0185546875
STEP: 24 Loss: 13430.5517578125
STEP: 25 Loss: 12629.5087890625
STEP: 26 Loss: 11876.787109375
STEP: 27 Loss: 11169.4716796875
STEP: 28 Loss: 10504.822265625
STEP: 29 Loss: 9880.265625
STEP: 30 Loss: 9293.3828125
STEP: 31 Loss: 8741.9052734375
STEP: 32 Loss: 8223.6943359375
STEP: 33 Loss: 7736.7421875
STEP: 34 Loss: 7279.16455078125
STEP: 35 Loss: 6849.189453125
STEP: 36 Loss: 6445.1494140625
STEP: 37 Loss: 6065.48291015625
STEP: 38 Loss: 5708.7177734375
STEP: 39 Loss: 5373.47314453125
STEP: 40 Loss: 5058.451171875
STEP: 41 Loss: 4762.43212890625
STEP: 42 Loss: 4484.271484375
STEP: 43 Loss: 4222.88916015625
STEP: 44 Loss: 3977.274658203125
STEP: 45 Loss: 3746.476318359375
STEP: 46 Loss: 3529.59912109375
STEP: 47 Loss: 3325.80517578125
STEP: 48 Loss: 3134.304443359375
STEP: 49 Loss: 2954.355712890625
STEP: 50 Loss: 2785.26220703125
STEP: 51 Loss: 2626.369140625
STEP: 52 Loss: 2477.061279296875
STEP: 53 Loss: 2336.760009765625
STEP: 54 Loss: 2204.922607421875
STEP: 55 Loss: 2081.03759765625
STEP: 56 Loss: 1964.624755859375
STEP: 57 Loss: 1855.2349853515625
STEP: 58 Loss: 1752.4427490234375
STEP: 59 Loss: 1655.8525390625
STEP: 60 Loss: 1565.087646484375
```

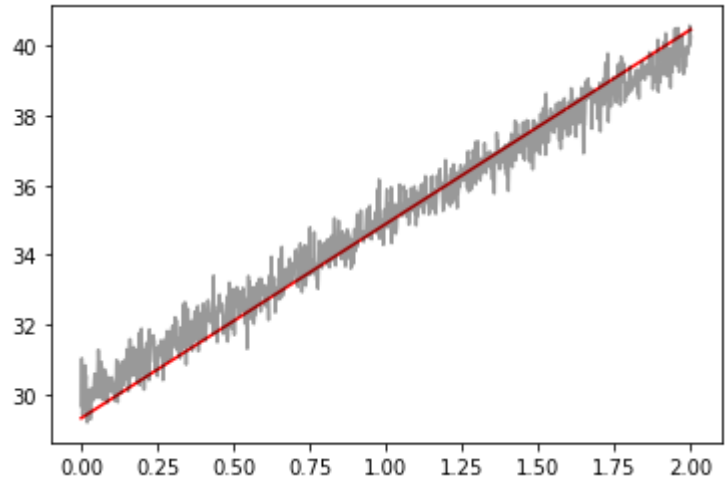
```
STEP: 61 Loss: 1479.7984619140625
STEP: 62 Loss: 1399.6533203125
STEP: 63 Loss: 1324.34326171875
STEP: 64 Loss: 1253.576416015625
STEP: 65 Loss: 1187.0787353515625
STEP: 66 Loss: 1124.592041015625
STEP: 67 Loss: 1065.8746337890625
STEP: 68 Loss: 1010.6986083984375
STEP: 69 Loss: 958.851318359375
STEP: 70 Loss: 910.1316528320312
STEP: 71 Loss: 864.3507080078125
STEP: 72 Loss: 821.3318481445312
STEP: 73 Loss: 780.9078369140625
STEP: 74 Loss: 742.92236328125
STEP: 75 Loss: 707.2280883789062
STEP: 76 Loss: 673.6869506835938
STEP: 77 Loss: 642.1688232421875
STEP: 78 Loss: 612.5526123046875
STEP: 79 Loss: 584.7227783203125
STEP: 80 Loss: 558.5713500976562
STEP: 81 Loss: 533.9979248046875
STEP: 82 Loss: 510.9064636230469
STEP: 83 Loss: 489.207763671875
STEP: 84 Loss: 468.8183288574219
STEP: 85 Loss: 449.658447265625
STEP: 86 Loss: 431.6545715332031
STEP: 87 Loss: 414.73663330078125
STEP: 88 Loss: 398.83935546875
STEP: 89 Loss: 383.9007873535156
STEP: 90 Loss: 369.86346435546875
STEP: 91 Loss: 356.67303466796875
STEP: 92 Loss: 344.2781982421875
STEP: 93 Loss: 332.6313171386719
STEP: 94 Loss: 321.6866149902344
STEP: 95 Loss: 311.40234375
STEP: 96 Loss: 301.7384948730469
STEP: 97 Loss: 292.6576232910156
STEP: 98 Loss: 284.12445068359375
STEP: 99 Loss: 276.10614013671875
STEP: 100 Loss: 276.10614013671875
W:5.5747
b:29.3183
```

In [ ]:

```
y_pred = W * x + b

plt.plot(x, y_pred, 'r')
plt.plot(x, y, 'k', alpha=0.4)
```

Out[ ]:

```
[<matplotlib.lines.Line2D at 0x7fe5c02fd280>]
```



Gradients wrt a Model

In [ ]:

```
layer = tf.keras.layers.Dense(2, activation='relu') # fully connected layer
# to leanrn any non-linearities
x = tf.constant([[1., 2., 3.]])

# 3 neurons - 2 neurons, total connections = 6
# how many parameters to train: 6 weights, 2 biases = 8 values to be updated

with tf.GradientTape() as tape:
  # Forward pass
  y = layer(x) # Functional syntax, output y
  loss = tf.reduce_mean(y**2) # y could be having multiple values

# Calculate gradients with respect to every trainable variable
grad = tape.gradient(loss, layer.trainable_variables)
```

In [ ]:

```
grad
```

Out[ ]:

```
[<tf.Tensor: shape=(3, 2), dtype=float32, numpy=
 array([[0., 0.],
        [0., 0.],
        [0., 0.]], dtype=float32)>,
 <tf.Tensor: shape=(2,), dtype=float32, numpy=array([0., 0.], dtype=float3
2)>]
```

In [ ]:

```
for var, g in zip(layer.trainable_variables, grad):
  print(f'{var.name}, shape: {g.shape}')
```

```
dense/kernel:0, shape: (3, 2)
dense/bias:0, shape: (2,)
```

Assignment

1. Create two random 0-d tensors x and y of uniform distribution. Create a TensorFlow object that returns x + y if x > y, and x - y otherwise. Hint: look up tf.cond()

In [ ]:

```
x = tf.random.uniform([])  # Empty array as shape creates a scalar
y = tf.random.uniform([])

out = tf.cond(tf.greater(x, y), lambda: tf.add(x, y), lambda: tf.subtract(x, y))
```

2. Create a tensor x of the value [[0, -2, -1], [0, 1, 2]] and y as a tensor of zeros with the same shape as x. Return a boolean tensor that yields True if x equals y element-wise. Hint: Look up tf.zeros_like() and tf.equal().

In [ ]:

```
x = tf.constant([[0, -2, -1], [0, 1, 2]])
y = tf.zeros_like(x)
out = tf.equal(x, y)
```

3. Create the tensor x of value

[29.05088806, 27.61298943, 31.19073486, 29.35532951,

30.97266006, 26.67541885, 38.08450317, 20.74983215,

34.94445419, 34.45999146, 29.06485367, 36.01657104,

27.88236427, 20.56035233, 30.20379066, 29.51215172,

33.71149445, 28.59134293, 36.05556488, 28.66994858].

Get the indices of elements in x whose values are greater than 30. Hint: Use tf.where(). Then extract elements whose values are greater than 30. Hint: Use tf.gather().

In [ ]:

```
x = tf.constant([29.05088806,  27.61298943,  31.19073486,  29.35532951,
                 30.97266006,  26.67541885,  38.08450317,  20.74983215,
                 34.94445419,  34.45999146,  29.06485367,  36.01657104,
                 27.88236427,  20.56035233,  30.20379066,  29.51215172,
                 33.71149445,  28.59134293,  36.05556488,  28.66994858])

indices = tf.where(x > 30)
out = tf.gather(x, indices)
```

4. Create a diagnoal 2-d tensor of size 6 x 6 with the diagonal values of 1, 2, ..., 6. Hint: Use tf.range() and tf.diag().

In [ ]:

```
values = tf.range(1, 7)
out = tf.linalg.diag(values)
```

5. Create a random 2-d tensor of size 10 x 10 from any distribution. Calculate its determinant.

In [ ]:

```
m = tf.random.normal([10, 10], mean=10, stddev=1)
out = tf.linalg.det(m)
```

6. Create tensor x with value [5, 2, 3, 5, 10, 6, 2, 3, 4, 2, 1, 1, 0, 9]. Return the unique elements in x. Hint: use tf.unique(). Keep in mind that tf.unique() returns a tuple.

In [ ]:

```
x = tf.constant([5, 2, 3, 5, 10, 6, 2, 3, 4, 2, 1, 1, 0, 9])
unique_values, indices = tf.unique(x)
```

7. Create two tensors x and y of shape 300 from any normal distribution, as long as they are from the same distribution. Use tf.cond() to return:

- The mean squared error of (x - y) if the average of all elements in (x - y) is negative, or

- The sum of absolute value of all elements in the tensor (x - y) otherwise.

In [ ]:

```
x = tf.random.normal([300], mean=5, stddev=1)
y = tf.random.normal([300], mean=5, stddev=1)
average = tf.reduce_mean(x - y)

def f1():
  return tf.reduce_mean(tf.square(x - y))

def f2():
  return tf.reduce_sum(tf.abs(x - y))

out = tf.cond(average < 0, f1, f2)
```

Loading and Parsing Data with TF2

In [ ]:

```
# from_tensor_slices

data = [8, 3, 0, 8, 2, 1] # List

dataset = tf.data.Dataset.from_tensor_slices(data)
# dataset is a tf Data object
for elem in dataset:
  print(elem.numpy())
```

```
8
3
0
8
2
1
```

In [ ]:

```
# from_tensor_slices
data2 = [[5, 10], [3, 6]] # matrix

dataset2 = tf.data.Dataset.from_tensor_slices(data2)
for elem in dataset2:
  print(elem.numpy())
```

```
[ 5 10]
[3 6]
```

In [ ]:

```
# shuffle

data = [10, 20, 30, 40, 50, 60]

dataset = tf.data.Dataset.from_tensor_slices(data).shuffle(len(data))
for elem in dataset:
  print(elem.numpy())
```

```
20
60
50
30
10
40
```

In [ ]:

```
# repeat

data = [10, 20, 30, 40, 50, 60]

dataset = tf.data.Dataset.from_tensor_slices(data).repeat(2)
for elem in dataset:
  print(elem.numpy())
```

```
10
20
30
40
50
60
10
20
30
40
50
60
```

In [ ]:

```
# repeat and shuffle

data = [10, 20, 30, 40, 50, 60]

dataset = tf.data.Dataset.from_tensor_slices(data).repeat(2).shuffle(len(data))
# repeat then shuffle => lose the epoch boundary
for elem in dataset:
  print(elem.numpy())
```

```
20
10
60
30
30
10
40
20
40
50
50
60
```

In [ ]:

```
# shuffle and repeat

data = [10, 20, 30, 40, 50, 60]

dataset = tf.data.Dataset.from_tensor_slices(data).shuffle(len(data)).repeat(2)
# shuffle then repeat => preserve the epoch boundary
for elem in dataset:
  print(elem.numpy())
```

```
10
60
40
30
50
20
30
10
60
20
40
50
```

In [ ]:

```
# batching the data

data = [10, 20, 30, 40, 50, 60]

dataset = tf.data.Dataset.from_tensor_slices(data).shuffle(len(data)).repeat(3).batch(b
atch_size = 4)
# each batch has 4 elements; total 12 values, so 3 batches

# to increase batches, increase value of repeat
for elem in dataset:
  print(elem.numpy())
```

```
[60 20 50 10]
[30 40 30 60]
[10 20 40 50]
[50 10 30 60]
[40 20]
```

In [ ]:

```
# to increase batches, increase value of repeat

data = [10, 20, 30, 40, 50, 60]
dataset = tf.data.Dataset.from_tensor_slices(data).shuffle(len(data)).repeat(3).batch(b
atch_size = 4)
# epoch not preserved
for elem in dataset:
  print(elem.numpy())
```

```
[60 40 10 30]
[50 20 50 20]
[60 10 30 40]
[40 10 20 30]
[60 50]
```

Pre-fetch

"When the GPU is working on forward / backward propagation on the current batch, we want the CPU to process the next batch of data so that it is immediately ready. As the most expensive part of the computer, we want the GPU to be fully used all the time during training. We call this consumer / producer overlap, where the consumer is the GPU and the producer is the CPU.

With tf.data, you can do this with a simple call to dataset.prefetch(1) at the end of the pipeline (after batching). This will always prefetch one batch of data and make sure that there is always one ready.

In [ ]:

```
# pre-fetch the data => always keep one batch ready in memory

data = [10, 20, 30, 40, 50, 60]

dataset = tf.data.Dataset.from_tensor_slices(data).shuffle(len(data)).repeat(3).batch(b
atch_size = 4)
dataset = dataset.prefetch(buffer_size = 1)
# each batch has 4 elements up to the len(data), then next batch will begin
for elem in dataset:
  print(elem.numpy())
```

```
[40 20 60 30]
[10 50 30 20]
[50 10 40 60]
[40 10 50 30]
[60 20]
```

Load Numpy Arrays

- Build a data pipeline over numpy arrays

In [ ]:

```
import numpy as np
```

In [ ]:

```
# Create a toy dataset (even and odd numbers, with respective labels of 0 and 1)

evens = np.arange(0, 100, step=2, dtype=np.int32)
evens_label = np.zeros(50, dtype=np.int32)

odds = np.arange(1, 100, step=2, dtype=np.int32)
odds_label = np.ones(50, dtype=np.int32)

# Concatenate arrays
features = np.concatenate([evens, odds])
labels = np.concatenate([evens_label, odds_label])
```

In [ ]:

```
print(features)
print(labels)
```

```
[ 0  2  4  6  8 10 12 14 16 18 20 22 24 26 28 30 32 34 36 38 40 42 44 46
 48 50 52 54 56 58 60 62 64 66 68 70 72 74 76 78 80 82 84 86 88 90 92 94
 96 98  1  3  5  7  9 11 13 15 17 19 21 23 25 27 29 31 33 35 37 39 41 43
 45 47 49 51 53 55 57 59 61 63 65 67 69 71 73 75 77 79 81 83 85 87 89 91
 93 95 97 99]
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1]
```

In [ ]:

```
# Load a numpy array using tf data api with `from_tensor_slices`

data = tf.data.Dataset.from_tensor_slices((features, labels))
for feat, lab in data:
  print(f'Feature = {feat.numpy()} and Label = {lab.numpy()}')
```

```
Feature = 0 and Label = 0
Feature = 2 and Label = 0
Feature = 4 and Label = 0
Feature = 6 and Label = 0
Feature = 8 and Label = 0
Feature = 10 and Label = 0
Feature = 12 and Label = 0
Feature = 14 and Label = 0
Feature = 16 and Label = 0
Feature = 18 and Label = 0
Feature = 20 and Label = 0
Feature = 22 and Label = 0
Feature = 24 and Label = 0
Feature = 26 and Label = 0
Feature = 28 and Label = 0
Feature = 30 and Label = 0
Feature = 32 and Label = 0
Feature = 34 and Label = 0
Feature = 36 and Label = 0
Feature = 38 and Label = 0
Feature = 40 and Label = 0
Feature = 42 and Label = 0
Feature = 44 and Label = 0
Feature = 46 and Label = 0
Feature = 48 and Label = 0
Feature = 50 and Label = 0
Feature = 52 and Label = 0
Feature = 54 and Label = 0
Feature = 56 and Label = 0
Feature = 58 and Label = 0
Feature = 60 and Label = 0
Feature = 62 and Label = 0
Feature = 64 and Label = 0
Feature = 66 and Label = 0
Feature = 68 and Label = 0
Feature = 70 and Label = 0
Feature = 72 and Label = 0
Feature = 74 and Label = 0
Feature = 76 and Label = 0
Feature = 78 and Label = 0
Feature = 80 and Label = 0
Feature = 82 and Label = 0
Feature = 84 and Label = 0
Feature = 86 and Label = 0
Feature = 88 and Label = 0
Feature = 90 and Label = 0
Feature = 92 and Label = 0
Feature = 94 and Label = 0
Feature = 96 and Label = 0
Feature = 98 and Label = 0
Feature = 1 and Label = 1
Feature = 3 and Label = 1
Feature = 5 and Label = 1
Feature = 7 and Label = 1
Feature = 9 and Label = 1
Feature = 11 and Label = 1
Feature = 13 and Label = 1
Feature = 15 and Label = 1
Feature = 17 and Label = 1
Feature = 19 and Label = 1
Feature = 21 and Label = 1
```

Feature = 23 and Label = 1
Feature = 25 and Label = 1
Feature = 27 and Label = 1
Feature = 29 and Label = 1
Feature = 31 and Label = 1
Feature = 33 and Label = 1
Feature = 35 and Label = 1
Feature = 37 and Label = 1
Feature = 39 and Label = 1
Feature = 41 and Label = 1
Feature = 43 and Label = 1
Feature = 45 and Label = 1
Feature = 47 and Label = 1
Feature = 49 and Label = 1
Feature = 51 and Label = 1
Feature = 53 and Label = 1
Feature = 55 and Label = 1
Feature = 57 and Label = 1
Feature = 59 and Label = 1
Feature = 61 and Label = 1
Feature = 63 and Label = 1
Feature = 65 and Label = 1
Feature = 67 and Label = 1
Feature = 69 and Label = 1
Feature = 71 and Label = 1
Feature = 73 and Label = 1
Feature = 75 and Label = 1
Feature = 77 and Label = 1
Feature = 79 and Label = 1
Feature = 81 and Label = 1
Feature = 83 and Label = 1
Feature = 85 and Label = 1
Feature = 87 and Label = 1
Feature = 89 and Label = 1
Feature = 91 and Label = 1
Feature = 93 and Label = 1
Feature = 95 and Label = 1
Feature = 97 and Label = 1
Feature = 99 and Label = 1

In [ ]:

```
# Shuffle, Repeat

data = tf.data.Dataset.from_tensor_slices((features, labels))
data = data.shuffle(len(data))
data = data.repeat(10) # repeat 10 times
for feat, lab in data:
  print(f'Feature = {feat.numpy()} and Label = {lab.numpy()}')
```

Feature = 11 and Label = 1
Feature = 68 and Label = 0
Feature = 74 and Label = 0
Feature = 94 and Label = 0
Feature = 43 and Label = 1
Feature = 40 and Label = 0
Feature = 73 and Label = 1
Feature = 84 and Label = 0
Feature = 55 and Label = 1
Feature = 38 and Label = 0
Feature = 47 and Label = 1
Feature = 61 and Label = 1
Feature = 97 and Label = 1
Feature = 0 and Label = 0
Feature = 81 and Label = 1
Feature = 71 and Label = 1
Feature = 88 and Label = 0
Feature = 1 and Label = 1
Feature = 48 and Label = 0
Feature = 4 and Label = 0
Feature = 99 and Label = 1
Feature = 83 and Label = 1
Feature = 9 and Label = 1
Feature = 96 and Label = 0
Feature = 82 and Label = 0
Feature = 7 and Label = 1
Feature = 63 and Label = 1
Feature = 46 and Label = 0
Feature = 27 and Label = 1
Feature = 60 and Label = 0
Feature = 65 and Label = 1
Feature = 45 and Label = 1
Feature = 79 and Label = 1
Feature = 72 and Label = 0
Feature = 67 and Label = 1
Feature = 44 and Label = 0
Feature = 51 and Label = 1
Feature = 13 and Label = 1
Feature = 36 and Label = 0
Feature = 86 and Label = 0
Feature = 58 and Label = 0
Feature = 20 and Label = 0
Feature = 62 and Label = 0
Feature = 2 and Label = 0
Feature = 89 and Label = 1
Feature = 15 and Label = 1
Feature = 76 and Label = 0
Feature = 41 and Label = 1
Feature = 35 and Label = 1
Feature = 32 and Label = 0
Feature = 70 and Label = 0
Feature = 24 and Label = 0
Feature = 14 and Label = 0
Feature = 31 and Label = 1
Feature = 12 and Label = 0
Feature = 69 and Label = 1
Feature = 85 and Label = 1
Feature = 50 and Label = 0
Feature = 42 and Label = 0
Feature = 87 and Label = 1
Feature = 80 and Label = 0

Feature = 91 and Label = 1
Feature = 16 and Label = 0
Feature = 90 and Label = 0
Feature = 98 and Label = 0
Feature = 10 and Label = 0
Feature = 56 and Label = 0
Feature = 3 and Label = 1
Feature = 34 and Label = 0
Feature = 28 and Label = 0
Feature = 66 and Label = 0
Feature = 21 and Label = 1
Feature = 77 and Label = 1
Feature = 22 and Label = 0
Feature = 54 and Label = 0
Feature = 29 and Label = 1
Feature = 30 and Label = 0
Feature = 19 and Label = 1
Feature = 57 and Label = 1
Feature = 75 and Label = 1
Feature = 25 and Label = 1
Feature = 78 and Label = 0
Feature = 5 and Label = 1
Feature = 8 and Label = 0
Feature = 92 and Label = 0
Feature = 49 and Label = 1
Feature = 93 and Label = 1
Feature = 23 and Label = 1
Feature = 37 and Label = 1
Feature = 39 and Label = 1
Feature = 52 and Label = 0
Feature = 33 and Label = 1
Feature = 6 and Label = 0
Feature = 59 and Label = 1
Feature = 64 and Label = 0
Feature = 26 and Label = 0
Feature = 18 and Label = 0
Feature = 95 and Label = 1
Feature = 17 and Label = 1
Feature = 53 and Label = 1
Feature = 27 and Label = 1
Feature = 36 and Label = 0
Feature = 23 and Label = 1
Feature = 87 and Label = 1
Feature = 34 and Label = 0
Feature = 76 and Label = 0
Feature = 7 and Label = 1
Feature = 91 and Label = 1
Feature = 1 and Label = 1
Feature = 65 and Label = 1
Feature = 84 and Label = 0
Feature = 43 and Label = 1
Feature = 10 and Label = 0
Feature = 13 and Label = 1
Feature = 61 and Label = 1
Feature = 21 and Label = 1
Feature = 35 and Label = 1
Feature = 19 and Label = 1
Feature = 77 and Label = 1
Feature = 48 and Label = 0
Feature = 32 and Label = 0
Feature = 37 and Label = 1

Feature = 9 and Label = 1
Feature = 66 and Label = 0
Feature = 5 and Label = 1
Feature = 73 and Label = 1
Feature = 70 and Label = 0
Feature = 62 and Label = 0
Feature = 97 and Label = 1
Feature = 78 and Label = 0
Feature = 82 and Label = 0
Feature = 38 and Label = 0
Feature = 67 and Label = 1
Feature = 56 and Label = 0
Feature = 95 and Label = 1
Feature = 57 and Label = 1
Feature = 75 and Label = 1
Feature = 11 and Label = 1
Feature = 15 and Label = 1
Feature = 58 and Label = 0
Feature = 45 and Label = 1
Feature = 92 and Label = 0
Feature = 42 and Label = 0
Feature = 96 and Label = 0
Feature = 69 and Label = 1
Feature = 18 and Label = 0
Feature = 40 and Label = 0
Feature = 51 and Label = 1
Feature = 4 and Label = 0
Feature = 41 and Label = 1
Feature = 33 and Label = 1
Feature = 60 and Label = 0
Feature = 68 and Label = 0
Feature = 88 and Label = 0
Feature = 24 and Label = 0
Feature = 99 and Label = 1
Feature = 54 and Label = 0
Feature = 2 and Label = 0
Feature = 93 and Label = 1
Feature = 94 and Label = 0
Feature = 0 and Label = 0
Feature = 64 and Label = 0
Feature = 46 and Label = 0
Feature = 26 and Label = 0
Feature = 98 and Label = 0
Feature = 8 and Label = 0
Feature = 74 and Label = 0
Feature = 90 and Label = 0
Feature = 12 and Label = 0
Feature = 6 and Label = 0
Feature = 16 and Label = 0
Feature = 44 and Label = 0
Feature = 85 and Label = 1
Feature = 25 and Label = 1
Feature = 63 and Label = 1
Feature = 17 and Label = 1
Feature = 39 and Label = 1
Feature = 71 and Label = 1
Feature = 3 and Label = 1
Feature = 22 and Label = 0
Feature = 14 and Label = 0
Feature = 55 and Label = 1
Feature = 50 and Label = 0

Feature = 89 and Label = 1
Feature = 28 and Label = 0
Feature = 79 and Label = 1
Feature = 53 and Label = 1
Feature = 59 and Label = 1
Feature = 83 and Label = 1
Feature = 72 and Label = 0
Feature = 20 and Label = 0
Feature = 29 and Label = 1
Feature = 49 and Label = 1
Feature = 52 and Label = 0
Feature = 81 and Label = 1
Feature = 80 and Label = 0
Feature = 30 and Label = 0
Feature = 86 and Label = 0
Feature = 47 and Label = 1
Feature = 31 and Label = 1
Feature = 31 and Label = 1
Feature = 28 and Label = 0
Feature = 74 and Label = 0
Feature = 81 and Label = 1
Feature = 32 and Label = 0
Feature = 83 and Label = 1
Feature = 94 and Label = 0
Feature = 27 and Label = 1
Feature = 6 and Label = 0
Feature = 77 and Label = 1
Feature = 30 and Label = 0
Feature = 80 and Label = 0
Feature = 71 and Label = 1
Feature = 95 and Label = 1
Feature = 59 and Label = 1
Feature = 25 and Label = 1
Feature = 87 and Label = 1
Feature = 84 and Label = 0
Feature = 46 and Label = 0
Feature = 82 and Label = 0
Feature = 36 and Label = 0
Feature = 14 and Label = 0
Feature = 85 and Label = 1
Feature = 50 and Label = 0
Feature = 88 and Label = 0
Feature = 3 and Label = 1
Feature = 43 and Label = 1
Feature = 8 and Label = 0
Feature = 97 and Label = 1
Feature = 60 and Label = 0
Feature = 55 and Label = 1
Feature = 7 and Label = 1
Feature = 38 and Label = 0
Feature = 69 and Label = 1
Feature = 92 and Label = 0
Feature = 75 and Label = 1
Feature = 76 and Label = 0
Feature = 57 and Label = 1
Feature = 1 and Label = 1
Feature = 54 and Label = 0
Feature = 2 and Label = 0
Feature = 12 and Label = 0
Feature = 58 and Label = 0
Feature = 64 and Label = 0

Feature = 70 and Label = 0
Feature = 61 and Label = 1
Feature = 9 and Label = 1
Feature = 53 and Label = 1
Feature = 73 and Label = 1
Feature = 13 and Label = 1
Feature = 98 and Label = 0
Feature = 33 and Label = 1
Feature = 86 and Label = 0
Feature = 56 and Label = 0
Feature = 91 and Label = 1
Feature = 40 and Label = 0
Feature = 72 and Label = 0
Feature = 29 and Label = 1
Feature = 39 and Label = 1
Feature = 5 and Label = 1
Feature = 11 and Label = 1
Feature = 41 and Label = 1
Feature = 0 and Label = 0
Feature = 20 and Label = 0
Feature = 47 and Label = 1
Feature = 52 and Label = 0
Feature = 66 and Label = 0
Feature = 4 and Label = 0
Feature = 16 and Label = 0
Feature = 65 and Label = 1
Feature = 78 and Label = 0
Feature = 18 and Label = 0
Feature = 93 and Label = 1
Feature = 22 and Label = 0
Feature = 68 and Label = 0
Feature = 21 and Label = 1
Feature = 17 and Label = 1
Feature = 45 and Label = 1
Feature = 37 and Label = 1
Feature = 48 and Label = 0
Feature = 24 and Label = 0
Feature = 34 and Label = 0
Feature = 89 and Label = 1
Feature = 49 and Label = 1
Feature = 23 and Label = 1
Feature = 51 and Label = 1
Feature = 67 and Label = 1
Feature = 15 and Label = 1
Feature = 44 and Label = 0
Feature = 99 and Label = 1
Feature = 90 and Label = 0
Feature = 19 and Label = 1
Feature = 10 and Label = 0
Feature = 63 and Label = 1
Feature = 62 and Label = 0
Feature = 26 and Label = 0
Feature = 42 and Label = 0
Feature = 79 and Label = 1
Feature = 96 and Label = 0
Feature = 35 and Label = 1
Feature = 89 and Label = 1
Feature = 85 and Label = 1
Feature = 0 and Label = 0
Feature = 37 and Label = 1
Feature = 38 and Label = 0

Feature = 21 and Label = 1
Feature = 52 and Label = 0
Feature = 23 and Label = 1
Feature = 62 and Label = 0
Feature = 80 and Label = 0
Feature = 77 and Label = 1
Feature = 92 and Label = 0
Feature = 43 and Label = 1
Feature = 55 and Label = 1
Feature = 74 and Label = 0
Feature = 22 and Label = 0
Feature = 18 and Label = 0
Feature = 75 and Label = 1
Feature = 69 and Label = 1
Feature = 36 and Label = 0
Feature = 61 and Label = 1
Feature = 78 and Label = 0
Feature = 57 and Label = 1
Feature = 16 and Label = 0
Feature = 44 and Label = 0
Feature = 53 and Label = 1
Feature = 63 and Label = 1
Feature = 26 and Label = 0
Feature = 15 and Label = 1
Feature = 39 and Label = 1
Feature = 47 and Label = 1
Feature = 31 and Label = 1
Feature = 66 and Label = 0
Feature = 79 and Label = 1
Feature = 98 and Label = 0
Feature = 76 and Label = 0
Feature = 9 and Label = 1
Feature = 65 and Label = 1
Feature = 14 and Label = 0
Feature = 99 and Label = 1
Feature = 91 and Label = 1
Feature = 6 and Label = 0
Feature = 41 and Label = 1
Feature = 32 and Label = 0
Feature = 94 and Label = 0
Feature = 13 and Label = 1
Feature = 17 and Label = 1
Feature = 20 and Label = 0
Feature = 24 and Label = 0
Feature = 67 and Label = 1
Feature = 10 and Label = 0
Feature = 12 and Label = 0
Feature = 5 and Label = 1
Feature = 19 and Label = 1
Feature = 64 and Label = 0
Feature = 46 and Label = 0
Feature = 27 and Label = 1
Feature = 4 and Label = 0
Feature = 81 and Label = 1
Feature = 34 and Label = 0
Feature = 59 and Label = 1
Feature = 68 and Label = 0
Feature = 70 and Label = 0
Feature = 51 and Label = 1
Feature = 72 and Label = 0
Feature = 58 and Label = 0

Feature = 54 and Label = 0
Feature = 48 and Label = 0
Feature = 96 and Label = 0
Feature = 56 and Label = 0
Feature = 45 and Label = 1
Feature = 3 and Label = 1
Feature = 25 and Label = 1
Feature = 73 and Label = 1
Feature = 83 and Label = 1
Feature = 42 and Label = 0
Feature = 97 and Label = 1
Feature = 40 and Label = 0
Feature = 8 and Label = 0
Feature = 50 and Label = 0
Feature = 2 and Label = 0
Feature = 49 and Label = 1
Feature = 93 and Label = 1
Feature = 7 and Label = 1
Feature = 90 and Label = 0
Feature = 28 and Label = 0
Feature = 87 and Label = 1
Feature = 84 and Label = 0
Feature = 35 and Label = 1
Feature = 88 and Label = 0
Feature = 11 and Label = 1
Feature = 86 and Label = 0
Feature = 82 and Label = 0
Feature = 1 and Label = 1
Feature = 60 and Label = 0
Feature = 95 and Label = 1
Feature = 33 and Label = 1
Feature = 30 and Label = 0
Feature = 29 and Label = 1
Feature = 71 and Label = 1
Feature = 68 and Label = 0
Feature = 21 and Label = 1
Feature = 70 and Label = 0
Feature = 63 and Label = 1
Feature = 19 and Label = 1
Feature = 38 and Label = 0
Feature = 37 and Label = 1
Feature = 42 and Label = 0
Feature = 95 and Label = 1
Feature = 87 and Label = 1
Feature = 18 and Label = 0
Feature = 8 and Label = 0
Feature = 35 and Label = 1
Feature = 28 and Label = 0
Feature = 94 and Label = 0
Feature = 5 and Label = 1
Feature = 26 and Label = 0
Feature = 78 and Label = 0
Feature = 54 and Label = 0
Feature = 3 and Label = 1
Feature = 53 and Label = 1
Feature = 83 and Label = 1
Feature = 79 and Label = 1
Feature = 64 and Label = 0
Feature = 62 and Label = 0
Feature = 89 and Label = 1
Feature = 55 and Label = 1

Feature = 11 and Label = 1
Feature = 50 and Label = 0
Feature = 84 and Label = 0
Feature = 47 and Label = 1
Feature = 52 and Label = 0
Feature = 91 and Label = 1
Feature = 60 and Label = 0
Feature = 51 and Label = 1
Feature = 33 and Label = 1
Feature = 82 and Label = 0
Feature = 45 and Label = 1
Feature = 27 and Label = 1
Feature = 96 and Label = 0
Feature = 24 and Label = 0
Feature = 61 and Label = 1
Feature = 4 and Label = 0
Feature = 98 and Label = 0
Feature = 22 and Label = 0
Feature = 15 and Label = 1
Feature = 73 and Label = 1
Feature = 71 and Label = 1
Feature = 77 and Label = 1
Feature = 92 and Label = 0
Feature = 32 and Label = 0
Feature = 58 and Label = 0
Feature = 80 and Label = 0
Feature = 85 and Label = 1
Feature = 57 and Label = 1
Feature = 46 and Label = 0
Feature = 99 and Label = 1
Feature = 16 and Label = 0
Feature = 39 and Label = 1
Feature = 43 and Label = 1
Feature = 97 and Label = 1
Feature = 56 and Label = 0
Feature = 29 and Label = 1
Feature = 25 and Label = 1
Feature = 0 and Label = 0
Feature = 12 and Label = 0
Feature = 67 and Label = 1
Feature = 34 and Label = 0
Feature = 44 and Label = 0
Feature = 6 and Label = 0
Feature = 88 and Label = 0
Feature = 41 and Label = 1
Feature = 65 and Label = 1
Feature = 14 and Label = 0
Feature = 40 and Label = 0
Feature = 93 and Label = 1
Feature = 13 and Label = 1
Feature = 49 and Label = 1
Feature = 23 and Label = 1
Feature = 74 and Label = 0
Feature = 90 and Label = 0
Feature = 59 and Label = 1
Feature = 7 and Label = 1
Feature = 76 and Label = 0
Feature = 72 and Label = 0
Feature = 1 and Label = 1
Feature = 48 and Label = 0
Feature = 9 and Label = 1

Feature = 69 and Label = 1
Feature = 2 and Label = 0
Feature = 10 and Label = 0
Feature = 81 and Label = 1
Feature = 36 and Label = 0
Feature = 30 and Label = 0
Feature = 75 and Label = 1
Feature = 17 and Label = 1
Feature = 31 and Label = 1
Feature = 20 and Label = 0
Feature = 66 and Label = 0
Feature = 86 and Label = 0
Feature = 92 and Label = 0
Feature = 65 and Label = 1
Feature = 15 and Label = 1
Feature = 87 and Label = 1
Feature = 74 and Label = 0
Feature = 66 and Label = 0
Feature = 4 and Label = 0
Feature = 68 and Label = 0
Feature = 5 and Label = 1
Feature = 47 and Label = 1
Feature = 6 and Label = 0
Feature = 13 and Label = 1
Feature = 83 and Label = 1
Feature = 71 and Label = 1
Feature = 80 and Label = 0
Feature = 58 and Label = 0
Feature = 10 and Label = 0
Feature = 77 and Label = 1
Feature = 19 and Label = 1
Feature = 54 and Label = 0
Feature = 90 and Label = 0
Feature = 98 and Label = 0
Feature = 38 and Label = 0
Feature = 64 and Label = 0
Feature = 60 and Label = 0
Feature = 81 and Label = 1
Feature = 35 and Label = 1
Feature = 88 and Label = 0
Feature = 39 and Label = 1
Feature = 25 and Label = 1
Feature = 75 and Label = 1
Feature = 36 and Label = 0
Feature = 82 and Label = 0
Feature = 2 and Label = 0
Feature = 93 and Label = 1
Feature = 24 and Label = 0
Feature = 43 and Label = 1
Feature = 78 and Label = 0
Feature = 16 and Label = 0
Feature = 57 and Label = 1
Feature = 31 and Label = 1
Feature = 51 and Label = 1
Feature = 49 and Label = 1
Feature = 45 and Label = 1
Feature = 95 and Label = 1
Feature = 72 and Label = 0
Feature = 46 and Label = 0
Feature = 23 and Label = 1
Feature = 11 and Label = 1

Feature = 29 and Label = 1
Feature = 67 and Label = 1
Feature = 42 and Label = 0
Feature = 1 and Label = 1
Feature = 73 and Label = 1
Feature = 76 and Label = 0
Feature = 56 and Label = 0
Feature = 28 and Label = 0
Feature = 50 and Label = 0
Feature = 91 and Label = 1
Feature = 48 and Label = 0
Feature = 89 and Label = 1
Feature = 14 and Label = 0
Feature = 53 and Label = 1
Feature = 27 and Label = 1
Feature = 20 and Label = 0
Feature = 12 and Label = 0
Feature = 37 and Label = 1
Feature = 44 and Label = 0
Feature = 26 and Label = 0
Feature = 85 and Label = 1
Feature = 96 and Label = 0
Feature = 3 and Label = 1
Feature = 40 and Label = 0
Feature = 7 and Label = 1
Feature = 30 and Label = 0
Feature = 0 and Label = 0
Feature = 59 and Label = 1
Feature = 33 and Label = 1
Feature = 69 and Label = 1
Feature = 41 and Label = 1
Feature = 63 and Label = 1
Feature = 9 and Label = 1
Feature = 32 and Label = 0
Feature = 94 and Label = 0
Feature = 22 and Label = 0
Feature = 62 and Label = 0
Feature = 99 and Label = 1
Feature = 18 and Label = 0
Feature = 97 and Label = 1
Feature = 55 and Label = 1
Feature = 52 and Label = 0
Feature = 21 and Label = 1
Feature = 84 and Label = 0
Feature = 34 and Label = 0
Feature = 86 and Label = 0
Feature = 8 and Label = 0
Feature = 61 and Label = 1
Feature = 79 and Label = 1
Feature = 17 and Label = 1
Feature = 70 and Label = 0
Feature = 93 and Label = 1
Feature = 78 and Label = 0
Feature = 47 and Label = 1
Feature = 17 and Label = 1
Feature = 6 and Label = 0
Feature = 36 and Label = 0
Feature = 83 and Label = 1
Feature = 81 and Label = 1
Feature = 76 and Label = 0
Feature = 79 and Label = 1

Feature = 3 and Label = 1
Feature = 51 and Label = 1
Feature = 62 and Label = 0
Feature = 40 and Label = 0
Feature = 27 and Label = 1
Feature = 87 and Label = 1
Feature = 96 and Label = 0
Feature = 77 and Label = 1
Feature = 70 and Label = 0
Feature = 68 and Label = 0
Feature = 59 and Label = 1
Feature = 1 and Label = 1
Feature = 31 and Label = 1
Feature = 21 and Label = 1
Feature = 15 and Label = 1
Feature = 71 and Label = 1
Feature = 19 and Label = 1
Feature = 10 and Label = 0
Feature = 66 and Label = 0
Feature = 75 and Label = 1
Feature = 86 and Label = 0
Feature = 2 and Label = 0
Feature = 80 and Label = 0
Feature = 30 and Label = 0
Feature = 11 and Label = 1
Feature = 14 and Label = 0
Feature = 73 and Label = 1
Feature = 54 and Label = 0
Feature = 18 and Label = 0
Feature = 42 and Label = 0
Feature = 92 and Label = 0
Feature = 29 and Label = 1
Feature = 28 and Label = 0
Feature = 74 and Label = 0
Feature = 61 and Label = 1
Feature = 7 and Label = 1
Feature = 85 and Label = 1
Feature = 69 and Label = 1
Feature = 12 and Label = 0
Feature = 99 and Label = 1
Feature = 43 and Label = 1
Feature = 20 and Label = 0
Feature = 26 and Label = 0
Feature = 60 and Label = 0
Feature = 41 and Label = 1
Feature = 72 and Label = 0
Feature = 53 and Label = 1
Feature = 52 and Label = 0
Feature = 82 and Label = 0
Feature = 46 and Label = 0
Feature = 25 and Label = 1
Feature = 97 and Label = 1
Feature = 56 and Label = 0
Feature = 90 and Label = 0
Feature = 49 and Label = 1
Feature = 44 and Label = 0
Feature = 35 and Label = 1
Feature = 65 and Label = 1
Feature = 22 and Label = 0
Feature = 32 and Label = 0
Feature = 64 and Label = 0

Feature = 33 and Label = 1
Feature = 16 and Label = 0
Feature = 48 and Label = 0
Feature = 57 and Label = 1
Feature = 34 and Label = 0
Feature = 91 and Label = 1
Feature = 9 and Label = 1
Feature = 95 and Label = 1
Feature = 84 and Label = 0
Feature = 55 and Label = 1
Feature = 89 and Label = 1
Feature = 0 and Label = 0
Feature = 67 and Label = 1
Feature = 88 and Label = 0
Feature = 45 and Label = 1
Feature = 38 and Label = 0
Feature = 23 and Label = 1
Feature = 13 and Label = 1
Feature = 39 and Label = 1
Feature = 63 and Label = 1
Feature = 58 and Label = 0
Feature = 50 and Label = 0
Feature = 98 and Label = 0
Feature = 8 and Label = 0
Feature = 4 and Label = 0
Feature = 94 and Label = 0
Feature = 5 and Label = 1
Feature = 24 and Label = 0
Feature = 37 and Label = 1
Feature = 72 and Label = 0
Feature = 81 and Label = 1
Feature = 36 and Label = 0
Feature = 94 and Label = 0
Feature = 28 and Label = 0
Feature = 52 and Label = 0
Feature = 27 and Label = 1
Feature = 29 and Label = 1
Feature = 32 and Label = 0
Feature = 30 and Label = 0
Feature = 56 and Label = 0
Feature = 48 and Label = 0
Feature = 79 and Label = 1
Feature = 93 and Label = 1
Feature = 33 and Label = 1
Feature = 37 and Label = 1
Feature = 24 and Label = 0
Feature = 92 and Label = 0
Feature = 25 and Label = 1
Feature = 73 and Label = 1
Feature = 38 and Label = 0
Feature = 61 and Label = 1
Feature = 98 and Label = 0
Feature = 89 and Label = 1
Feature = 50 and Label = 0
Feature = 40 and Label = 0
Feature = 13 and Label = 1
Feature = 35 and Label = 1
Feature = 85 and Label = 1
Feature = 4 and Label = 0
Feature = 67 and Label = 1
Feature = 66 and Label = 0

Feature = 77 and Label = 1
Feature = 62 and Label = 0
Feature = 51 and Label = 1
Feature = 55 and Label = 1
Feature = 39 and Label = 1
Feature = 18 and Label = 0
Feature = 10 and Label = 0
Feature = 91 and Label = 1
Feature = 42 and Label = 0
Feature = 44 and Label = 0
Feature = 86 and Label = 0
Feature = 46 and Label = 0
Feature = 54 and Label = 0
Feature = 1 and Label = 1
Feature = 20 and Label = 0
Feature = 22 and Label = 0
Feature = 9 and Label = 1
Feature = 3 and Label = 1
Feature = 17 and Label = 1
Feature = 5 and Label = 1
Feature = 65 and Label = 1
Feature = 2 and Label = 0
Feature = 88 and Label = 0
Feature = 19 and Label = 1
Feature = 60 and Label = 0
Feature = 43 and Label = 1
Feature = 15 and Label = 1
Feature = 8 and Label = 0
Feature = 47 and Label = 1
Feature = 12 and Label = 0
Feature = 95 and Label = 1
Feature = 70 and Label = 0
Feature = 7 and Label = 1
Feature = 49 and Label = 1
Feature = 57 and Label = 1
Feature = 16 and Label = 0
Feature = 87 and Label = 1
Feature = 41 and Label = 1
Feature = 58 and Label = 0
Feature = 78 and Label = 0
Feature = 96 and Label = 0
Feature = 53 and Label = 1
Feature = 82 and Label = 0
Feature = 0 and Label = 0
Feature = 76 and Label = 0
Feature = 14 and Label = 0
Feature = 23 and Label = 1
Feature = 59 and Label = 1
Feature = 63 and Label = 1
Feature = 83 and Label = 1
Feature = 11 and Label = 1
Feature = 74 and Label = 0
Feature = 69 and Label = 1
Feature = 90 and Label = 0
Feature = 31 and Label = 1
Feature = 75 and Label = 1
Feature = 21 and Label = 1
Feature = 84 and Label = 0
Feature = 71 and Label = 1
Feature = 64 and Label = 0
Feature = 45 and Label = 1

Feature = 6 and Label = 0
Feature = 26 and Label = 0
Feature = 99 and Label = 1
Feature = 68 and Label = 0
Feature = 97 and Label = 1
Feature = 34 and Label = 0
Feature = 80 and Label = 0
Feature = 20 and Label = 0
Feature = 99 and Label = 1
Feature = 47 and Label = 1
Feature = 7 and Label = 1
Feature = 8 and Label = 0
Feature = 56 and Label = 0
Feature = 82 and Label = 0
Feature = 41 and Label = 1
Feature = 55 and Label = 1
Feature = 83 and Label = 1
Feature = 66 and Label = 0
Feature = 24 and Label = 0
Feature = 79 and Label = 1
Feature = 33 and Label = 1
Feature = 75 and Label = 1
Feature = 64 and Label = 0
Feature = 27 and Label = 1
Feature = 54 and Label = 0
Feature = 30 and Label = 0
Feature = 58 and Label = 0
Feature = 84 and Label = 0
Feature = 52 and Label = 0
Feature = 63 and Label = 1
Feature = 0 and Label = 0
Feature = 5 and Label = 1
Feature = 73 and Label = 1
Feature = 90 and Label = 0
Feature = 93 and Label = 1
Feature = 39 and Label = 1
Feature = 18 and Label = 0
Feature = 48 and Label = 0
Feature = 81 and Label = 1
Feature = 98 and Label = 0
Feature = 76 and Label = 0
Feature = 77 and Label = 1
Feature = 1 and Label = 1
Feature = 19 and Label = 1
Feature = 53 and Label = 1
Feature = 12 and Label = 0
Feature = 43 and Label = 1
Feature = 62 and Label = 0
Feature = 16 and Label = 0
Feature = 32 and Label = 0
Feature = 26 and Label = 0
Feature = 3 and Label = 1
Feature = 96 and Label = 0
Feature = 44 and Label = 0
Feature = 13 and Label = 1
Feature = 74 and Label = 0
Feature = 70 and Label = 0
Feature = 28 and Label = 0
Feature = 37 and Label = 1
Feature = 22 and Label = 0
Feature = 88 and Label = 0

Feature = 94 and Label = 0
Feature = 14 and Label = 0
Feature = 6 and Label = 0
Feature = 86 and Label = 0
Feature = 17 and Label = 1
Feature = 2 and Label = 0
Feature = 45 and Label = 1
Feature = 72 and Label = 0
Feature = 80 and Label = 0
Feature = 71 and Label = 1
Feature = 89 and Label = 1
Feature = 95 and Label = 1
Feature = 46 and Label = 0
Feature = 31 and Label = 1
Feature = 68 and Label = 0
Feature = 65 and Label = 1
Feature = 34 and Label = 0
Feature = 59 and Label = 1
Feature = 29 and Label = 1
Feature = 78 and Label = 0
Feature = 51 and Label = 1
Feature = 42 and Label = 0
Feature = 97 and Label = 1
Feature = 91 and Label = 1
Feature = 69 and Label = 1
Feature = 35 and Label = 1
Feature = 85 and Label = 1
Feature = 21 and Label = 1
Feature = 57 and Label = 1
Feature = 36 and Label = 0
Feature = 49 and Label = 1
Feature = 15 and Label = 1
Feature = 61 and Label = 1
Feature = 4 and Label = 0
Feature = 10 and Label = 0
Feature = 9 and Label = 1
Feature = 67 and Label = 1
Feature = 11 and Label = 1
Feature = 50 and Label = 0
Feature = 40 and Label = 0
Feature = 23 and Label = 1
Feature = 92 and Label = 0
Feature = 60 and Label = 0
Feature = 38 and Label = 0
Feature = 25 and Label = 1
Feature = 87 and Label = 1
Feature = 56 and Label = 0
Feature = 46 and Label = 0
Feature = 24 and Label = 0
Feature = 48 and Label = 0
Feature = 86 and Label = 0
Feature = 2 and Label = 0
Feature = 31 and Label = 1
Feature = 71 and Label = 1
Feature = 43 and Label = 1
Feature = 19 and Label = 1
Feature = 84 and Label = 0
Feature = 95 and Label = 1
Feature = 81 and Label = 1
Feature = 76 and Label = 0
Feature = 29 and Label = 1

Feature = 91 and Label = 1
Feature = 25 and Label = 1
Feature = 69 and Label = 1
Feature = 21 and Label = 1
Feature = 87 and Label = 1
Feature = 97 and Label = 1
Feature = 7 and Label = 1
Feature = 57 and Label = 1
Feature = 33 and Label = 1
Feature = 61 and Label = 1
Feature = 82 and Label = 0
Feature = 28 and Label = 0
Feature = 41 and Label = 1
Feature = 44 and Label = 0
Feature = 12 and Label = 0
Feature = 51 and Label = 1
Feature = 11 and Label = 1
Feature = 32 and Label = 0
Feature = 52 and Label = 0
Feature = 68 and Label = 0
Feature = 40 and Label = 0
Feature = 78 and Label = 0
Feature = 45 and Label = 1
Feature = 18 and Label = 0
Feature = 98 and Label = 0
Feature = 20 and Label = 0
Feature = 37 and Label = 1
Feature = 85 and Label = 1
Feature = 65 and Label = 1
Feature = 79 and Label = 1
Feature = 5 and Label = 1
Feature = 8 and Label = 0
Feature = 55 and Label = 1
Feature = 72 and Label = 0
Feature = 96 and Label = 0
Feature = 0 and Label = 0
Feature = 27 and Label = 1
Feature = 80 and Label = 0
Feature = 62 and Label = 0
Feature = 89 and Label = 1
Feature = 83 and Label = 1
Feature = 54 and Label = 0
Feature = 77 and Label = 1
Feature = 67 and Label = 1
Feature = 10 and Label = 0
Feature = 63 and Label = 1
Feature = 60 and Label = 0
Feature = 88 and Label = 0
Feature = 35 and Label = 1
Feature = 49 and Label = 1
Feature = 6 and Label = 0
Feature = 38 and Label = 0
Feature = 30 and Label = 0
Feature = 58 and Label = 0
Feature = 1 and Label = 1
Feature = 22 and Label = 0
Feature = 70 and Label = 0
Feature = 15 and Label = 1
Feature = 4 and Label = 0
Feature = 75 and Label = 1
Feature = 23 and Label = 1

Feature = 3 and Label = 1
Feature = 50 and Label = 0
Feature = 39 and Label = 1
Feature = 9 and Label = 1
Feature = 94 and Label = 0
Feature = 90 and Label = 0
Feature = 59 and Label = 1
Feature = 36 and Label = 0
Feature = 92 and Label = 0
Feature = 66 and Label = 0
Feature = 64 and Label = 0
Feature = 93 and Label = 1
Feature = 34 and Label = 0
Feature = 17 and Label = 1
Feature = 73 and Label = 1
Feature = 16 and Label = 0
Feature = 26 and Label = 0
Feature = 47 and Label = 1
Feature = 99 and Label = 1
Feature = 14 and Label = 0
Feature = 74 and Label = 0
Feature = 13 and Label = 1
Feature = 53 and Label = 1
Feature = 42 and Label = 0

In [ ]:

```
# Shuffle, Repeat, Batch and Pre-fetch

data = tf.data.Dataset.from_tensor_slices((features, labels))
data = data.shuffle(len(data))
data = data.repeat(10) # repeat 10 times
data = data.batch(10)
data = data.prefetch(1)
for feat, lab in data:
  print(f'Feature = {feat.numpy()} and Label = {lab.numpy()}')
```

```
Feature = [38 43 74 63  7 82 13 19 81 95] and Label = [0 1 0 1 1 0 1 1 1
1]
Feature = [64 44 17 86 56 51 34 37 12 75] and Label = [0 0 1 0 0 1 0 1 0
1]
Feature = [36  4 66 10 48 97  5 54 96 52] and Label = [0 0 0 0 0 1 1 0 0
0]
Feature = [ 0 85 47 21 53 87 24 94 41 79] and Label = [0 1 1 1 1 1 0 0 1
1]
Feature = [98 61  1 70 26 80 69  2  9 76] and Label = [0 1 1 0 0 0 1 0 1
0]
Feature = [57 60 62 35 83 40 32 72 68 20] and Label = [1 0 0 1 1 0 0 0 0
0]
Feature = [14 55 25 92 71 88 91 39 99 58] and Label = [0 1 1 0 1 0 1 1 1
0]
Feature = [78 27 33  8 73 77 15 90 31 18] and Label = [0 1 1 0 1 1 1 0 1
0]
Feature = [29 67 50 45 46 59 65 28 84 49] and Label = [1 1 0 1 0 1 1 0 0
1]
Feature = [22 30 23 11  3 16 42 89 93  6] and Label = [0 0 1 1 1 0 0 1 1
0]
Feature = [43 99 68 58 17 82 67 79 46 45] and Label = [1 1 0 0 1 0 1 1 0
1]
Feature = [51 75 81 59 16 40 48 88 55 29] and Label = [1 1 1 1 0 0 0 0 1
1]
Feature = [37 21 27  2 64 32 60 72 97 28] and Label = [1 1 1 0 0 0 0 0 1
0]
Feature = [30  8 61 56 76 47 52 74 78 26] and Label = [0 0 1 0 0 1 0 0 0
0]
Feature = [57 24 63  1 38  6 90  9 39 91] and Label = [1 0 1 1 0 0 0 1 1
1]
Feature = [20 54 14 95 92 80 25 87 22  7] and Label = [0 0 0 1 0 0 1 1 0
1]
Feature = [69  5 71  4 18 12 83  0 42 23] and Label = [1 1 1 0 0 0 1 0 0
1]
Feature = [31 11 77 66 35 33 36 65 98 13] and Label = [1 1 1 0 1 1 0 1 0
1]
Feature = [ 3 93 50 86 70 96 15 85 84 19] and Label = [1 1 0 0 0 0 1 1 0
1]
Feature = [73 34 53 49 94 62 41 10 89 44] and Label = [1 0 1 1 0 0 1 0 1
0]
Feature = [40 47 27 99 42 14 50 79 60 92] and Label = [0 1 1 1 0 0 0 1 0
0]
Feature = [70 49 62 11 74 24 87  3 16 63] and Label = [0 1 0 1 0 0 1 1 0
1]
Feature = [44  2  1 46  8 77 26 59 93 19] and Label = [0 0 1 0 0 1 0 1 1
1]
Feature = [ 6 13 95 15 52 56 80 25 90 28] and Label = [0 1 1 1 0 0 0 1 0
0]
Feature = [67 37 29 54  4 97 35 86 43 48] and Label = [1 1 1 0 0 1 1 0 1
0]
Feature = [84 66  0 83 45 64 75 89 78 65] and Label = [0 0 0 1 1 0 1 1 0
1]
Feature = [17 38 94 53 88 34 85 10 61 33] and Label = [1 0 0 1 0 0 1 0 1
1]
Feature = [41 39 30 73  5 81 69 68 21  7] and Label = [1 1 0 1 1 1 1 0 1
1]
Feature = [96 22 31 55 12 57  9 36 58 20] and Label = [0 0 1 1 0 1 1 0 0
0]
Feature = [91 32 72 82 71 18 98 23 76 51] and Label = [1 0 0 0 1 0 0 1 0
1]
Feature = [66 48 73 14 36  7 88 97  4 64] and Label = [0 0 1 0 0 1 0 1 0
```

0]
Feature = [5 47 56 37 49 35 85 34 98 40] and Label = [1 1 0 1 1 1 1 0 0
0]
Feature = [18 32 19 68 1 55 87 17 51 95] and Label = [0 0 1 0 1 1 1 1 1
1]
Feature = [31 78 33 60 46 79 41 77 96 91] and Label = [1 0 1 0 0 1 1 1 0
1]
Feature = [24 94 53 54 25 67 9 28 72 43] and Label = [0 0 1 0 1 1 1 0 0
1]
Feature = [6 3 81 52 20 59 13 29 58 63] and Label = [0 1 1 0 0 1 1 1 0
1]
Feature = [21 50 89 84 80 12 22 10 90 74] and Label = [1 0 1 0 0 0 0 0 0
0]
Feature = [61 71 30 82 83 92 39 45 11 26] and Label = [1 1 0 0 1 0 1 1 1
0]
Feature = [15 16 86 2 62 0 75 93 65 27] and Label = [1 0 0 0 0 0 1 1 1
1]
Feature = [76 42 69 23 57 8 70 99 44 38] and Label = [0 0 1 1 1 0 0 1 0
0]
Feature = [49 36 96 15 56 99 33 69 23 46] and Label = [1 0 0 1 0 1 1 1 1
0]
Feature = [83 59 38 20 39 58 10 87 90 54] and Label = [1 1 0 0 1 0 0 1 0
0]
Feature = [14 28 19 73 43 63 61 34 48 44] and Label = [0 0 1 1 1 1 1 0 0
0]
Feature = [66 2 51 79 85 78 16 11 37 80] and Label = [0 0 1 1 1 0 0 1 1
0]
Feature = [47 31 98 29 82 70 97 67 50 1] and Label = [1 1 0 1 0 0 1 1 0
1]
Feature = [3 84 71 7 52 32 0 5 89 62] and Label = [1 0 1 1 0 0 0 1 1
0]
Feature = [26 88 86 57 24 45 68 4 22 64] and Label = [0 0 0 1 0 1 0 0 0
0]
Feature = [9 74 8 75 60 35 42 76 21 91] and Label = [1 0 0 1 0 1 0 0 1
1]
Feature = [94 18 6 41 55 72 81 92 93 77] and Label = [0 0 0 1 1 0 1 0 1
1]
Feature = [13 30 40 53 27 65 25 95 17 12] and Label = [1 0 0 1 1 1 1 1 1
0]
Feature = [54 42 78 7 98 44 8 63 36 21] and Label = [0 0 0 1 0 0 0 1 0
1]
Feature = [23 60 39 40 50 68 24 20 85 18] and Label = [1 0 1 0 0 0 0 0 1
0]
Feature = [89 53 19 6 56 99 58 31 1 96] and Label = [1 1 1 0 0 1 0 1 1
0]
Feature = [66 67 5 13 83 81 59 69 27 91] and Label = [0 1 1 1 1 1 1 1 1
1]
Feature = [47 95 62 88 46 77 11 2 94 33] and Label = [1 1 0 0 0 1 1 0 0
1]
Feature = [29 65 80 57 16 92 75 12 71 9] and Label = [1 1 0 1 0 0 1 0 1
1]
Feature = [34 35 3 90 48 52 87 22 79 55] and Label = [0 1 1 0 0 0 1 0 1
1]
Feature = [15 73 72 0 28 74 30 49 43 32] and Label = [1 1 0 0 0 0 0 1 1
0]
Feature = [4 64 97 17 26 10 93 82 25 45] and Label = [0 0 1 1 0 0 1 0 1
1]
Feature = [84 51 86 70 76 14 61 41 38 37] and Label = [0 1 0 0 0 0 1 1 0
1]
Feature = [95 30 70 1 62 67 98 66 87 0] and Label = [1 0 0 1 0 1 0 0 1
0]

```
Feature = [24 44  2 46 59 65 56 28 51 89] and Label = [0 0 0 0 1 1 0 0 1
1]
Feature = [12 74 33 94 21 26 90 29 32 61] and Label = [0 0 1 0 1 0 0 1 0
1]
Feature = [54 96 73 81 45 85 19 37 18  8] and Label = [0 0 1 1 1 1 1 1 0
0]
Feature = [42 80 36 15 14  4 91 63  5 34] and Label = [0 0 0 1 0 0 1 1 1
0]
Feature = [57 48 25 41 77 79 60 88  7 69] and Label = [1 0 1 1 1 1 0 0 1
1]
Feature = [76 55 52 11 71 49 43 40 47  9] and Label = [0 1 0 1 1 1 1 0 1
1]
Feature = [17 84 10 31 86 27 22 20 53 99] and Label = [1 0 0 1 0 1 0 0 1
1]
Feature = [ 6 50 35 92 75 97 38 78 16  3] and Label = [0 0 1 0 1 1 0 0 0
1]
Feature = [72 23 64 93 39 83 58 82 68 13] and Label = [0 1 0 1 1 1 0 0 0
1]
Feature = [89 46 67 74 66 34  8 33  2 96] and Label = [1 0 1 0 0 0 0 1 0
0]
Feature = [69 19  6 38 63 17 37 82 93 51] and Label = [1 1 0 0 1 1 1 0 1
1]
Feature = [49 60 84 31 54 98  1  0 91  5] and Label = [1 0 0 1 0 0 1 0 1
1]
Feature = [14 12 72 52 79 97 15 95 30 88] and Label = [0 0 0 0 1 1 1 1 0
0]
Feature = [65 80 48  3 47 71 40 21 41 39] and Label = [1 0 0 1 1 1 0 1 1
1]
Feature = [13 45 68 56 24 18 73 86 85 75] and Label = [1 1 0 0 0 0 1 0 1
1]
Feature = [58 90 78 11 28 87 29 64 94 20] and Label = [0 0 0 1 0 1 1 0 0
0]
Feature = [43 42 81 32 57  7 83 50  9 26] and Label = [1 0 1 0 1 1 1 0 1
0]
Feature = [59 44 23 35 61 55 70 92 36 27] and Label = [1 0 1 1 1 1 0 0 0
1]
Feature = [ 4 16 25 10 53 22 76 62 77 99] and Label = [0 0 1 0 1 0 0 0 1
1]
Feature = [ 5 92 36 79 25 45 77  4 96 15] and Label = [1 0 0 1 1 1 1 0 0
1]
Feature = [86  1 88 31 12 14 95 72  6 64] and Label = [0 1 0 1 0 0 1 0 0
0]
Feature = [97 51 34  3 84 40 10 29 90 37] and Label = [1 1 0 1 0 0 0 1 0
1]
Feature = [ 7 17 41 30 13 26  9 69 94 21] and Label = [1 1 1 0 1 0 1 1 0
1]
Feature = [81 74 57 43 28 73 66 18 82 20] and Label = [1 0 1 1 0 1 0 0 0
0]
Feature = [ 2 93 98 56 50 80 52 47 99 63] and Label = [0 1 0 0 0 0 0 1 1
1]
Feature = [48 75 61 59 70 68 16 33 85 91] and Label = [0 1 1 1 0 0 0 1 1
1]
Feature = [27 38 87 23  8 83  0 42 39 49] and Label = [1 0 1 1 0 1 0 0 1
1]
Feature = [65 53 32 55 62 54 44 67 89 35] and Label = [1 1 0 1 0 0 0 1 1
1]
Feature = [78 11 22 58 60 46 19 76 24 71] and Label = [0 1 0 0 0 0 1 0 0
1]
Feature = [ 4 49  6 39 40 86 31  2 93 83] and Label = [0 1 0 1 0 0 1 0 1
1]
Feature = [70 52 58 47 95  3 38 77 54 48] and Label = [0 0 0 1 1 1 0 1 0
```

```
0]
Feature = [84 96 78 94 74 63 32 72 28 67] and Label = [0 0 0 0 0 1 0 0 0
1]
Feature = [87 34 51 20 82 99 16 44 30 98] and Label = [1 0 1 0 0 1 0 0 0
0]
Feature = [19 89 25 22 33 80 27 43 21  9] and Label = [1 1 1 0 1 0 1 1 1
1]
Feature = [53  8 50 71 79 92 57 45 85  0] and Label = [1 0 0 1 1 0 1 1 1
0]
Feature = [14 41 59 29 10 68 35 81 97 66] and Label = [0 1 1 1 0 0 1 1 1
0]
Feature = [18 75 23 36 73 11 90 62 46 64] and Label = [0 1 1 0 1 1 0 0 0
0]
Feature = [ 1 26  5 13 56 60 37 65 42 17] and Label = [1 0 1 1 0 0 1 1 0
1]
Feature = [ 7 15 12 61 55 76 88 69 91 24] and Label = [1 1 0 1 1 0 0 1 1
0]
```

In [ ]:

```python
# Take

data = tf.data.Dataset.from_tensor_slices((features, labels))
data = data.repeat(10) # repeat 10 times
data = data.shuffle(len(data))
data = data.batch(10)
data = data.prefetch(1)
for batch_x, batch_y in data.take(8):
  print(f'Feature = {batch_x.numpy()} and Label = {batch_y.numpy()}')
```

```
Feature = [33 27 83 81 80 34 37 94 31 60] and Label = [1 1 1 1 0 0 1 0 1
0]
Feature = [74 40 44 32 82  3 54 87 48 79] and Label = [0 0 0 0 0 1 0 1 0
1]
Feature = [57 65 65 34 32 68 25 19 12 27] and Label = [1 1 1 0 0 0 1 1 0
1]
Feature = [19 83 90 30 29 26 35 97  8 24] and Label = [1 1 0 0 1 0 1 1 0
0]
Feature = [22 84 76 10 31 98  9 97 68 41] and Label = [0 0 0 0 1 0 1 1 0
1]
Feature = [39 96 59 98 61  7 31 16 92 80] and Label = [1 0 1 0 1 1 1 0 0
0]
Feature = [44  8 75 54 84 74 43 28 92 45] and Label = [0 0 1 0 0 0 1 0 0
1]
Feature = [15 39 56 83  5 58 46 68 50 46] and Label = [1 1 0 1 1 0 0 0 0
0]
```

In [ ]:

```
# Using Iterator - if we want to call multiple times

iter_data = iter(data)
for i in range(5):
    batch_x, batch_y = next(iter_data)
    print(f'Feature = {batch_x.numpy()} and Label = {batch_y.numpy()}')

print()

for i in range(10):
    batch_x, batch_y = next(iter_data)
    print(f'Feature = {batch_x.numpy()} and Label = {batch_y.numpy()}')
```

```
Feature = [60 80 96 70 64 22 63 37 99 60] and Label = [0 0 0 0 0 0 1 1 1
0]
Feature = [89 11 20 68 96 91 43 68 50  7] and Label = [1 1 0 0 0 1 1 0 0
1]
Feature = [12 24 33 30 63 24 93 18 62 50] and Label = [0 0 1 0 1 0 1 0 0
0]
Feature = [61 56 62 42 85 69 64 94 79 52] and Label = [1 0 0 0 1 1 0 0 1
0]
Feature = [89 26 33 90 41 93 57 65 94 71] and Label = [1 0 1 0 1 1 1 1 0
1]

Feature = [64 69  1 51 27  7 60 66 37 35] and Label = [0 1 1 1 1 1 0 0 1
1]
Feature = [39 96 93 59 66 61 94  9 53 47] and Label = [1 0 1 1 0 1 0 1 1
1]
Feature = [73 90 61 60 65 47 30 86 54 34] and Label = [1 0 1 0 1 1 0 0 0
0]
Feature = [46 67 21 47 30 88 85 22 41 26] and Label = [0 1 1 1 0 0 1 0 1
0]
Feature = [96 76 39 93 32 77 53 30 52 33] and Label = [0 0 1 1 0 1 1 0 0
1]
Feature = [92 35 35 32 46 85 31 63 22  8] and Label = [0 1 1 0 0 1 1 1 0
0]
Feature = [46 46 44 39  6 14 73 25 23 22] and Label = [0 0 0 1 0 0 1 1 1
0]
Feature = [48 57 77 42 82  9 91 40  7 38] and Label = [0 1 1 0 0 1 1 0 1
0]
Feature = [70 57 52 69 31 21 88 26 89 89] and Label = [0 1 0 1 1 1 0 0 1
1]
Feature = [89 20 25 90 67 83 16 43  2 76] and Label = [1 0 1 0 1 1 0 1 0
0]
```

Loading CSV files using TF2

In [ ]:

```
import requests

# Download Titanic dataset (in csv format).
d = requests.get("https://raw.githubusercontent.com/tflearn/tflearn.github.io/master/re
sources/titanic_dataset.csv")
with open("titanic_dataset.csv", "wb") as f: # temp file in File Storage section
    f.write(d.content)
```

In [ ]:

```
# Load the Titanic dataset

# Original features: survived,pclass,name,sex,age,sibsp,parch,ticket,fare
# Select specific columns: survived,pclass,name,sex,age,fare
column_to_use = [0, 1, 2, 3, 4, 8]
record_defaults = [tf.int32, tf.int32, tf.string, tf.string, tf.float32, tf.float32]

# Load the whole dataset file, and slice each line
data = tf.data.experimental.CsvDataset("titanic_dataset.csv", record_defaults, header=T
rue, select_cols=column_to_use)

# Shuffle data
data = data.shuffle(buffer_size=1000)

# Refill data indefinitely
data = data.repeat()

# Batch data (aggregate records together)
data = data.batch(batch_size=5)

# Prefetch batch (pre-load batch for faster consumption)
data = data.prefetch(buffer_size=1)
```

In [ ]:

```
# Snapshot
for survived, pclass, name, sex, age, fare in data.take(2):
    print(survived.numpy())
    print(pclass.numpy())
    print(name.numpy())
    print(sex.numpy())
    print(age.numpy())
    print(fare.numpy())
    print()
```

```
[0 0 0 0 1]
[3 2 1 3 1]
[b'Burns, Miss. Mary Delia' b'Greenberg, Mr. Samuel'
 b'Ross, Mr. John Hugo' b'Kraeff, Mr. Theodor'
 b'Marvin, Mrs. Daniel Warner (Mary Graham Carmichael Farquarson)']
[b'female' b'male' b'male' b'male' b'female']
[18. 52. 36.  0. 18.]
[ 7.8792 13.     40.125   7.8958 53.1   ]

[1 0 0 1 0]
[3 3 3 1 3]
[b'Hedman, Mr. Oskar Arvid' b'Hansen, Mr. Claus Peter'
 b'Cacic, Mr. Jego Grga' b'Chibnall, Mrs. (Edith Martha Bowerman)'
 b'Boulos, Mr. Hanna']
[b'male' b'male' b'male' b'female' b'male']
[27. 41. 18.  0.  0.]
[ 6.975  14.1083  8.6625 55.      7.225 ]
```

Neural Network Example using TF2

- Build a 2-hidden layers fully connected neural network (i.e. multilayer perceptron) with TensorFlow v2 for the FMNIST data

In [ ]:

```
# FMNIST dataset parameters
num_classes = 10 # total classes (0-9 classes)
num_features = 784 # data features (img shape: 28*28)

batch_size = 256
display_step = 100
```

In [ ]:

```
# Prepare Fashion MNIST data
(x_train, y_train), (x_test, y_test) = tf.keras.datasets.fashion_mnist.load_data()

# Convert to float32
x_train, x_test = np.array(x_train, np.float32), np.array(x_test, np.float32)

# Flatten images to 1-D vector of 784 features (28*28)
x_train, x_test = x_train.reshape([-1, num_features]), x_test.reshape([-1, num_feature
s])
# -1 in the first dimension means retain the first dimension as it is, reshape the late
r dimensions
# Normalize images value from [0, 255] to [0, 1]
x_train, x_test = x_train / 255.0, x_test / 255.0
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-d
atasets/train-labels-idx1-ubyte.gz
29515/29515 [==============================] - 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-d
atasets/train-images-idx3-ubyte.gz
26421880/26421880 [==============================] - 2s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-d
atasets/t10k-labels-idx1-ubyte.gz
5148/5148 [==============================] - 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-d
atasets/t10k-images-idx3-ubyte.gz
4422102/4422102 [==============================] - 0s 0us/step
```

In [ ]:

```
# Use tf.data API to shuffle, repeat, batch and pre-fetch data

train_data = tf.data.Dataset.from_tensor_slices((x_train, y_train))
train_data = train_data.shuffle(len(x_train)).repeat().batch(batch_size).prefetch(1)
```

In [ ]:

```
# Create TF network model

# Dense network parameters
n_hidden_1 = 256 # 1st layer number of neurons
n_hidden_2 = 128 # 2nd layer number of neurons

class NeuralNet(tf.keras.Model):
    # Set layers
    def __init__(self):
        super(NeuralNet, self).__init__()

        # First fully-connected hidden layer
        self.fc1 = tf.keras.layers.Dense(n_hidden_1, activation=tf.nn.relu)

        # Second fully-connected hidden layer
        self.fc2 = tf.keras.layers.Dense(n_hidden_2, activation=tf.nn.relu)

        # Output layer
        self.out = tf.keras.layers.Dense(num_classes)

    # Set forward pass
    def call(self, x, is_training=False):
        x = self.fc1(x)
        x = self.fc2(x)
        x = self.out(x)
        if not is_training:
            # tf cross entropy expect logits without softmax, so only
            # apply softmax when not training
            x = tf.nn.softmax(x)
        return x
```

In [ ]:

```
# Build the neural network model
neural_net = NeuralNet()
```

In [ ]:

```
# Cross-Entropy Loss
# Note that this will apply 'softmax' to the logits
# logit(p) = log[p/(1-p)]
# Probability of 0.5 corresponds to a logit of 0
# Negative logit correspond to probabilities less than 0.5, positive to > 0.5

# H(p, q) = - \sum_i p_i log q_i
# difference between two prob. distributions
# minimize this loss

def cross_entropy_loss(x, y):
    # Convert labels to int 64 for tf cross-entropy function
    y = tf.cast(y, tf.int64) # type-casting

    # Apply softmax to logits and compute cross-entropy
    loss = tf.nn.sparse_softmax_cross_entropy_with_logits(labels=y, logits=x)

    # Average loss across the batch
    return tf.reduce_mean(loss)
```

In [ ]:

```
# Accuracy metric
def accuracy(y_pred, y_true):
    # Predicted class is the index of highest score in prediction vector (i.e. argmax)
    y_pred = tf.argmax(y_pred, 1) # one-hot encoding
    y_true = tf.cast(y_true, tf.int64) # type-casting
    correct_prediction = tf.equal(y_pred, y_true)
    correct_prediction = tf.cast(correct_prediction, tf.float32)
    return tf.reduce_mean(correct_prediction, axis=-1) # avg accuracy accross the batch
```

In [ ]:

```
# Stochastic gradient descent optimizer
# Training parameters
learning_rate = 0.1
optimizer = tf.optimizers.SGD(learning_rate)
```

- Dense neural network
- 4 == 2 == 1
- How many parameters to be learned?
- 8 + 2 = 10
- 2 + 1 = 3
- 13 parameters
- Backprop: diff loss wrt all 13 params

In [ ]:

```
def run_optimization(x, y):
    # Wrap computation inside a GradientTape for automatic differentiation
    with tf.GradientTape() as tape:
        # Forward pass
        pred = neural_net(x, is_training=True)
        # Compute loss
        loss = cross_entropy_loss(pred, y) # objective function to be minimized

    # Variables to update, i.e. trainable variables
    trainable_variables = neural_net.trainable_variables
    # weights and biases to be learned by NN

    # Compute gradients
    grad = tape.gradient(loss, trainable_variables)
    # derivative of loss wrt all parameters

    # Update W and b following gradients
    optimizer.apply_gradients(zip(grad, trainable_variables))
    # W = W - lr * dL/dW
```

In [ ]:

```
training_steps = 2000

# Run training for the given number of steps, (2000 defined above)
for step, (batch_x, batch_y) in enumerate(train_data.take(training_steps), 1):
    # Run the optimization to update W and b values
    run_optimization(batch_x, batch_y)

    if step % display_step == 0:
        pred = neural_net(batch_x, is_training=True)
        loss = cross_entropy_loss(pred, batch_y)
        acc = accuracy(pred, batch_y)
        print("step: %i, loss: %f, accuracy: %f" % (step, loss, acc))
```

```
step: 100, loss: 0.729585, accuracy: 0.734375
step: 200, loss: 0.476698, accuracy: 0.832031
step: 300, loss: 0.594607, accuracy: 0.812500
step: 400, loss: 0.485102, accuracy: 0.816406
step: 500, loss: 0.401873, accuracy: 0.875000
step: 600, loss: 0.429214, accuracy: 0.859375
step: 700, loss: 0.384918, accuracy: 0.863281
step: 800, loss: 0.281053, accuracy: 0.902344
step: 900, loss: 0.350127, accuracy: 0.859375
step: 1000, loss: 0.379919, accuracy: 0.839844
step: 1100, loss: 0.353262, accuracy: 0.890625
step: 1200, loss: 0.353572, accuracy: 0.878906
step: 1300, loss: 0.314607, accuracy: 0.902344
step: 1400, loss: 0.295047, accuracy: 0.906250
step: 1500, loss: 0.314240, accuracy: 0.910156
step: 1600, loss: 0.275921, accuracy: 0.902344
step: 1700, loss: 0.264784, accuracy: 0.917969
step: 1800, loss: 0.306570, accuracy: 0.886719
step: 1900, loss: 0.200740, accuracy: 0.933594
step: 2000, loss: 0.281033, accuracy: 0.925781
```

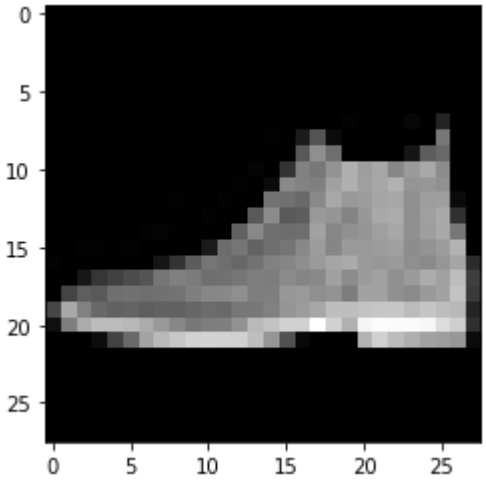
In [ ]:

```
# Test model on validation set
pred = neural_net(x_test, is_training=False)
print("Test Accuracy: %f" % accuracy(pred, y_test))
```

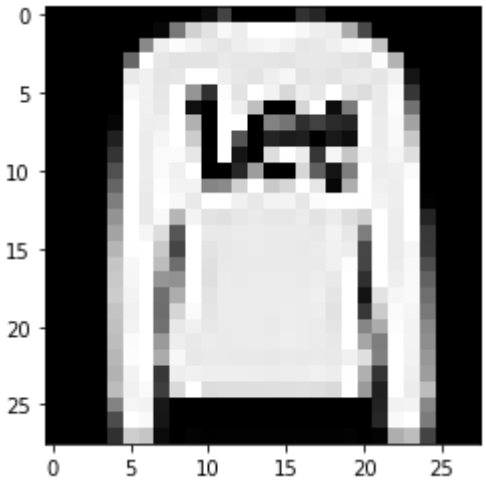
```
Test Accuracy: 0.861700
```

In [ ]:

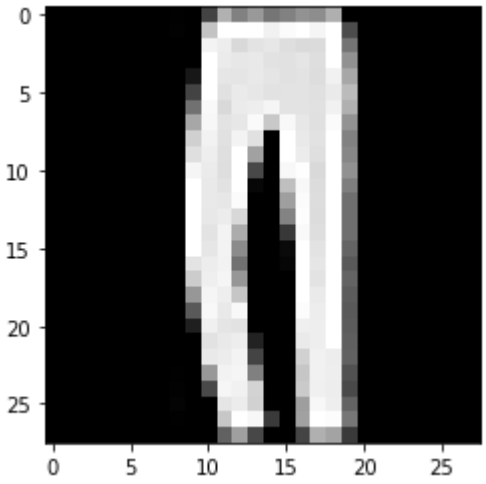
```
# Visualize predictions
import matplotlib.pyplot as plt

# Predict 5 images from test set
n_images = 5
test_images = x_test[:n_images] # slicing
predictions = neural_net(test_images)

# Display image and model prediction
for i in range(n_images):
    plt.imshow(np.reshape(test_images[i], [28, 28]), cmap='gray')
    plt.show()
    print("Model prediction: %i" % np.argmax(predictions.numpy()[i]))
    print("True label: %i" % y_test[i])
```



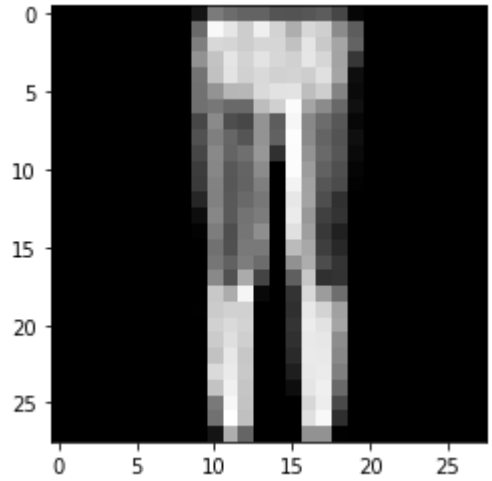
Model prediction: 9
True label: 9



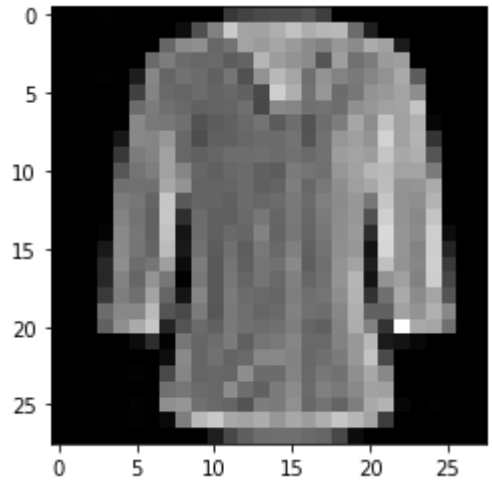
Model prediction: 2
True label: 2



Model prediction: 1
True label: 1




```
Model prediction: 1
True label: 1
```




```
Model prediction: 6
True label: 6
```

Save and Restore Models with TF2

In [ ]:

```
# Save TF model
neural_net.save_weights(filepath="./tfmodel")
```

In [ ]:

```
# Re-build neural network model with default initial values
neural_net_2 = NeuralNet()

# Test model performance
pred = neural_net_2(batch_x)

print("accuracy: %f" % accuracy(pred, batch_y))
```

```
accuracy: 0.074219
```

In [ ]:

```
# Load saved weights
neural_net_2.load_weights(filepath="./tfmodel")
```

Out[ ]:

```
<tensorflow.python.training.tracking.util.CheckpointLoadStatus at 0x7f5120
ac4310>
```

In [ ]:

```
# Test that weights loaded correctly
pred = neural_net_2(batch_x)
print("accuracy: %f" % accuracy(pred, batch_y))
```

```
accuracy: 0.925781
```