

# Advanced operations

Puteaux, Fall/Winter 2020-2021

```
#####  
##                               ##  
##  Deep Learning in Python  ##  
##                               ##  
#####
```

§2 Introduction to TensorFlow in Python

§2.1 Introduction to TensorFlow

§2.1.3 Advanced operations

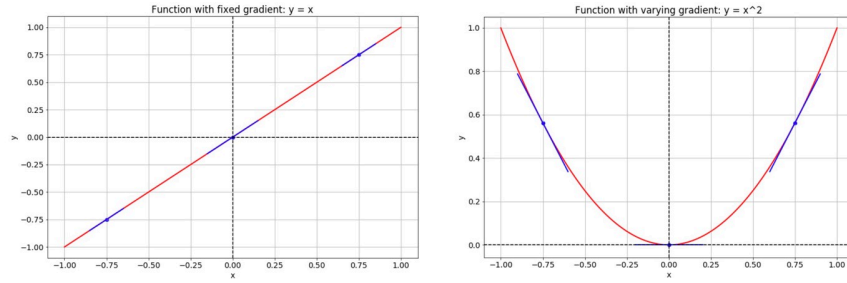
1. What are the advanced operations?

Operation	Use
<code>gradient()</code>	Computes the slope of a function at a point
<code>reshape()</code>	Reshapes a tensor (e.g. 10x10 to 100x1)
<code>random()</code>	Populates tensor with entries drawn from a probability distribution

2. How to find the optimum?

- In many problems, it is in need to find the optimum of a function:
  - **Minimum:** the lowest value of a loss function
  - **Maximum:** the highest value of the objective function
- It is possible to do this by using the `gradient()` operation:
  - **Optimum:** find a point where  $gradient = 0$
  - **Minimum:** change in  $gradient > 0$
  - **Maximum:** change in  $gradient < 0$

3. How to calculate the gradient?



#### 4. Code of gradients in TensorFlow:

```
[1]: # Import tensorflow under the alias tf
import tensorflow as tf

# Define x
x = tf.Variable(-1.0)

x
```

```
[1]: <tf.Variable 'Variable:0' shape=() dtype=float32, numpy=-1.0>
```

```
[2]: # Define y within instance of GradientTape
with tf.GradientTape() as tape:
    tape.watch(x)
    y = tf.multiply(x, x)

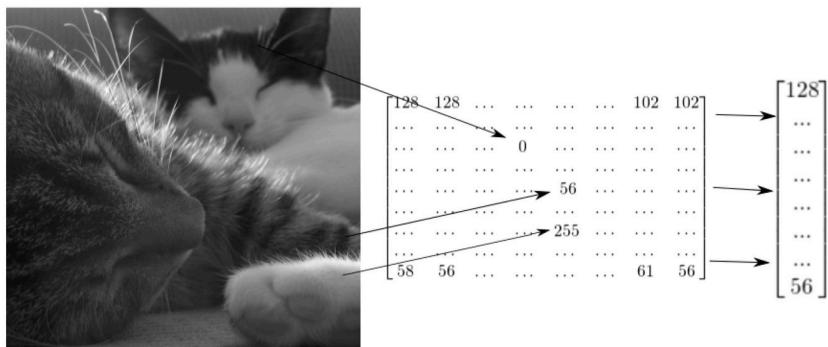
y
```

```
[2]: <tf.Tensor: shape=(), dtype=float32, numpy=1.0>
```

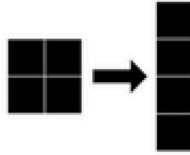
```
[3]: # Evaluate the gradient of y at x = -1
g = tape.gradient(y, x)
print(g.numpy())
```

```
-2.0
```

#### 5. How to deal with images as tensors?



## 6. How to reshape a grayscale image?



## 7. Code of reshaping a grayscale image:

```
[4]: # Import tensorflow as alias tf
import tensorflow as tf

# Generate grayscale image
gray = tf.random.uniform([2, 2], maxval=255, dtype='int32')

gray
```

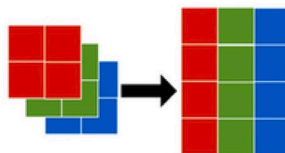
```
[4]: <tf.Tensor: shape=(2, 2), dtype=int32, numpy=
array([[ 42, 254],
       [ 15,  48]], dtype=int32)>
```

```
[5]: # Reshape grayscale image
gray = tf.reshape(gray, [2 * 2, 1])

gray
```

```
[5]: <tf.Tensor: shape=(4, 1), dtype=int32, numpy=
array([[ 42],
       [254],
       [ 15],
       [ 48]], dtype=int32)>
```

## 8. How to reshape a color image?



## 9. Code of reshaping a color image:

```
[6]: # Import tensorflow as alias tf
import tensorflow as tf

# Generate color image
```

```
color = tf.random.uniform([2, 2, 3], maxval=255, dtype='int32')

color
```

```
[6]: <tf.Tensor: shape=(2, 2, 3), dtype=int32, numpy=
      array([[193, 207,   2],
             [188,  27, 105]],

             [[138, 244,  80],
             [177,  52,  60]]], dtype=int32)>
```

```
[7]: # Reshape color image
color = tf.reshape(color, [2 * 2, 3])

color
```

```
[7]: <tf.Tensor: shape=(4, 3), dtype=int32, numpy=
      array([[193, 207,   2],
             [188,  27, 105],
             [138, 244,  80],
             [177,  52,  60]]], dtype=int32)>
```

## 10. Practice exercises for advanced operations:

### ► Diagram of images for reshaping:



### ► Package pre-loading:

```
[8]: import numpy as np
      from tensorflow import reshape
```

### ► Data pre-loading:

```
[9]: gray_tensor = np.loadtxt("ref11. Gray tensor.csv", delimiter=',')

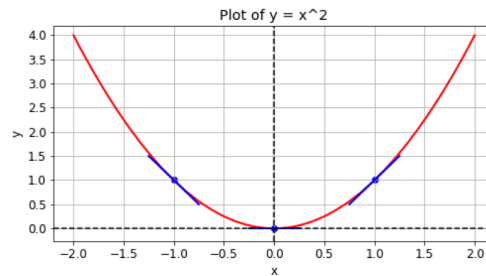
      color_tensor = np.loadtxt("ref12. Color tensor.csv", delimiter=',')
      color_tensor = color_tensor.reshape(color_tensor.shape[0],
                                           color_tensor.shape[1] // 3, 3)
```

### ► Tensor reshaping practice:

```
[10]: # Reshape the grayscale image tensor into a vector
gray_vector = reshape(gray_tensor, (-1, 1))

# Reshape the color image tensor into a vector
color_vector = reshape(color_tensor, (-1, 1))
```

► Diagram of gradient descent:



► Package re-pre-loading:

```
[11]: from tensorflow import Variable, GradientTape, multiply
```

► Gradients optimization practice:

```
[12]: def compute_gradient(x0):
    # Define x as a variable with an initial value of x0
    x = Variable(x0)
    with GradientTape() as tape:
        tape.watch(x)
        # Define y using the multiply operation
        y = multiply(x, x)
        # Return the gradient of y with respect to x
    return tape.gradient(y, x).numpy()

# Compute and print gradients at x = -1, 1, and 0
print(compute_gradient(-1.0))
print(compute_gradient(1.0))
print(compute_gradient(0.0))
```

```
-2.0
2.0
0.0
```

► Package re-pre-loading:

```
[13]: from tensorflow import constant, matmul, reduce_sum
```

► Data re-pre-loading:

```
[14]: letter = constant([[1., 0., 1.], [1., 1., 0.], [1., 0., 1.]])  
  
      model = constant([[1., 0., -1.]])
```

► Image data working practice:

```
[15]: # Reshape model from a 1x3 to a 3x1 tensor  
      model = reshape(model, (3, 1))  
  
      # Multiply letter by model  
      output = matmul(letter, model)  
  
      # Sum over output and print prediction using the numpy method  
      prediction = reduce_sum(output)  
      print(prediction.numpy())
```

1.0