- Complete the code for the following function

```
import numpy as np

def calculate_percent_error(self,YA,YT):
""" Given a batch of input, actual outputs, and desired outputs, this function
calculates percent error. For each sample, if the actual output vector is not
exactly the same as the desired output, it is considered one error.
Percent error is 100*(number_of_errors/ number_of_samples)

:param YA: Array of actual outputs [number_of_nodes, ,n_samples]. Assume that
the each element of YA is either 0 or 1 (Result of hard-limit activation
function).
param YT: Array of desired (target) outputs [number_of_nodes ,n_samples]
Assume that all the values of array YT are zero except one of them which is
equal to 1.
return percent_error"""
```

- Complete the following code. The code should compute the output, loss, and gradients, and perform a single weight update of a neural network with **784 inputs**, **100 nodes with sigmoid activation in the first layer, and 10 output nodes with linear activation**. Assume you have a batch of inputs called **X** that has dimensions **[16, 784]**, and a batch of targets called **y** that has dimensions **[16, 1]**

```
import tensorflow as tf
import numpy as np
def my_loss(y, y_hat):
    return tf.reduce_mean(
           tf.nn.sparse_softmax_cross_entropy_with_logits(
           labels=y,logits=y_hat))
# Create random weights and biases
W_1 = tf.Variable(np.random.randn(                  ), trainable=True)
b_1 = tf.Variable(np.random.randn(                  ), trainable=True)
W_2 = tf.Variable(np.random.randn(                  ), trainable=True)
b_2 = tf.Variable(np.random.randn(                  ), trainable=True)
with tf.GradientTape(persistent=True) as tape:
    # Calculate output
    output=
    # Calculate loss
    loss
    # Calculate gradients
    dW_1, db_1 =
    dW_2, db_2 =
# Calculate new values of weights and biases
```

- Complete the following function. Assume this function will be called by the main program shown below.

```
import numpy as np
import tensorflow.keras as keras
def get_biases(model, layer_number=None, layer_name=None):
"""This function returns the biases for a layer.
:param model: keras model
:param layer_number: Layer number starting from layer 0
:param layer_name: Layer name (if both layer_number and layer_name are specified, layer number takes precedence).
:return: biases for the given layer (If the given layer does not have bias then None should be returned)"""
```

```
my_model = keras.applications.VGG16(weights='imagenet', include_top=True)
for k in range(23):
    print(get_biases(model=my_model,layer_number=k))
print(get_biases(model=my_model,layer_name="fc1"))
```

- Consider a single neuron with linear activation : $\boldsymbol{output = Wx + b}$

  Write a Python function, using **Tensorflow** to adjust the weights and biases (one step) and return the gradients of loss with respect to $\boldsymbol{W}$ and gradients of loss with respect to $\boldsymbol{b}$

  Notes:

  Assume loss is the defined as the **MSE**

  You must use Tensorflow without using Keras

```
import tensorflow as tf
import numpy as np
def calculate_loss(x,y,w,b):
    # x: input batch (input_dimensions, batch_size)
    # y: desired output (output_dimensions, batch_size)
    # w: weight matrix (input_dimensions, output_dimensions)
    # b: bias (1, output_dimensions)
```

- Consider a convolutional neural network. The dimensions of layer K are given as 63x63x32.

  Layer K+1 is a convolutional layer with 16 filters of size 5 by 5 and stride=2 and padding =1

  a. Find the dimensions of the layer K+1

  b. How many distinct weights is shared between the filters in layer K+1. Assume there is no bias.

- Consider a convolutional neural network.
  Note: **Do NOT consider Biases**.

Input layer:

Input to this CNN are color images of size **64x52x3** with the batch size = 128

Note: Input image has **different horizontal and vertical** resolution.

Next layer is Conv2D layer:

number of filters: 30, filter size: 11x11 ; stride: 3x3 ; padding: 2x2

What is the shape of the weight matrix for this layer? _____

What is the shape of the output (tensor) of this layer? _____

Next layer is Flatten layer:

What is the shape of the output (tensor) for this layer? _____

Next layer is Dense layer:

number of nodes: 100

What is the shape of the weight matrix (tensor) for this layer? _____

What is the shape of the output (tensor) for this layer? _____

- Given the actual outputs and the desired output indexes of a neural network**, calculate and return the confusion matrix**. y_hat is a matrix presenting the actual output. Each row of this matrix is the actual output of the neural network for an input sample. The number of rows in this matrix is equal to the number of samples in the input batch. Y is the desired output. Each number in in the y is the index of the desired class for the corresponding input sample.
  Hint: This is similar to the function in Assignment 03.

```
import numpy as np
```

```
def calculate_confusion_matrix(y_hat, y):
""" Given the actual outputs and the desired output index of a neural
network, this function calculates the confusion matrix.
:param y_hat: Array of actual outputs [num_of samples,num_of classes]
:param y: Array of desired (target) outputs [num_of samples]. This
array includes the indexes of the desired (true) class.
:return confusion_matrix[number_of_classes,number_of_classes].
Confusion matrix should be shown as the number of times that an input
of class n is classified as class m."""
```

- Assuming that the actual output and the desired output of a neural network are given. Complete the code for the following function to calculate the overall cross entropy loss for a batch of data.y_hat is a matrix presenting the actual output. Each row of this matrix is the actual output of the neural network for an input sample. The number of rows in this matrix is equal to the number of samples in the input batch. y is the desired output. Each row is a one-hot representation of the desired class. This means that all entries in each row are zeros except one of them which is equal to 1 indicating the correct class.
  Notes:
  Do NOT use TensorFlow or Keras.
  You may use the numpy helper functions `np.nonzero()` and `np.log()` and `np.sum()`

```
def calculate_overall_cross_entropy_loss(y_hat,y):
Import numpy as np
""" This function calculates the overall cross entropy loss.
layer numbers start from zero.
:param y_hat: actual output [number_of_samples,number_of_classes].
:param Y: Desired output [number_of_samples,number_of_classes].
:return: overall cross-entropy loss
"""
```

- Consider a convolutional neural network.
  Note: **DO NOT consider biases**.

Input layer:

Input to this CNN are color images of size 64x64x3 with the batch size = 30

Next layer is Conv2D layer:

number of filters: 100, filter size: 7x7 ; stride: 3x3 ; padding: 3

What is the shape of the weight matrix (tensor) for this layer? _____

What is the shape of the output (tensor) of this layer? _____

Next layer is MaxPool2D:

pool size: 2x2 strides: 2x2 padding: 0 (Valid)

What is the shape of the output (tensor) for this layer? _____

Next layer is Flatten layer:

What is the shape of the output (tensor) for this layer? _____

Next layer is Dense layer:

number of nodes: 300

What is the shape of the weight matrix (tensor) for this layer? _____

What is the shape of the output (tensor) for this layer? _____

- Consider a convolutional neural network.
  Input to this CNN are color images of size 65x65x3.
  Batch size =100
  Notes:
  DO NOT consider biases.

First layer: filter size 11x11 ; stride: 2x2 ; padding: 0, number of filters: 18

What is the shape of the weight matrix (tensor) for the first layer? _____

What is the shape of the output (tensor) of the first layer? _____

Second layer: filter size 3x3 ; stride: 3x3 ; padding: 1, number of filters: 50

What is the shape of the weight matrix (tensor) for the second layer? _____

What is the shape of the output (tensor) of the second layer? _____

max_pool after the second layer: size: 4x4 strides: 2x2 padding: 0

What is the shape of the output (tensor) after max_pool layer? _____

Third layer: FC (fully connected) number of nodes=10.

What is the shape of the weight matrix (tensor) for the third layer? _____

- Consider the expression: $\boldsymbol{f(x,y) = x^2 + y}$
  Given the inputs $\boldsymbol{x = 2,}$ $\boldsymbol{y = 5}$, write a Python program, using Tensorflow to print the value of the output $\boldsymbol{f(x,y)}$ and partial derivatives of the $\boldsymbol{f(x,y)}$with respect to $\boldsymbol{x}$ and $\boldsymbol{y}$

```
import tensorflow as tf
import numpy as np
```

- Consider a convolutional neural network.
  Note: **DO NOT consider biases**.

Input to this CNN are color images of size 32x32x3.
Batch size =10

First layer: filter size 5x5 ; stride: 1x1 ; padding: "SAME", number of filters: 50

What is the shape of the weight matrix (tensor) for the first layer? _____

What is the shape of the output (tensor) of the first layer? _____

max_pool after the first layer: size: 2x2 strides: 2x2 padding: "SAME"

What is the shape of the output (tensor) after max_pool layer? _____

Second layer: filter size 3x3 ; stride: 1x1 ; padding: "SAME", number of filters: 64

What is the shape of the weight matrix (tensor) for the second layer? _____

What is the shape of the output (tensor) of the second layer? _____

- Write a program using Tensorflow to implement a neural network with one hidden layer and one output layer.

  The dimension of the input data data : **3**

  Number of nodes in hidden layer: **10**, activation: **sigmoid**

  Number of nodes in the output layer: **5** activation : **linear**

  Just implement the forward path to calculate and display output. **No training loop , or calculation of gradients.**