✓ **Congratulations! You passed!**

TO PASS 80% or higher

Keep Learning

GRADE
100%

Image Segmentation

LATEST SUBMISSION GRADE

100%

1. At the heart of image segmentation with neural networks is an encoder/decoder architecture. What functionalities do they perform ?

1 / 1 point

- [x] The encoder extracts features from an image and the decoder takes those extracted features, and assigns class labels to each pixel of the image.

✓ **Correct**
Correct!

- [ ] The decoder extracts features from an image and the encoder takes those extracted features, and assigns class label to the entire image.
- [ ] The encoder extracts features from an image and the decoder takes those extracted features, and assigns class label to the entire image.
- [ ] The decoder extracts features from an image and the encoder takes those extracted features, and assigns class labels to each pixel of the image.

2. Is the following statement true regarding SegNet, UNet and Fully Convolutional Neural Networks (FCNNs):

1 / 1 point

Unlike the similarity between the architecture design of SegNet & UNet, FCNNs do not have a symmetric architecture design.

[✓ True]
False

✓ **Correct**
Correct!

3. What architectural difference does the *number*represent in the names of FCN-32, FCN-16, FCN-8 ?

1 / 1 point

[✓ The *number* represents the factor by which the final pooling layer in the architecture up-samples the image to make predictions.]

The *number* represents the total number of pooling layers used in the architecture to help make predictions.

The *number* represents the total number of convolutional layers used in the final pooling layer in the architecture to make predictions.

The *number* represents the total number of filters used in the final pooling layer in the architecture to make predictions.

✓ **Correct**
Correct!

4. Take a look at the following code and select the type of scaling that will be performed

1 / 1 point

```
x = UpSampling2D(
    size=(2, 2),
    data_format=None,
    interpolation='bilinear')(x)
```

The upsampling of the image will be done by copying the value from the closest pixels.

[✓ The upsampling of the image will be done by means of linear interpolation from the closest pixel values]

✓ **Correct**
Correct!

5. What does the following code do?

1 / 1 point

```
    Conv2DTranspose(
        filters=32,
        kernel_size=(3, 3)
    )
```

It takes pixel values in the image, in a 3x3 array, and using the specified filters, creates a transpose of that array.

[✓ It takes the pixel values and filters and tries to reverse the convolution process to return back a 3x3 array which *could* have been the original array of the image.]

✓ **Correct**
Correct!

6. The following is the code for the *last layer* of a FCN-8 decoder. What *key change* is required if we want this to be the *last layer* of a FCN-16 decoder ?

1 / 1 point

```
def fcn8_decoder(convs, n_classes):
  …

  o = tf.keras.layers.Conv2DTranspose(n_classes , kernel_size=(8,8),
                                      strides=(8,8))(o)

  o = (tf.keras.layers.Activation('softmax'))(o)

  return o
```

○ *n_classes=16*

○ Using *sigmoid* instead of *softmax.*

● *kernel_size=(16, 16)*

○ *strides=(16, 16)*

✓ **Correct**
Correct!

7. Which of the following is true about Intersection Over Union (IoU) and Dice Score, when it comes to evaluating image segmentation? (Choose all that apply.)

1 / 1 point

☐ For both, IoU & Dice Score the denominator is the total area of both the labels, predicted and ground truth

☐ Unlike IoU, for Dice Score the closer the value is to 0 the closer the prediction is to the ground truth.

☑ For IoU the numerator is the area of overlap for both the labels, predicted and ground truth, whereas for Dice Score the numerator is 2 times that.

✓ **Correct**
Correct!

☑ Both have a range between 0 and 1

✓ **Correct**
Correct!

8. Consider the following code for building the *encoder blocks* for a *U-Net.* What should this function return?

1 / 1 point

```
def unet_encoder_block(inputs, n_filters, pool_size, dropout):
  blocks = conv2d_block(inputs, n_filters=n_filters)
  after_pooling = tf.keras.layers.MaxPooling2D(pool_size)(blocks)
  after_dropout = tf.keras.layers.Dropout(dropout)(after_pooling)

  return # your code here
```

○ *after_dropout, after_pooling* (you need to return *after_pooling* to be used in skip connections)

● *blocks, after_dropout*

○ *blocks*

○ *after_dropout*

✓ **Correct**
Correct!

9. For U-Net, on the *decoder* side you combine *skip connections* which come from the corresponding level of the *encoder.* Consider the following code and provide the missing line required to account for those skip connections with the upsampling.

1 / 1 point

(**Important Notes**: Use TensorFlow as *tf,* Keras as *keras.* And be mindful of python spacing convention, i.e (x, y) *not* (x,y))

```
def decoder_block(inputs, conv_output, n_filters, kernel_size, strides, dropout):
  upsampling_layer = tf.keras.layers.Conv2DTranspose(n_filters, kernel_size, strides = strides,
   padding = 'same')(inputs)

  skip_connection_layer = # your code here
  skip_connection_layer = tf.keras.layers.Dropout(dropout)(skip_connection_layer)
  skip_connection_layer = conv2d_block(skip_connection_layer, n_filters, kernel_size=3)

  return skip_connection_layer
```

tf.keras.layers.concatenate([upsampling_layer, conv_output])

✓ **Correct**
Correct!