! **Try again once you are ready**

TO PASS 80% or higher

Try again

GRADE
75%

Style Transfer

LATEST SUBMISSION GRADE
75%

1. In Neural Style Transfer when initializing the *generated image* from the *content image*,which of the following is true? Check all that apply. **0.75 / 1 point**

- [ ] Initially the *style loss* will be equal or close to zero because both, the *content* and *generated*, images are the same.
- [x] Initially the *content loss* will be equal or close to zero because both the *content image* and *generated* image are the same image.

✓ **Correct**
Correct!

- [ ] Your goal for the *generated image* is to increase the *content loss* and decrease the *style loss*, while keeping the overall *accumulated loss* low.
- [ ] Your goal for the *generated image* is to increase the *style loss* and decrease the *content loss* while keeping the overall *accumulated loss* low.

You didn't select all the correct answers

2. What does tf.keras.applications.vgg19.preprocess_input do? **0 / 1 point**

- (•) The function sets the pixel values of an image between 0 and 1.
- ( ) The function centers the distribution of pixel values of an image around zero.

! **Incorrect**
Incorrect! This is called normalizing an image, and *preprocess_input* does not perform normalization.

3. From which part of a CNN architecture can you extract the "content" of an image? **1 / 1 point**

- (•) From the deeper layers of the architecture.
- ( ) The initial layers of the architecture.

✓ **Correct**
Correct! If you recall the lecture we used only the deeper layer of the CNN for computing *content loss*because that layer holds the information of the *content* of an image.

4. Consider the values given in the image below and calculate the *content loss* value. **1 / 1 point**

Generated image

5	2
1	7

Content image

3	5
5	4

19

✓ **Correct**
Correct!

5. Fill in the missing code below: **0.75 / 1 point**

```
def get_content_loss(generated_image, content_image):
    return # YOUR CODE HERE
```

- [ ] tf.reduce_sum(tf.square(generated_image - content_image))
- [ ] 0.5 * tf.reduce_sum(tf.square(content_image - generated_image))
- [ ] tf.reduce_sum(tf.square(content_image - generated_image))
- [x] 0.5 * tf.reduce_sum(tf.square(generated_image - content_image))

✓ **Correct**
Correct!

You didn't select all the correct answers

6. Consider the following code snippet. How will you include *Total Loss Variation* in it? Use TensorFlow as *tf*. **1 / 1 point**

(Answer in the format, **x + y(z)**, considering python's spacing convention)

```
def calculate_gradients(image, content_targets,
                        style_targets, style_weight,
                        content_weight,with_regularization=True):

    total_variation_weight = 30
    with tf.GradientTape() as tape:
      if with_regularization:
        loss += # YOUR CODE HERE

    gradients = tape.gradient(loss, image)
    return gradients
```

total_variation_weight * tf.image.total_variation(image)

✓ **Correct**
Correct!