

## Your grade: 100%

Your latest: 100% • Your highest: 100% • To pass you need at least 80%. We keep your highest score.

[Next item →](#)

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

1 / 1 point

- ☒ 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.

✓ **Correct**  
Correct! Since the *generated image* is initialized from the *content image*, you want it to inherit attributes from the *style image* (reduce *style loss*), but also not lose all of its attributes inherited from the *content image* (increase *content loss*).

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

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

2. What does `tf.keras.applications.vgg19.preprocess_input` do?

1 / 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.

✓ **Correct**  
Correct! This is called standardization.

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:

1 / 1 point

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

☒ `0.5 * tf.reduce_sum(tf.square(generated_image - content_image))`

☒ **Correct**  
Correct!

☒ `0.5 * tf.reduce_sum(tf.square(content_image - generated_image))`

☒ **Correct**  
Correct! Even though the original formula is *generated\_image - content\_image*, since you are *squaring* the difference it doesn't matter what you subtract out of what.

☐ `tf.reduce_sum(tf.square(generated_image - content_image))`

☐ `tf.reduce_sum(tf.square(content_image - generated_image))`

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!