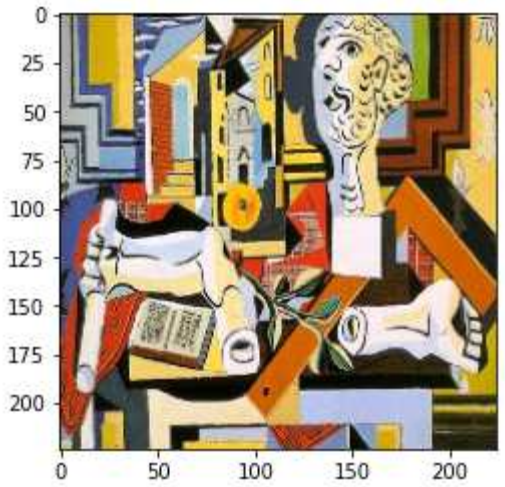
8.3.1 컨볼루션 신경망을 사용한 텍스쳐 합성

```
[ ]  1 from google.colab import drive
     2 drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount
```

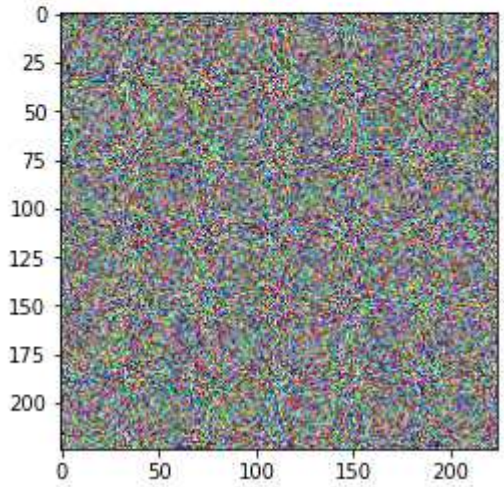
```
 1 # 8.42 원본 텍스쳐 이미지 불러오기
 2 import tensorflow as tf
 3 import matplotlib.pyplot as plt
 4 import cv2
 5
 6 # 스타일 이미지
 7 style_path=r'/content/drive/MyDrive/pp.jpg'
 8
 9 style_image = plt.imread(style_path)
10 style_image = cv2.resize(style_image, dsize=(224, 224))
11 style_image = style_image / 255.0
12 plt.imshow(style_image)
```

```
<matplotlib.image.AxesImage at 0x7fbc037c0a20>
```



```
# 8.43 타겟 텍스쳐 만들기
target_image = tf.random.uniform(style_image.shape)
print(target_image[0,0,:])
plt.imshow(target_image)
```

```
tf.Tensor([0.49465215 0.15899134 0.05912387], shape=(3,), dtype=float32)
<matplotlib.image.AxesImage at 0x7fbc03797358>
```




```
# 8.44 VGG-19 네트워크 불러오기
from tensorflow.keras.applications import VGG19
from tensorflow.keras.applications.vgg19 import preprocess_input

vgg = VGG19(include_top=False, weights='imagenet')

for layer in vgg.layers:
    print(layer.name)
```

```
input_2
block1_conv1
block1_conv2
block1_pool
block2_conv1
block2_conv2
block2_pool
block3_conv1
block3_conv2
block3_conv3
block3_conv4
block3_pool
block4_conv1
block4_conv2
```

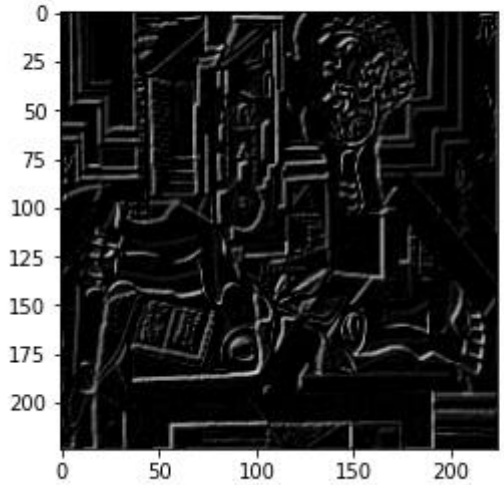
```
[ ]  1 # 8.45 특징 추출 모델 만들기
     2 style_layers = ['block1_conv1',
     3                 'block2_conv1',
     4                 'block3_conv1',
     5                 'block4_conv1',
     6                 'block5_conv1']
     7
     8 vgg.trainable = False
     9 outputs = [vgg.get_layer(name).output for name in style_layers]
    10 model = tf.keras.Model([vgg.input], outputs)
```

```
[ ]  1 # 8.46 Gram matrix 계산 함수 정의
     2 def gram_matrix(input_tensor):
     3     channels = int(input_tensor.shape[-1])
     4     a = tf.reshape(input_tensor, [-1, channels])
     5     n = tf.shape(a)[0]
     6     gram = tf.matmul(a, a, transpose_a=True)
     7     return gram / tf.cast(n, tf.float32)
```

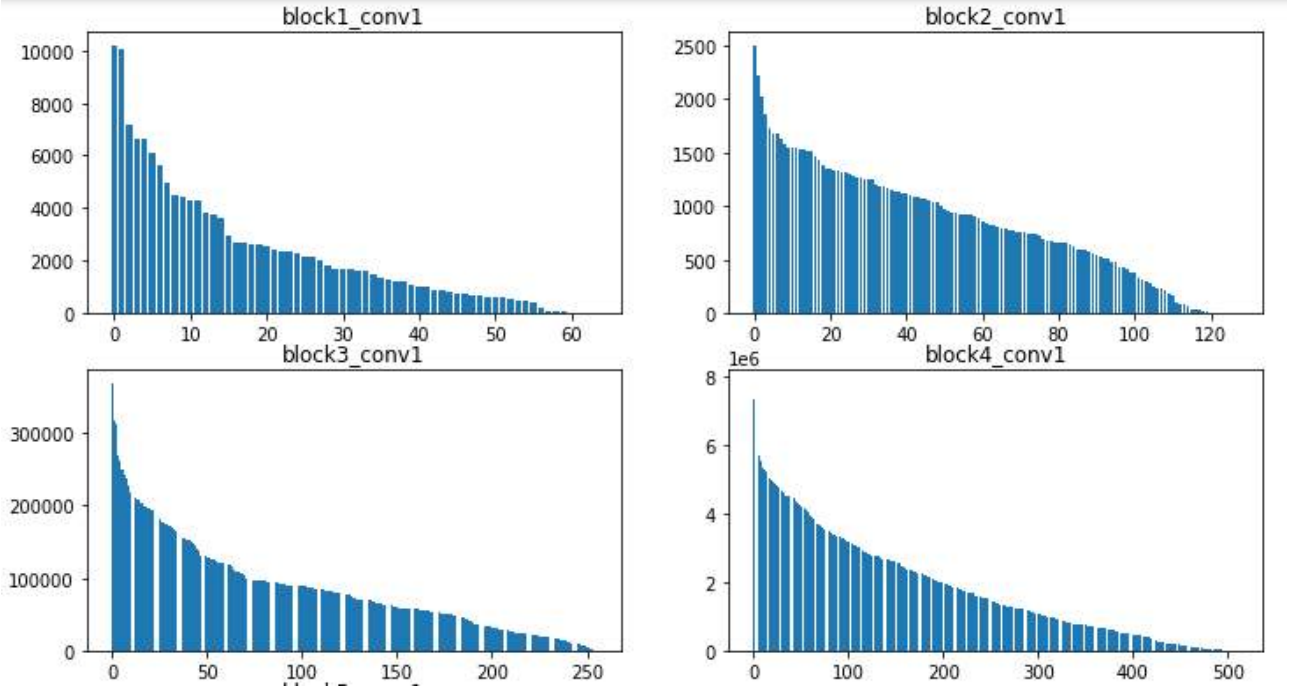
```
1 # 8.47 원본 텍스쳐에서 gram matrix 계산
2 style_image = plt.imread(style_path)
3 style_image = cv2.resize(style_image, dsize=(224, 224))
4 style_image = style_image / 255.0
5
6 style_batch = style_image.astype('float32')
7 style_batch = tf.expand_dims(style_batch, axis=0)
8 style_output = model(preprocess_input(style_batch * 255.0))
```

```
1 # 8.48 원본 텍스쳐의 첫번째 특징 추출값 확인
2 print(style_output[0].shape)
3 plt.imshow(tf.squeeze(style_output[0][:,:,:,0], 0), cmap='gray')
```

```
(1, 224, 224, 64)
<matplotlib.image.AxesImage at 0x7fbc036ea128>
```



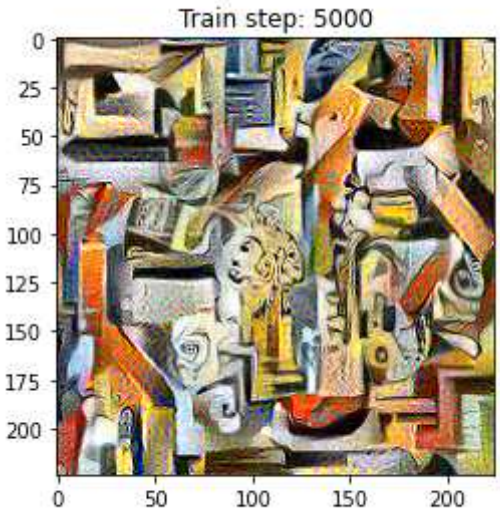
```
# 8.49 원본 텍스쳐의 gram matrix 계산값 만들기, 분포 확인
style_outputs = [gram_matrix(out) for out in style_output]

plt.figure(figsize=(12,10))
for c in range(5):
    plt.subplot(3,2,c+1)
    array = sorted(style_outputs[c].numpy()[0].tolist())
    array = array[::-1]
    plt.bar(range(style_outputs[c].shape[0]), array)
    plt.title(style_layers[c])
plt.show()
```



```python
# 8.50 타겟 텍스쳐를 업데이트하기 위한 함수 정의
def get_outputs(image):
    image_batch = tf.expand_dims(image, axis=0)
    output = model(preprocess_input(image_batch * 255.0))
    outputs = [gram_matrix(out) for out in output]
    return outputs

def get_loss(outputs, style_outputs):
    return tf.reduce_sum([tf.reduce_mean((o-s)**2) for o,s in zip(outputs, style_outputs)])

def clip_0_1(image):
    return tf.clip_by_value(image, clip_value_min=0.0, clip_value_max=1.0)
```

```python
# 8.51 tf.function과 GradientTape을 이용한 이미지 업데이트 함수 정의
opt = tf.optimizers.Adam(learning_rate=0.2, beta_1=0.99, epsilon=1e-1)

@tf.function()
def train_step(image):
    with tf.GradientTape() as tape:
        outputs = get_outputs(image)
        loss = get_loss(outputs, style_outputs)

    grad = tape.gradient(loss, image)
    opt.apply_gradients([(grad, image)])
    image.assign(clip_0_1(image))
```

```
# 8.52 텍스쳐 합성 알고리즘 실행
import IPython.display as display
import time
import imageio

start = time.time()

image = tf.Variable(target_image)

epochs = 50
steps_per_epoch = 100

step = 0
for n in range(epochs):
    for m in range(steps_per_epoch):
        step += 1
        train_step(image)
    if n % 5 == 0 or n == epochs - 1:
        imageio.imwrite('style_epoch_{0}.png'.format(n), image.read_value().numpy())
    display.clear_output(wait=True)
    plt.imshow(image.read_value())
    plt.title("Train step: {}".format(step))
    plt.show()

end = time.time()
print("Total time: {:.1f}".format(end-start))
```




```
Total time: 67.8
```

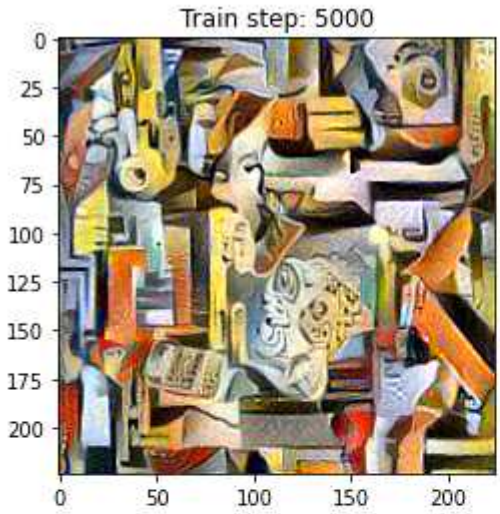
```python
# 8.53 varitation loss 함수 정의
def high_pass_x_y(image):
    x_var = image[:,1:,:] - image[:,:-1,:]
    y_var = image[1:,:,:] - image[:-1,:,:]
    return x_var, y_var

def total_variation_loss(image):
    x_deltas, y_deltas = high_pass_x_y(image)
    return tf.reduce_mean(x_deltas**2) + tf.reduce_mean(y_deltas**2)
```

```python
# 8.54 variation loss 비교
print('target   :', total_variation_loss(image.read_value()))
print('noise    :', total_variation_loss(tf.random.uniform(style_image.shape)))
print('original :', total_variation_loss(style_image))
```

```
target   : tf.Tensor(0.11866248, shape=(), dtype=float32)
noise    : tf.Tensor(0.33224145, shape=(), dtype=float32)
original : tf.Tensor(0.055965573881594025, shape=(), dtype=float64)
```

```python
# 8.55 variation loss를 loss 계산식에 추가, 각 loss의 가중치 추가
total_variation_weight = 1e9
style_weight = 1e-1

@tf.function()
def train_step(image):
    with tf.GradientTape() as tape:
        outputs = get_outputs(image)
        loss = style_weight * get_loss(outputs, style_outputs)
        loss += total_variation_weight * total_variation_loss(image)

    grad = tape.gradient(loss, image)
    opt.apply_gradients([(grad, image)])
    image.assign(clip_0_1(image))
```

```python
# 8.56 variation loss를 추가한 텍스쳐 합성 알고리즘 실행
start = time.time()

target_image = tf.random.uniform(style_image.shape)
image = tf.Variable(target_image)

epochs = 50
steps_per_epoch = 100

step = 0
for n in range(epochs):
    for m in range(steps_per_epoch):
        step += 1
        train_step(image)
    if n % 5 == 0 or n == epochs - 1:
        imageio.imwrite('style_variation_epoch_{0}.png'.format(n), image.read_value().numpy())
    display.clear_output(wait=True)
    plt.imshow(image.read_value())
    plt.title("Train step: {}".format(step))
    plt.show()

end = time.time()
print("Total time: {:.1f}".format(end-start))
```




```
Total time: 68.2
```

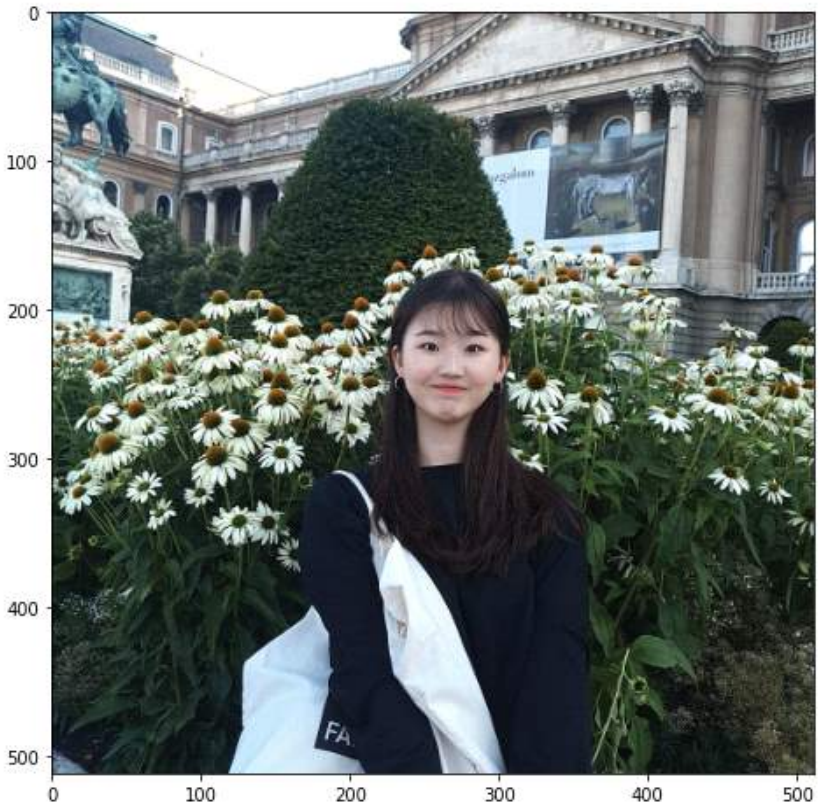
```python
# 8.57 원본과 타겟의 variation loss 비교
print('target   :', total_variation_loss(image.read_value()))
print('original :', total_variation_loss(style_image))
```

```
target   : tf.Tensor(0.04533544, shape=(), dtype=float32)
original : tf.Tensor(0.055965573881594025, shape=(), dtype=float64)
```

8.3.2 컨볼루션 신경망을 사용한 Style Transfer (Neural Style Transfer)

```
[ ]
# 8.58 content 텍스쳐 불러오기
import matplotlib.pyplot as plt
import cv2

# content_path = tf.keras.utils.get_file('content.jpg', 'http://bit.ly/2mAfUX1')
content_path=r'/content/drive/MyDrive/soob.jpg'

content_image = plt.imread(content_path)
# content_image=cv2.rotate(content_image, cv2.ROTATE_90_CLOCKWISE)  # 사진 회전되어 있으면
max_dim = 512
long_dim = max(content_image.shape[:-1])
scale = max_dim / long_dim
new_height = int(content_image.shape[0] * scale)
new_width = int(content_image.shape[1] * scale)

content_image = cv2.resize(content_image, dsize=(new_width, new_height))
content_image = content_image / 255.0
plt.figure(figsize=(8,8))
plt.imshow(content_image)
```

```
<matplotlib.image.AxesImage at 0x7fbd05664f98>
```

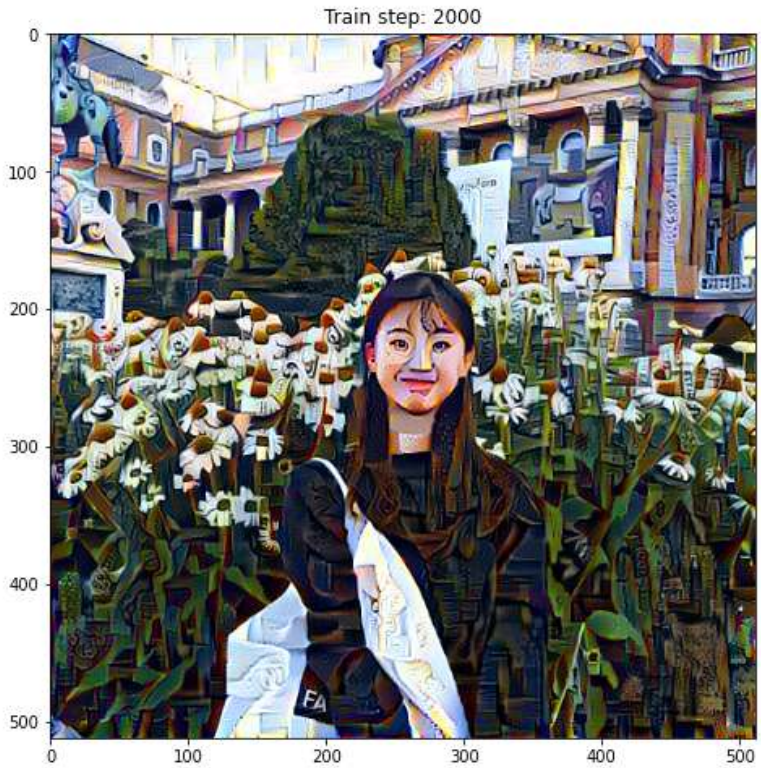


```python
# 8.59 content 특징 추출 모델 만들기
content_batch = content_image.astype('float32')
content_batch = tf.expand_dims(content_batch, axis=0)

content_layers = ['block5_conv2']

vgg.trainable = False
outputs = [vgg.get_layer(name).output for name in content_layers]
model_content = tf.keras.Model([vgg.input], outputs)
content_output = model_content(preprocess_input(content_batch * 255.0))
```

```python
# 8.60 content output, loss 함수 정의
def get_content_output(image):
    image_batch = tf.expand_dims(image, axis=0)
    output = model_content(preprocess_input(image_batch * 255.0))
    return output

def get_content_loss(image, content_output):
    return tf.reduce_sum(tf.reduce_mean(image-content_output)**2)
```

```python
# 8.61 content loss를 loss 계산식에 추가
opt = tf.optimizers.Adam(learning_rate=0.001, beta_1=0.99, epsilon=1e-1)

total_variation_weight = 1e9
style_weight = 1e-2
content_weight = 1e4

@tf.function()
def train_step(image):
    with tf.GradientTape() as tape:
        outputs = get_outputs(image)
        output2 = get_content_output(image)
        loss = style_weight * get_loss(outputs, style_outputs)
        loss += content_weight * get_content_loss(output2, content_output)
        loss += total_variation_weight * total_variation_loss(image)

    grad = tape.gradient(loss, image)
    opt.apply_gradients([(grad, image)])
    image.assign(clip_0_1(image))
```

```python
# 8.62 Neural Style Transfer 실행
start = time.time()

# target_image = tf.random.uniform(content_image.shape)
image = tf.Variable(content_image.astype('float32'))

epochs = 20
steps_per_epoch = 100

step = 0
for n in range(epochs):
    for m in range(steps_per_epoch):
        step += 1
        train_step(image)
        print(".", end='')
    if n % 5 == 0 or n == epochs - 1:
        imageio.imwrite('style_{0}_content_{1}_transfer_epoch_{2}.png'
        .format(style_weight, content_weight, n), image.read_value().numpy())
    display.clear_output(wait=True)
    plt.figure(figsize=(8,8))
    plt.imshow(image.read_value())
    plt.title("Train step: {}".format(step))
    plt.show()

end = time.time()
print("Total time: {:.1f}".format(end-start))
```




```
Total time: 164.5
```

```
# 그림 8.24 출력 코드
style_image = plt.imread(style_path)
style_image = cv2.resize(style_image, dsize=(224, 224))
style_image = style_image / 255.0

style_batch = style_image.astype('float32')
style_batch = tf.expand_dims(style_batch, axis=0)
style_output = model(preprocess_input(style_batch * 255.0))

plt.figure(figsize=(16,16))

for c in range(style_output[0].shape[-1]):
    plt.subplot(8,8,c+1)
    plt.axis('off')
    plt.imshow(tf.squeeze(style_output[0][:,:,:,c], 0), cmap='gray')
```