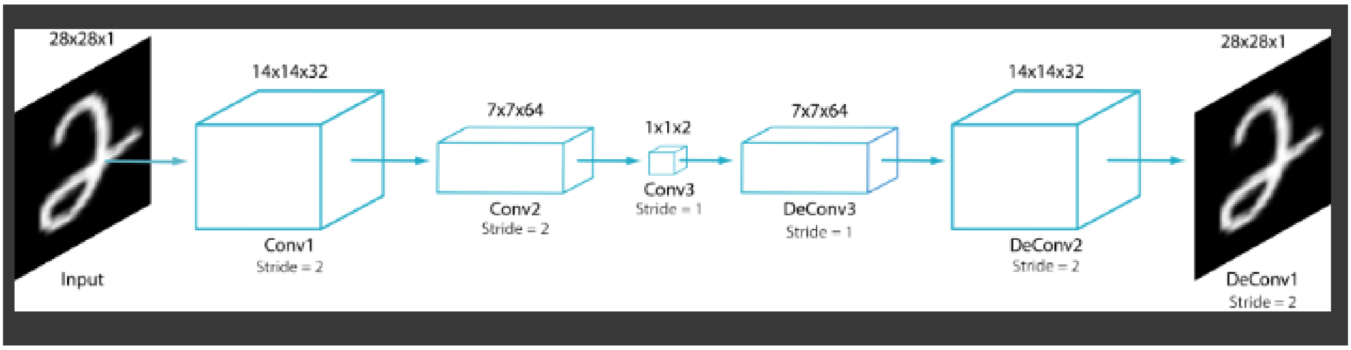
디지털 영상처리 연구실 연구보고서

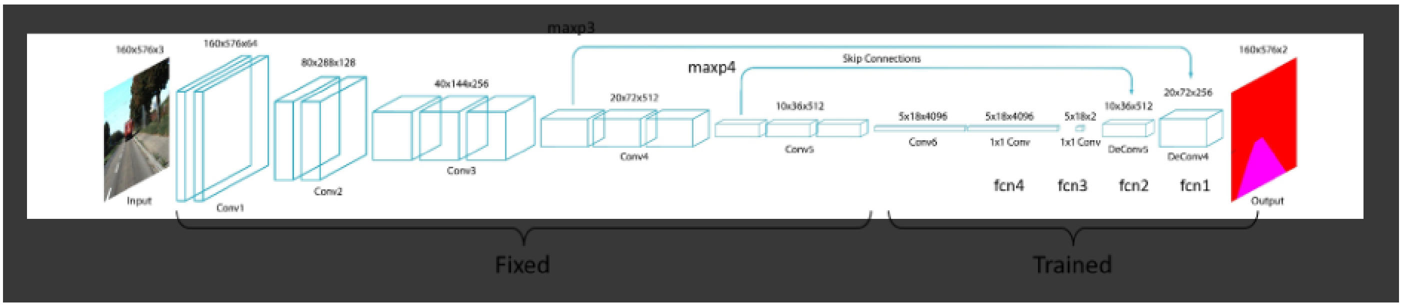
김우헌

#Fully Convolutional Networks (FCN) for Segmentation

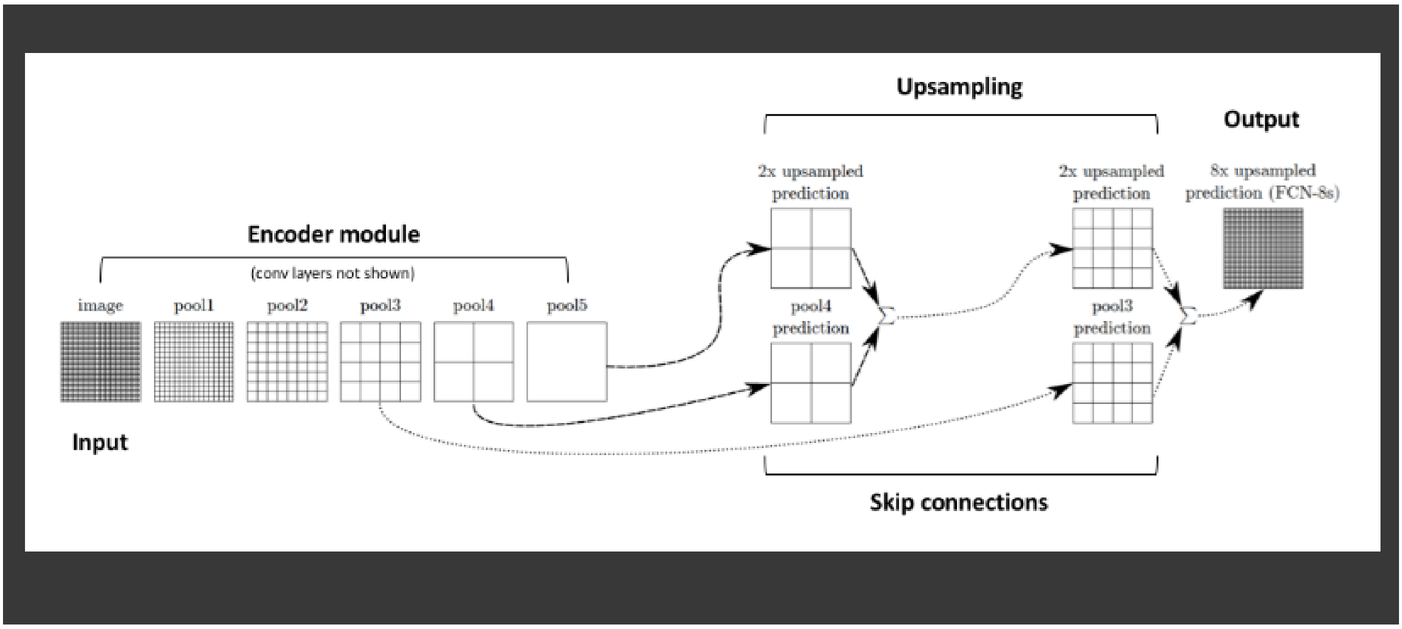
## Convolutional Auto Encoder(CAE)



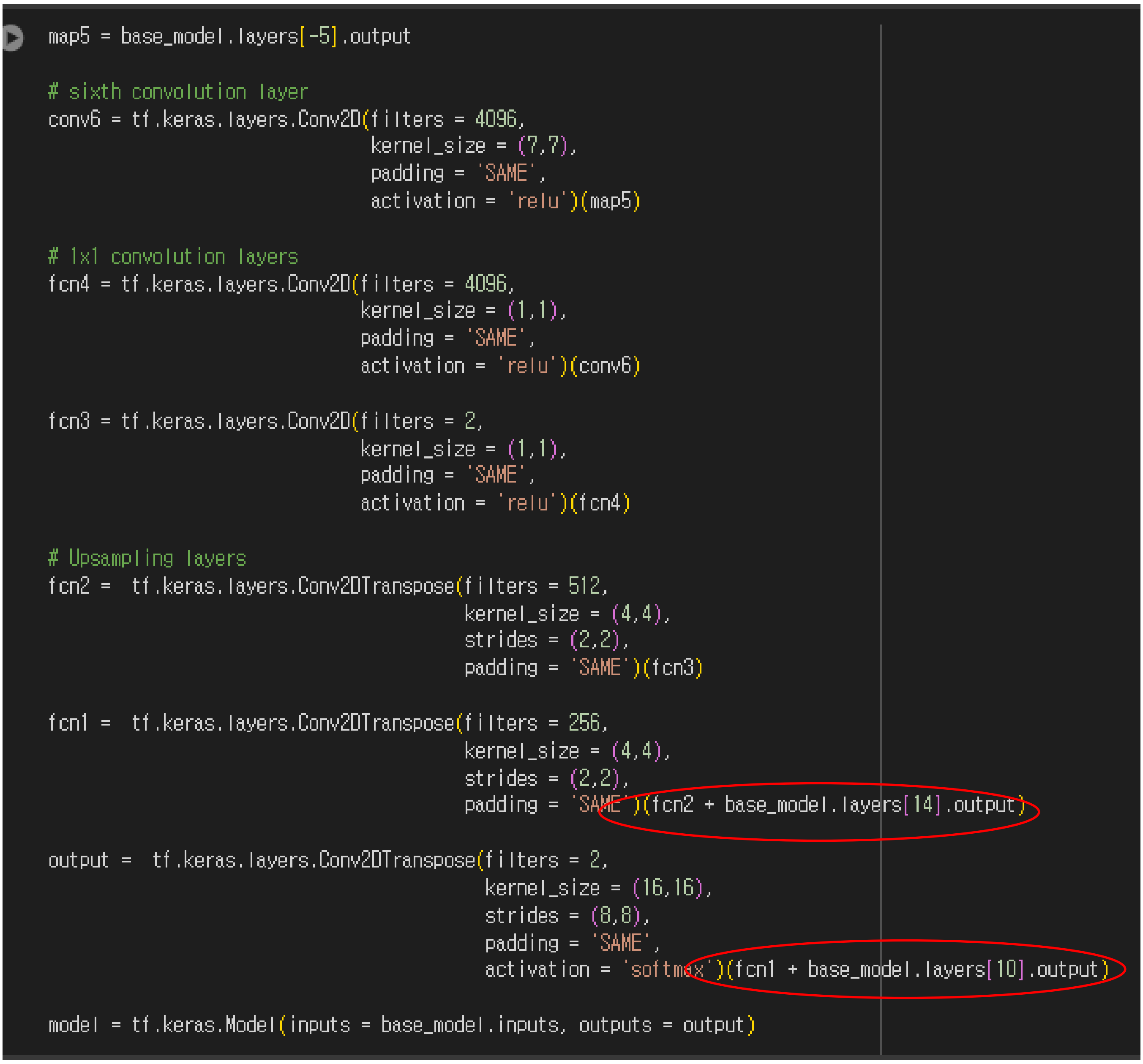
## FCN



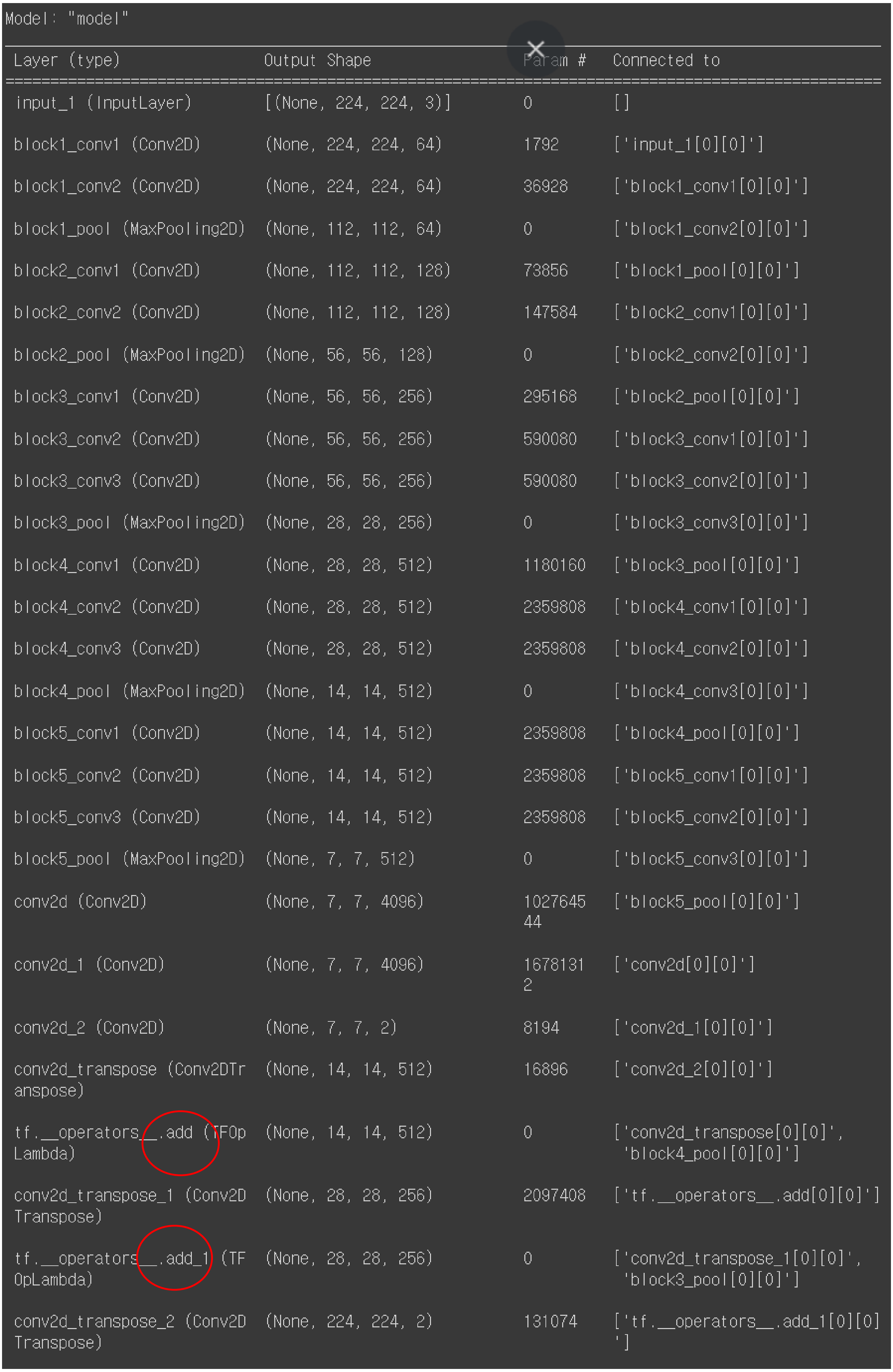
- skip connection



- FCN model 생성

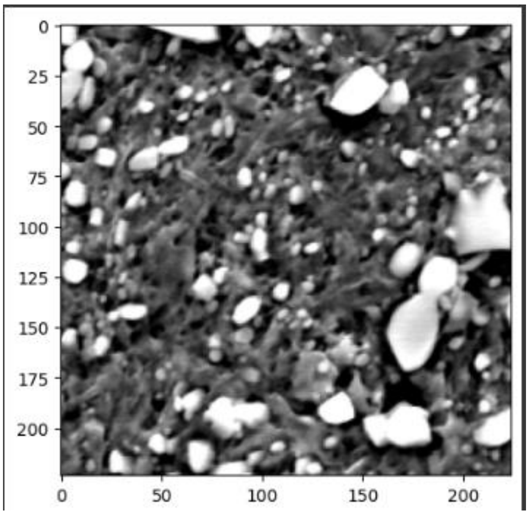


```
map5 = base_model.layers[-5].output

# sixth convolution layer
conv6 = tf.keras.layers.Conv2D(filters = 4096,
                               kernel_size = (7,7),
                               padding = 'SAME',
                               activation = 'relu')(map5)

# 1x1 convolution layers
fcn4 = tf.keras.layers.Conv2D(filters = 4096,
                              kernel_size = (1,1),
                              padding = 'SAME',
                              activation = 'relu')(conv6)

fcn3 = tf.keras.layers.Conv2D(filters = 2,
                              kernel_size = (1,1),
                              padding = 'SAME',
                              activation = 'relu')(fcn4)

# Upsampling layers
fcn2 =  tf.keras.layers.Conv2DTranspose(filters = 512,
                                        kernel_size = (4,4),
                                        strides = (2,2),
                                        padding = 'SAME')(fcn3)

fcn1 =  tf.keras.layers.Conv2DTranspose(filters = 256,
                                        kernel_size = (4,4),
                                        strides = (2,2),
                                        padding = 'SAME')(fcn2 + base_model.layers[14].output)

output =  tf.keras.layers.Conv2DTranspose(filters = 2,
                                          kernel_size = (16,16),
                                          strides = (8,8),
                                          padding = 'SAME',
                                          activation = 'softmax')(fcn1 + base_model.layers[10].output)

model = tf.keras.Model(inputs = base_model.inputs, outputs = output)
```

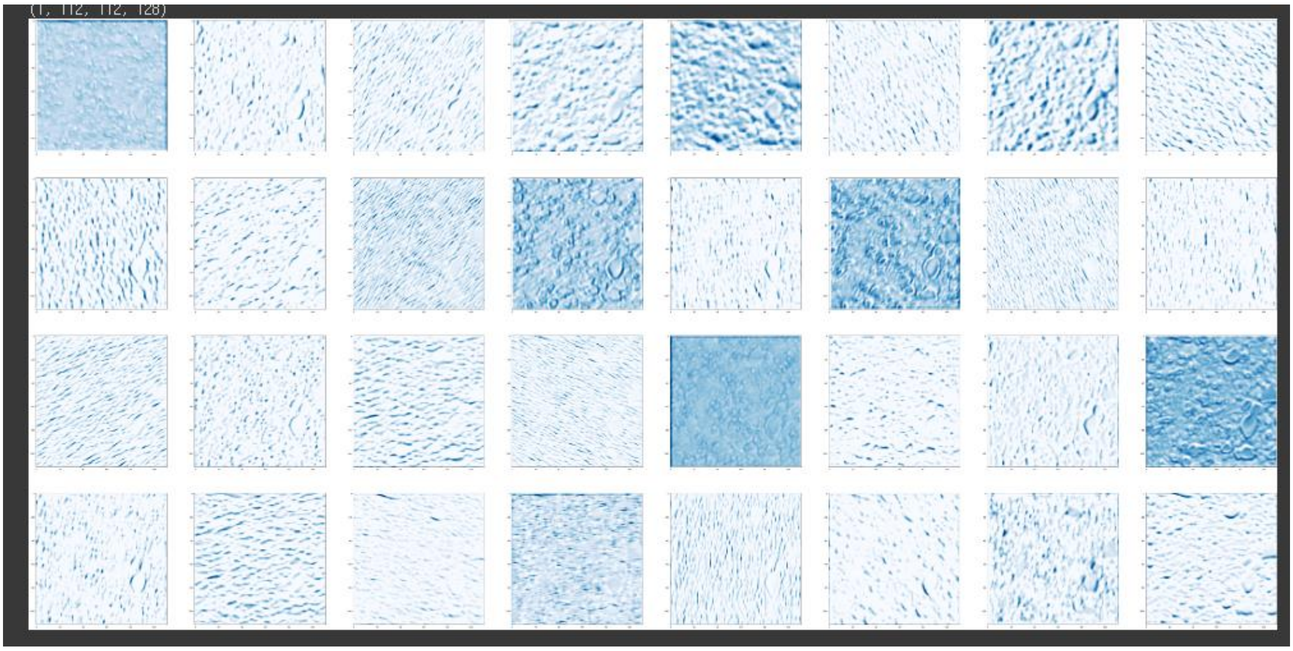


```
Model: "model"
______________________________________________________________________________________________
 Layer (type)                   Output Shape          Param #     Connected to
==============================================================================================
 input_1 (InputLayer)           [(None, 224, 224, 3)]  0          []

 block1_conv1 (Conv2D)          (None, 224, 224, 64)  1792        ['input_1[0][0]']

 block1_conv2 (Conv2D)          (None, 224, 224, 64)  36928       ['block1_conv1[0][0]']

 block1_pool (MaxPooling2D)     (None, 112, 112, 64)  0           ['block1_conv2[0][0]']

 block2_conv1 (Conv2D)          (None, 112, 112, 128)  73856      ['block1_pool[0][0]']

 block2_conv2 (Conv2D)          (None, 112, 112, 128)  147584     ['block2_conv1[0][0]']

 block2_pool (MaxPooling2D)     (None, 56, 56, 128)   0           ['block2_conv2[0][0]']

 block3_conv1 (Conv2D)          (None, 56, 56, 256)   295168      ['block2_pool[0][0]']

 block3_conv2 (Conv2D)          (None, 56, 56, 256)   590080      ['block3_conv1[0][0]']

 block3_conv3 (Conv2D)          (None, 56, 56, 256)   590080      ['block3_conv2[0][0]']

 block3_pool (MaxPooling2D)     (None, 28, 28, 256)   0           ['block3_conv3[0][0]']

 block4_conv1 (Conv2D)          (None, 28, 28, 512)   1180160     ['block3_pool[0][0]']

 block4_conv2 (Conv2D)          (None, 28, 28, 512)   2359808     ['block4_conv1[0][0]']

 block4_conv3 (Conv2D)          (None, 28, 28, 512)   2359808     ['block4_conv2[0][0]']

 block4_pool (MaxPooling2D)     (None, 14, 14, 512)   0           ['block4_conv3[0][0]']

 block5_conv1 (Conv2D)          (None, 14, 14, 512)   2359808     ['block4_pool[0][0]']

 block5_conv2 (Conv2D)          (None, 14, 14, 512)   2359808     ['block5_conv1[0][0]']

 block5_conv3 (Conv2D)          (None, 14, 14, 512)   2359808     ['block5_conv2[0][0]']

 block5_pool (MaxPooling2D)     (None, 7, 7, 512)     0           ['block5_conv3[0][0]']

 conv2d (Conv2D)                (None, 7, 7, 4096)    1027645     ['block5_pool[0][0]']
                                                      44

 conv2d_1 (Conv2D)              (None, 7, 7, 4096)    1678131     ['conv2d[0][0]']
                                                      2

 conv2d_2 (Conv2D)              (None, 7, 7, 2)       8194        ['conv2d_1[0][0]']

 conv2d_transpose (Conv2DTr     (None, 14, 14, 512)   16896       ['conv2d_2[0][0]']
 anspose)

 tf.__operators__.add (TFOp     (None, 14, 14, 512)   0           ['conv2d_transpose[0][0]',
 Lambda)                                                           'block4_pool[0][0]']

 conv2d_transpose_1 (Conv2D     (None, 28, 28, 256)   2097408     ['tf.__operators__.add[0][0]']
 Transpose)

 tf.__operators__.add_1 (TF     (None, 28, 28, 256)   0           ['conv2d_transpose_1[0][0]',
 OpLambda)                                                         'block3_pool[0][0]']

 conv2d_transpose_2 (Conv2D     (None, 224, 224, 2)   131074      ['tf.__operators__.add_1[0][0]
 Transpose)                                                       ']
```

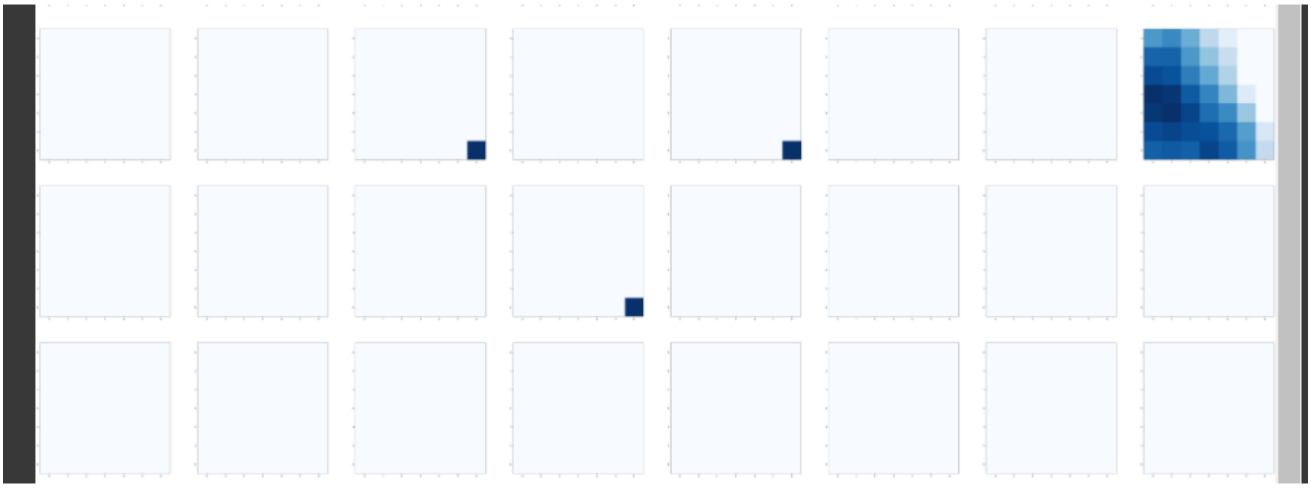
- 데이터 시각화



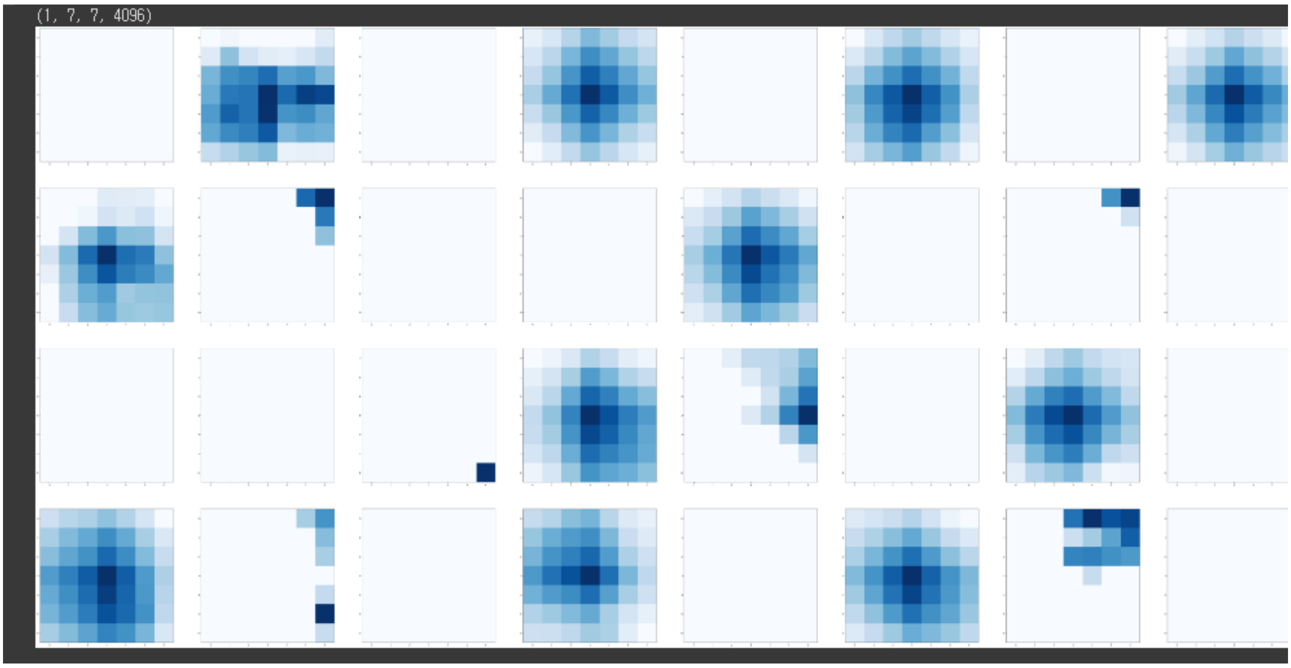
->input



->block2_conv2 (Conv2D) -> (None, 112, 112, 128)

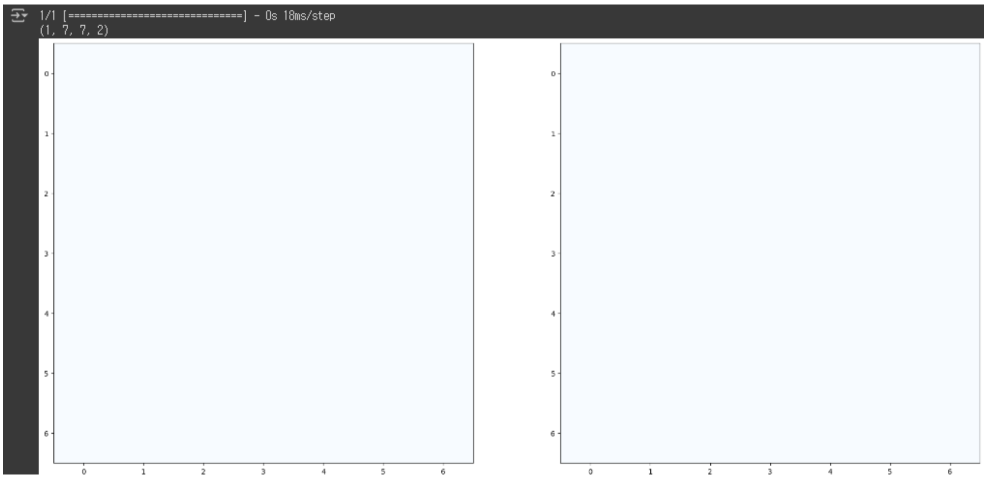


-> conv2d (Conv2D) -> (None, 7, 7, 4096)



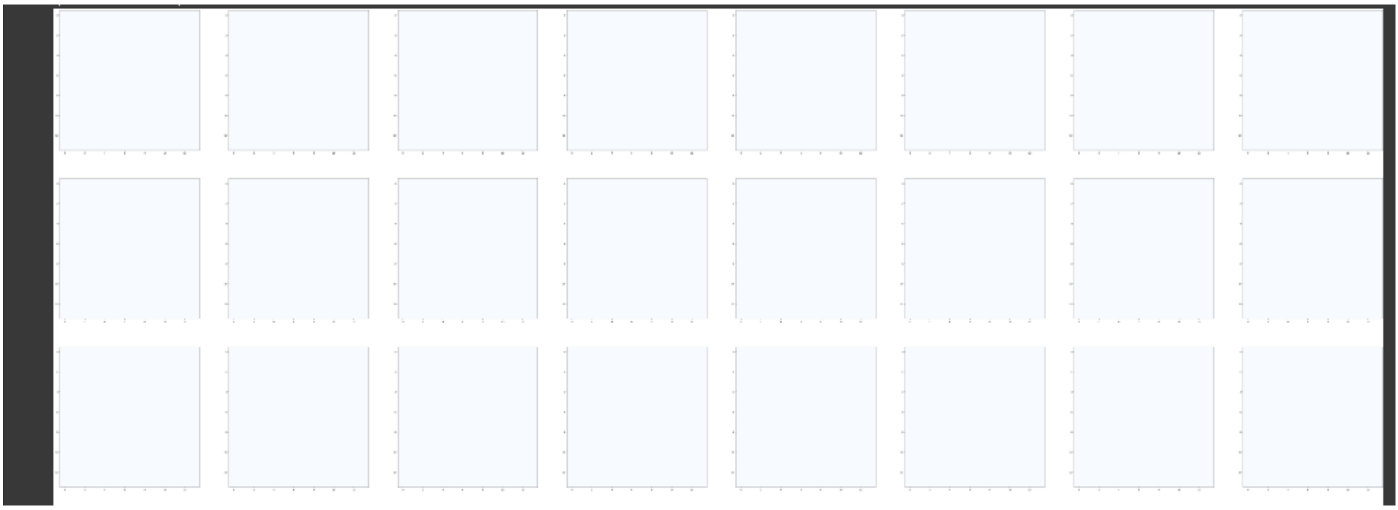
```
(1, 7, 7, 4096)
```

-> conv2d_1 (Conv2D) -> (None, 7, 7, 4096)

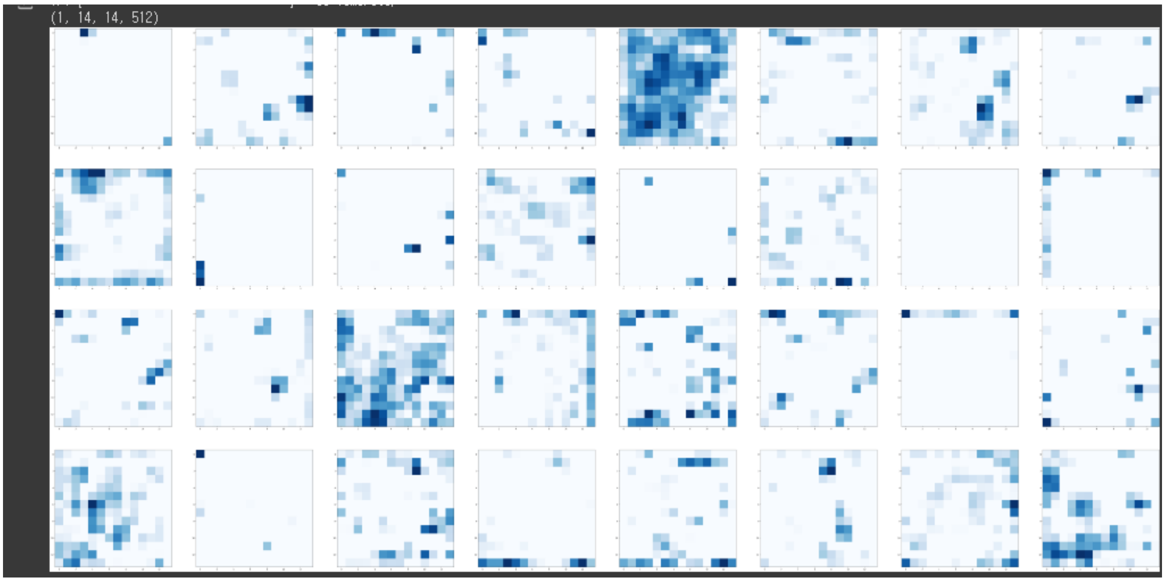


```
1/1 [==============================] - 0s 18ms/step
(1, 7, 7, 2)
```

-> conv2d_2 (Conv2D) -> (None, 7, 7, 2)

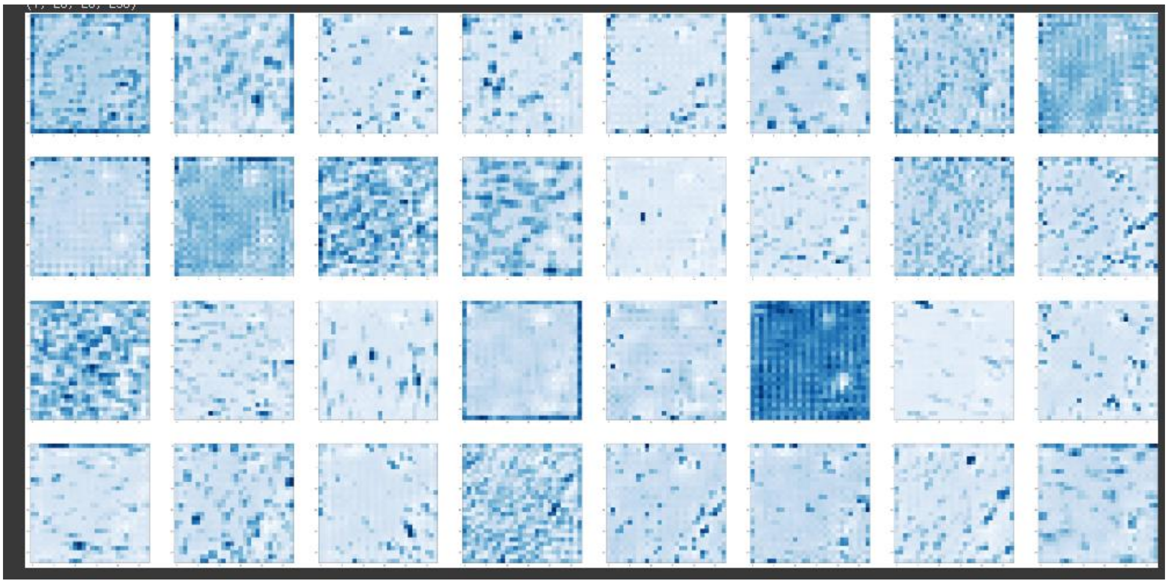


->conv2d_transpose_6 (Conv2D Transpose) ->(None, 14, 14, 512)



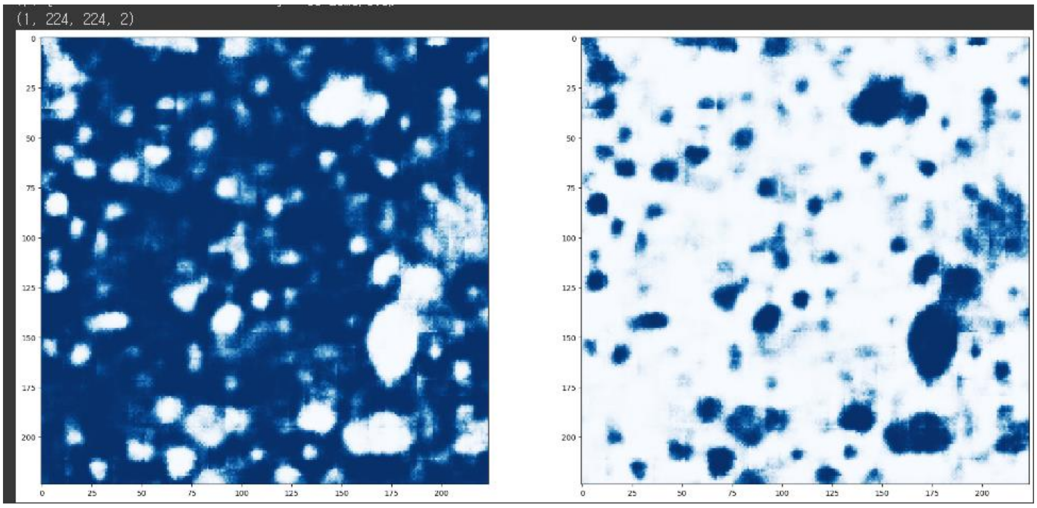
```
(1, 14, 14, 512)
```

->tf.__operators__.add (TFOp Lambda) ->(None, 14, 14, 512) + block4_pool (MaxPooling2D)



```
(1, 28, 28, 256)
```

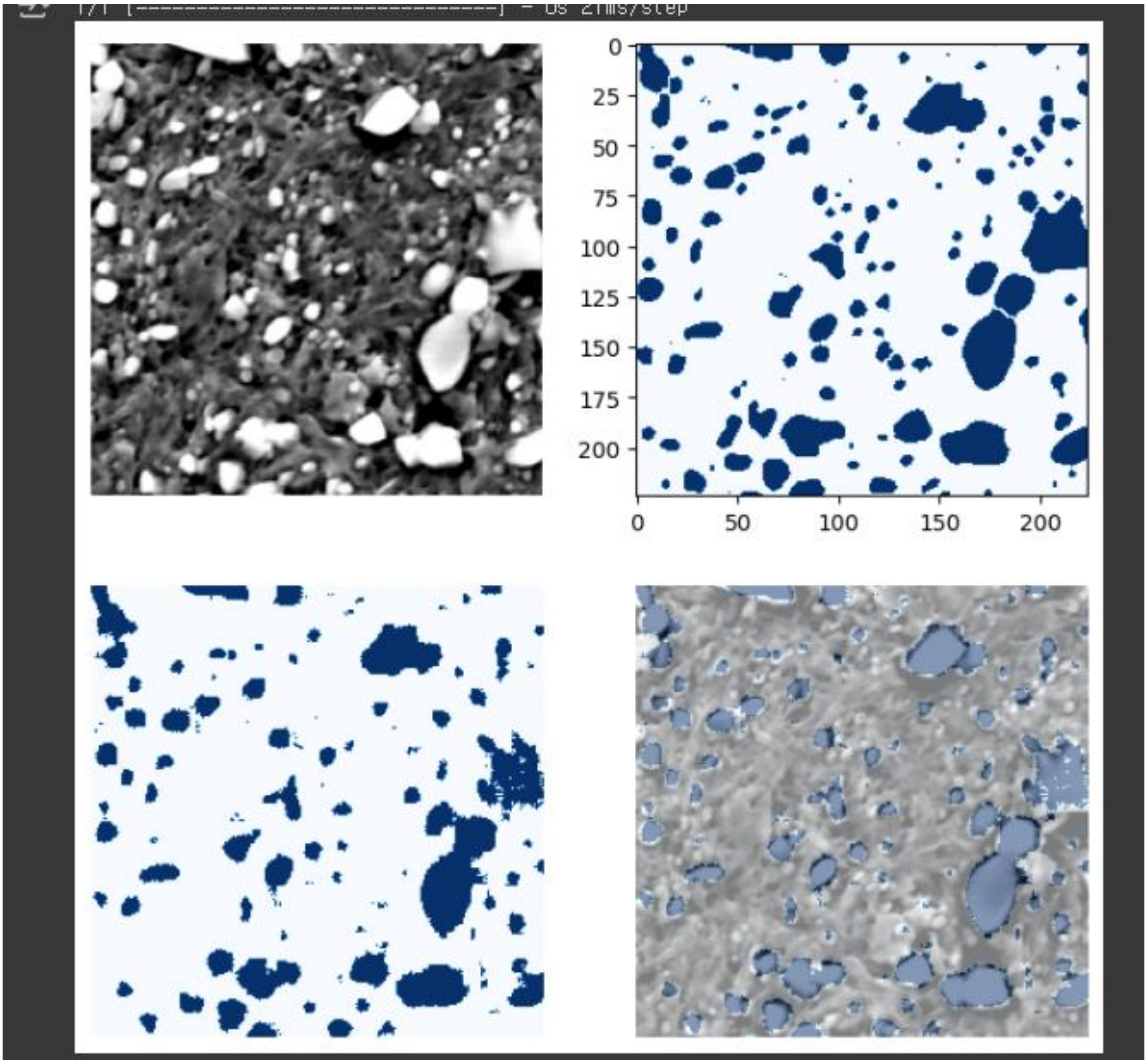
->tf.__operators__.add_1 (TF OpLambda) -> (None, 28, 28, 256)+ block3_pool (MaxPooling2D)



```
(1, 224, 224, 2)
```

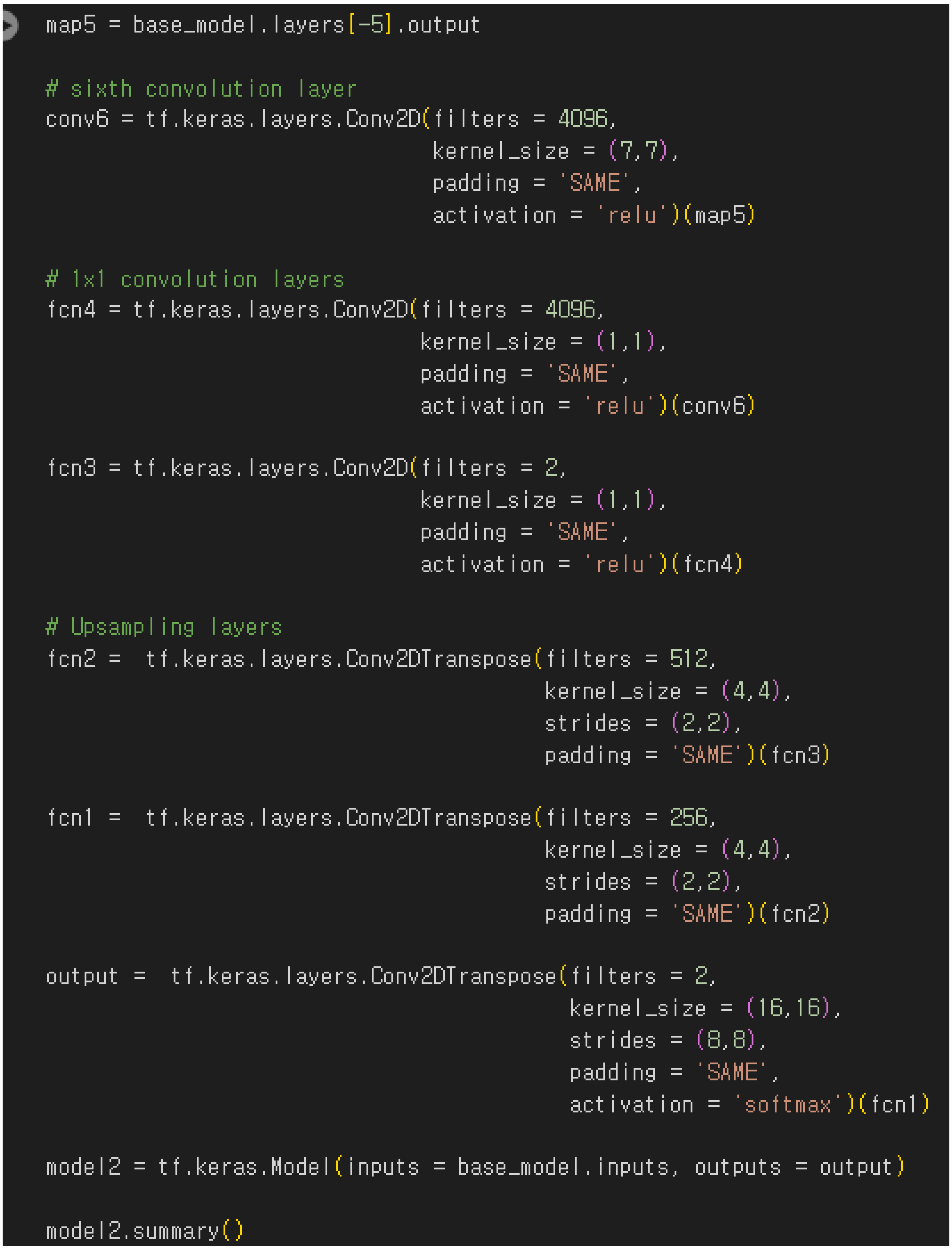
->conv2d_transpose_2 (Conv2D Transpose) ->(None, 224, 224, 2)

- result

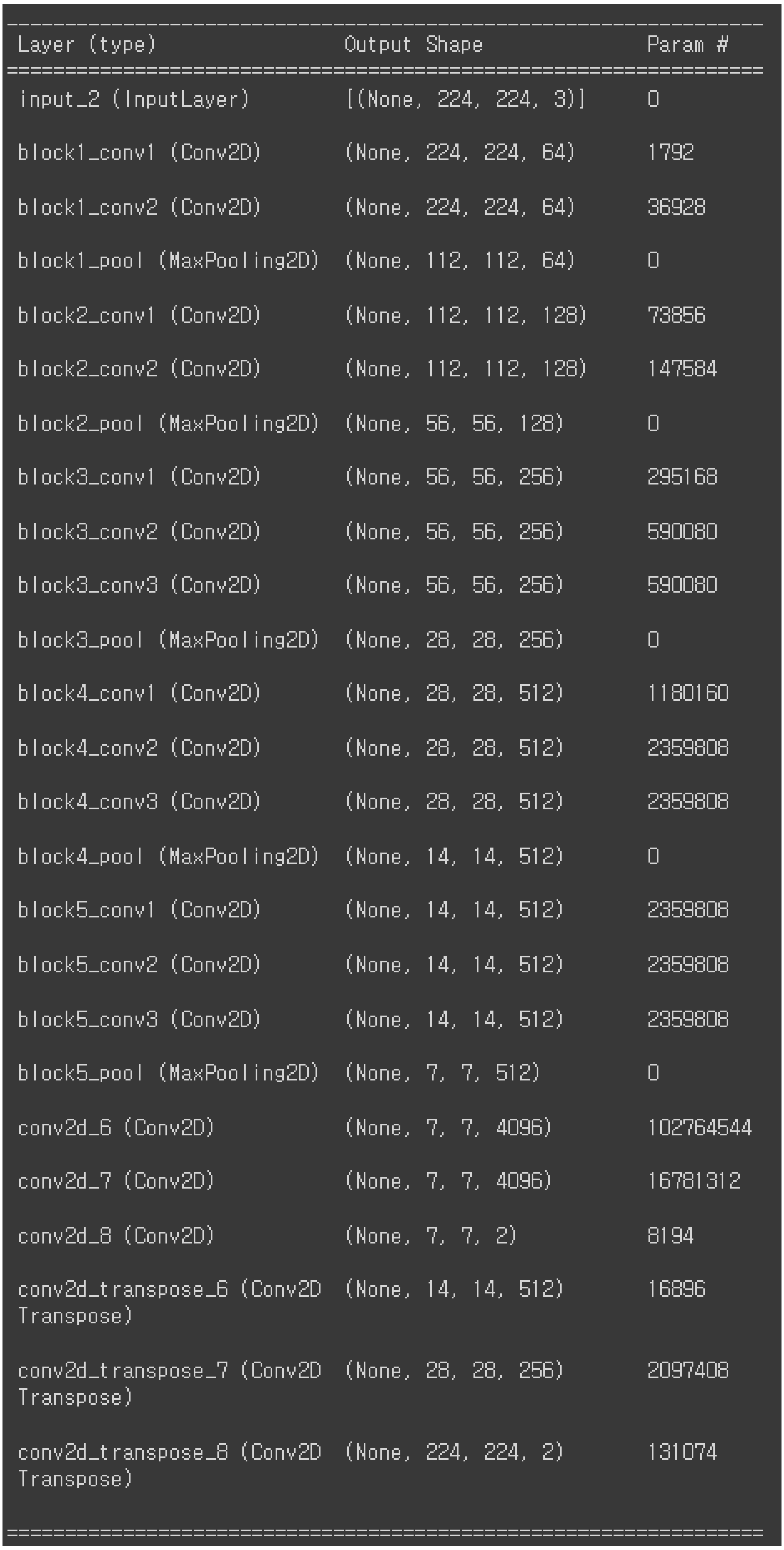


```
1/1 [==============================] - 0s 21ms/step
```

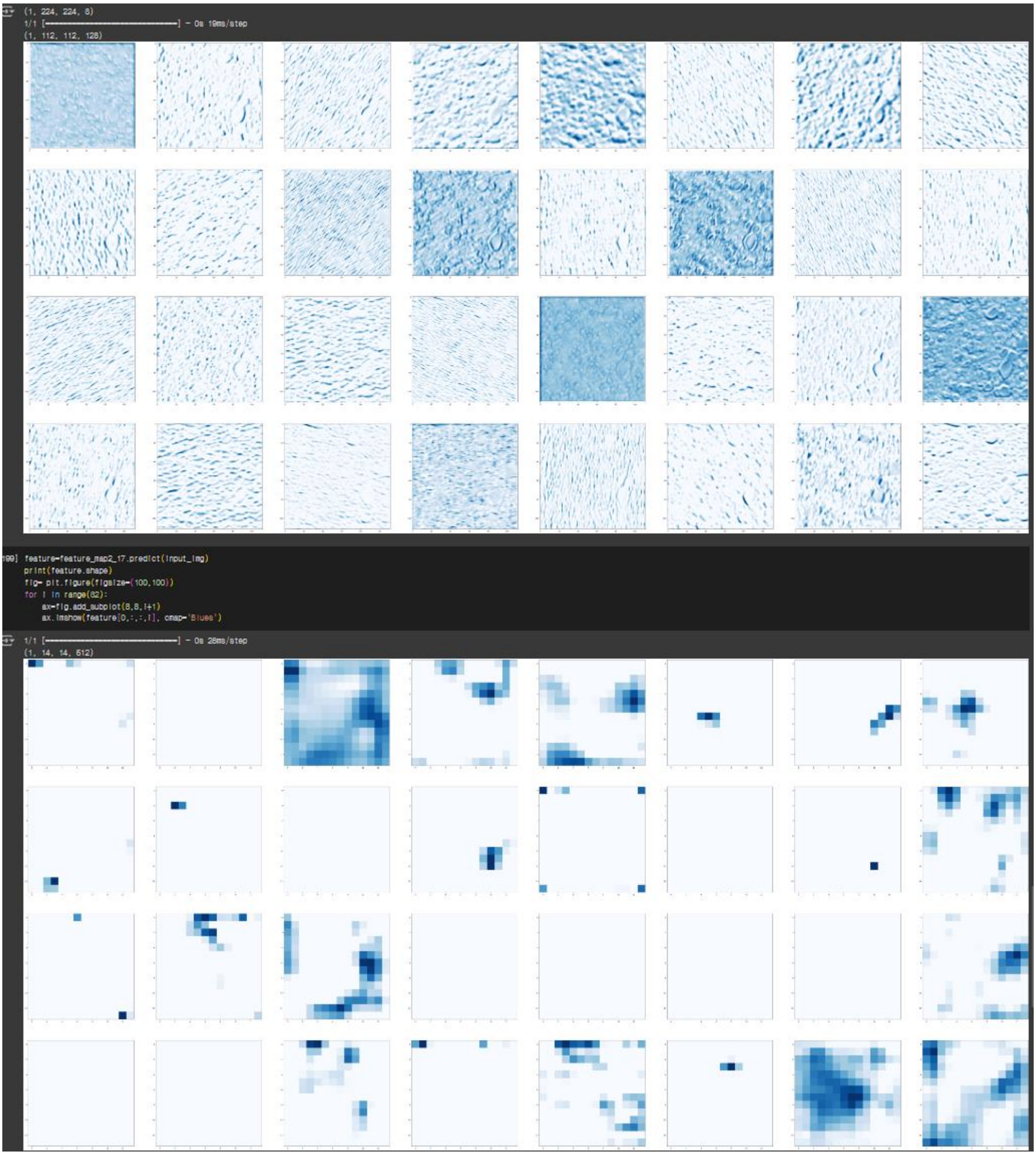
##skip connection 제거?



```
map5 = base_model.layers[-5].output

# sixth convolution layer
conv6 = tf.keras.layers.Conv2D(filters = 4096,
                               kernel_size = (7,7),
                               padding = 'SAME',
                               activation = 'relu')(map5)

# 1x1 convolution layers
fcn4 = tf.keras.layers.Conv2D(filters = 4096,
                              kernel_size = (1,1),
                              padding = 'SAME',
                              activation = 'relu')(conv6)

fcn3 = tf.keras.layers.Conv2D(filters = 2,
                              kernel_size = (1,1),
                              padding = 'SAME',
                              activation = 'relu')(fcn4)

# Upsampling layers
fcn2 =  tf.keras.layers.Conv2DTranspose(filters = 512,
                                        kernel_size = (4,4),
                                        strides = (2,2),
                                        padding = 'SAME')(fcn3)

fcn1 =  tf.keras.layers.Conv2DTranspose(filters = 256,
                                        kernel_size = (4,4),
                                        strides = (2,2),
                                        padding = 'SAME')(fcn2)

output =  tf.keras.layers.Conv2DTranspose(filters = 2,
                                          kernel_size = (16,16),
                                          strides = (8,8),
                                          padding = 'SAME',
                                          activation = 'softmax')(fcn1)

model2 = tf.keras.Model(inputs = base_model.inputs, outputs = output)

model2.summary()
```



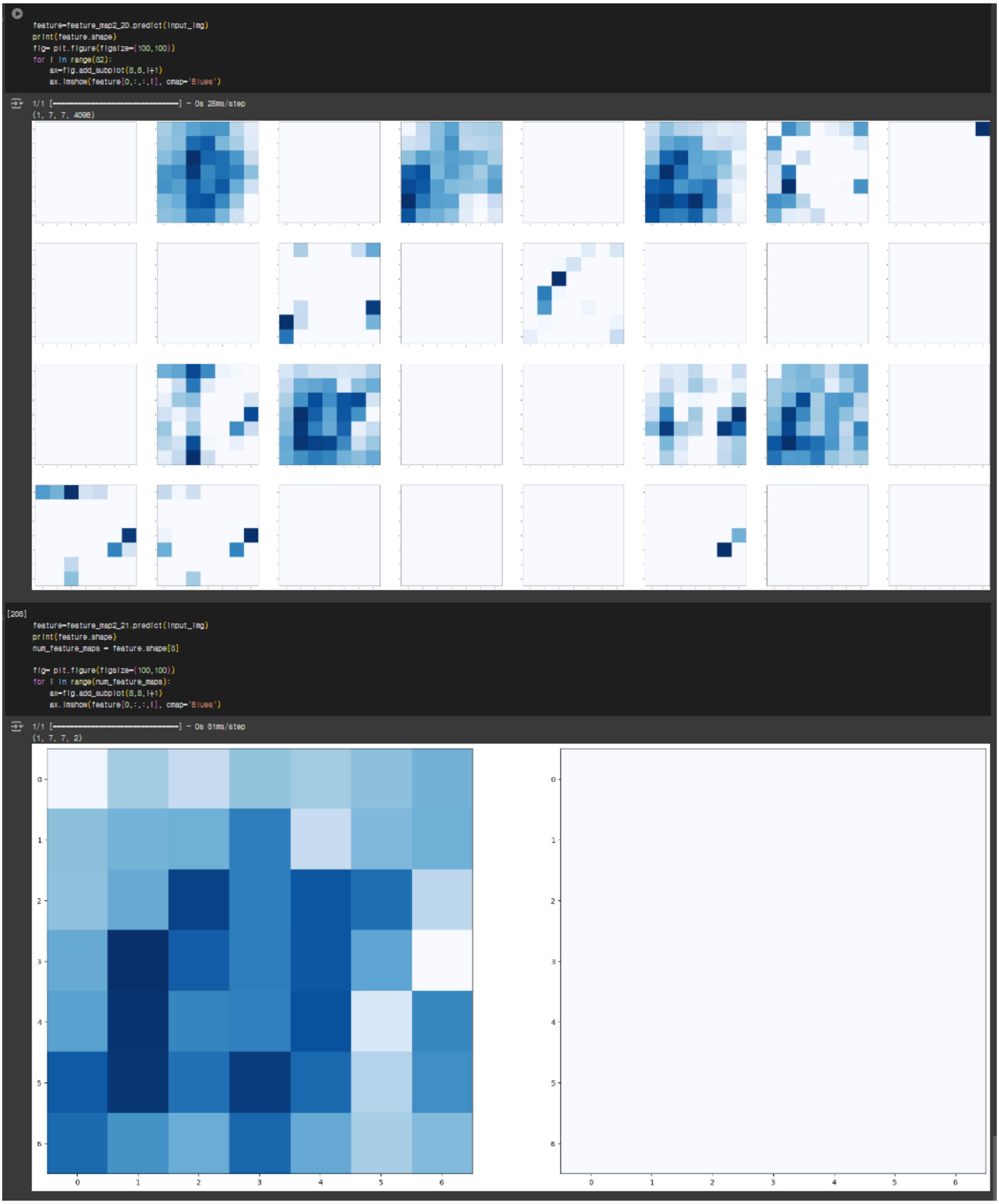
```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_2 (InputLayer)        [(None, 224, 224, 3)]     0

 block1_conv1 (Conv2D)       (None, 224, 224, 64)      1792

 block1_conv2 (Conv2D)       (None, 224, 224, 64)      36928

 block1_pool (MaxPooling2D)  (None, 112, 112, 64)      0

 block2_conv1 (Conv2D)       (None, 112, 112, 128)     73856

 block2_conv2 (Conv2D)       (None, 112, 112, 128)     147584

 block2_pool (MaxPooling2D)  (None, 56, 56, 128)       0

 block3_conv1 (Conv2D)       (None, 56, 56, 256)       295168

 block3_conv2 (Conv2D)       (None, 56, 56, 256)       590080

 block3_conv3 (Conv2D)       (None, 56, 56, 256)       590080

 block3_pool (MaxPooling2D)  (None, 28, 28, 256)       0

 block4_conv1 (Conv2D)       (None, 28, 28, 512)       1180160

 block4_conv2 (Conv2D)       (None, 28, 28, 512)       2359808

 block4_conv3 (Conv2D)       (None, 28, 28, 512)       2359808

 block4_pool (MaxPooling2D)  (None, 14, 14, 512)       0

 block5_conv1 (Conv2D)       (None, 14, 14, 512)       2359808

 block5_conv2 (Conv2D)       (None, 14, 14, 512)       2359808

 block5_conv3 (Conv2D)       (None, 14, 14, 512)       2359808

 block5_pool (MaxPooling2D)  (None, 7, 7, 512)         0

 conv2d_6 (Conv2D)           (None, 7, 7, 4096)        102764544

 conv2d_7 (Conv2D)           (None, 7, 7, 4096)        16781312

 conv2d_8 (Conv2D)           (None, 7, 7, 2)           8194

 conv2d_transpose_6 (Conv2D  (None, 14, 14, 512)       16896
 Transpose)

 conv2d_transpose_7 (Conv2D  (None, 28, 28, 256)       2097408
 Transpose)

 conv2d_transpose_8 (Conv2D  (None, 224, 224, 2)       131074
 Transpose)

=================================================================
```



```
(1, 224, 224, 8)
1/1 [==============================] - 0s 19ms/step
(1, 112, 112, 128)
```

```
199] feature=feature_map2_17.predict(input_img)
     print(feature.shape)
     fig= plt.figure(figsize=(100,100))
     for i in range(32):
         ax=fig.add_subplot(8,8,i+1)
         ax.imshow(feature[0,:,:,i], cmap='Blues')
```

```
1/1 [==============================] - 0s 28ms/step
(1, 14, 14, 512)
```



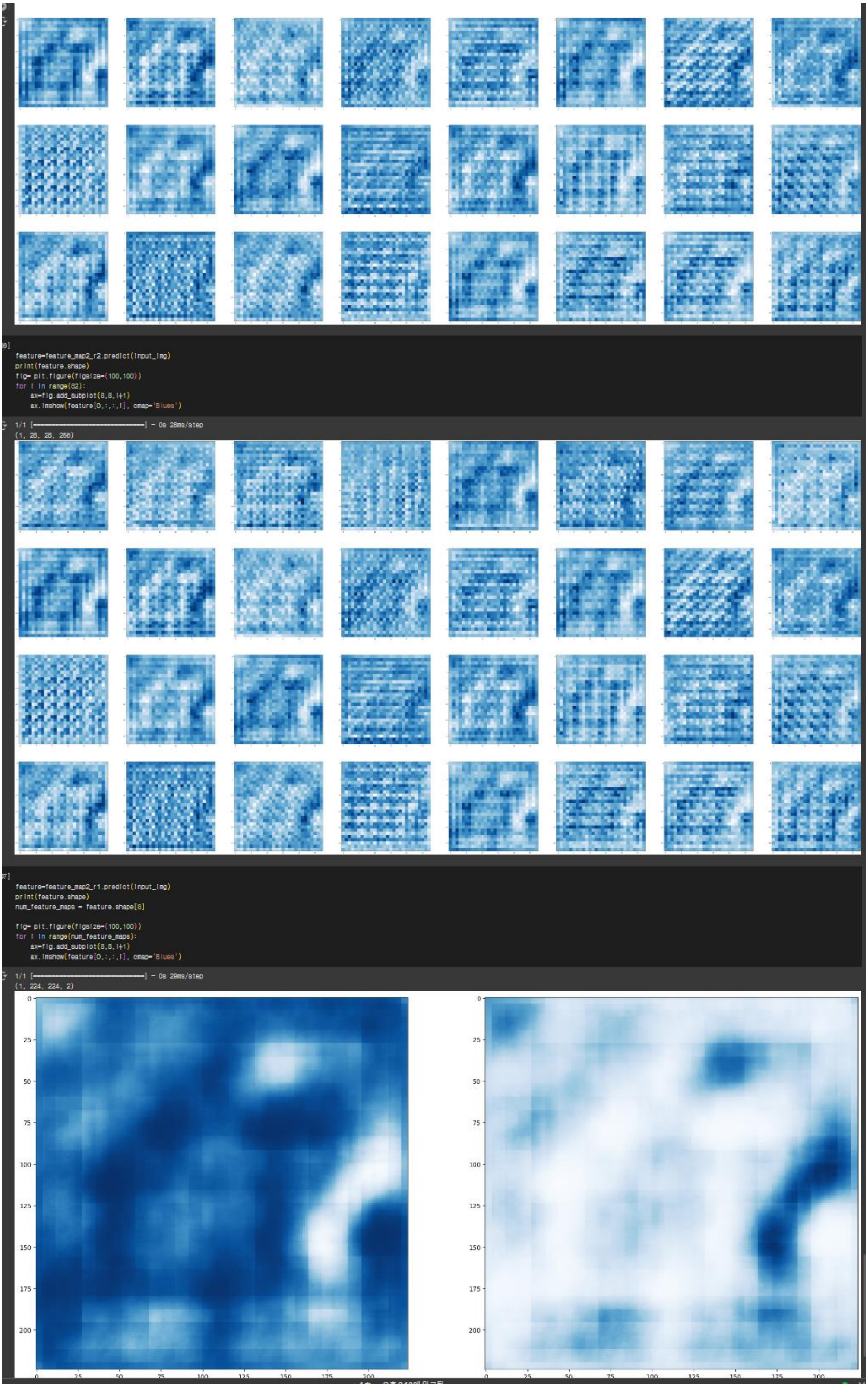
```
feature=feature_map2_20.predict(input_img)
print(feature.shape)
fig= plt.figure(figsize=(100,100))
for i in range(32):
    ax=fig.add_subplot(8,8,i+1)
    ax.imshow(feature[0,:,:,i], cmap='Blues')
```

```
1/1 [==============================] - 0s 25ms/step
(1, 7, 7, 4096)
```

```
[206]
feature=feature_map2_21.predict(input_img)
print(feature.shape)
num_feature_maps = feature.shape[3]

fig= plt.figure(figsize=(100,100))
for i in range(num_feature_maps):
    ax=fig.add_subplot(8,8,i+1)
    ax.imshow(feature[0,:,:,i], cmap='Blues')
```

```
1/1 [==============================] - 0s 61ms/step
(1, 7, 7, 2)
```

->encoder 부분



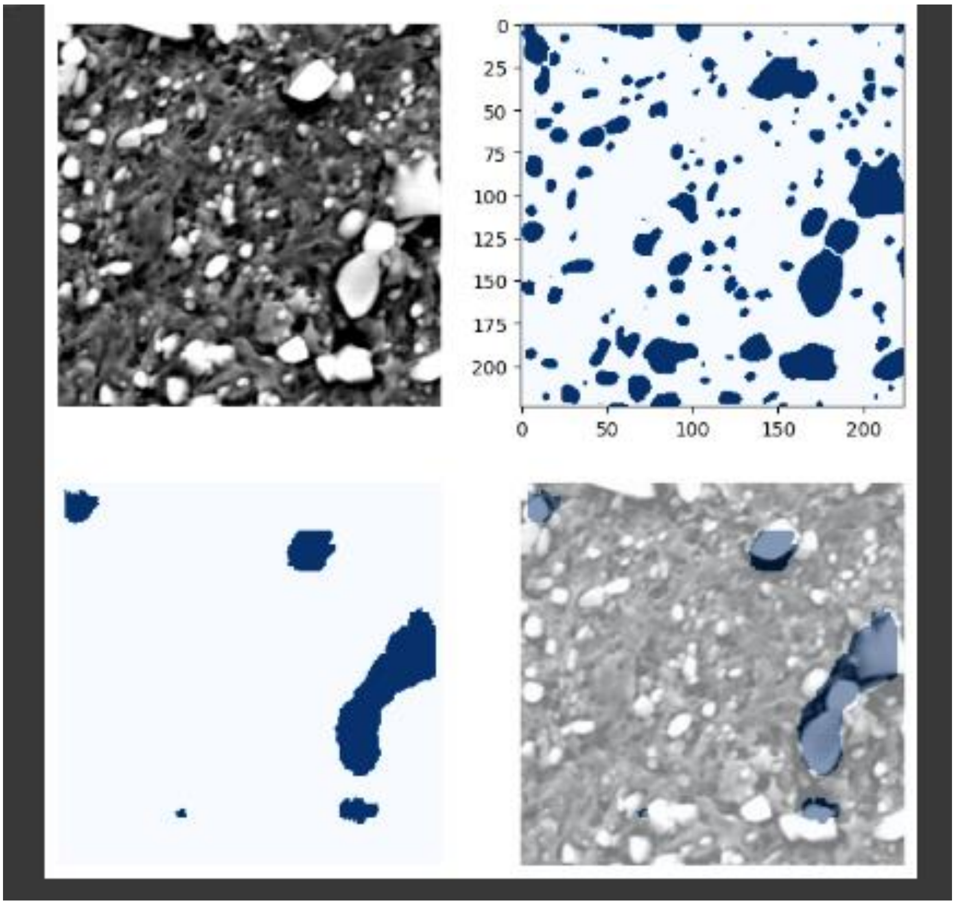
```
8]
feature=feature_map2_r2.predict(input_img)
print(feature.shape)
fig= plt.figure(figsize=(100,100))
for i in range(32):
    ax=fig.add_subplot(8,8,i+1)
    ax.imshow(feature[0,:,:,i], cmap='Blues')
```

```
1/1 [==============================] - 0s 28ms/step
(1, 28, 28, 256)
```

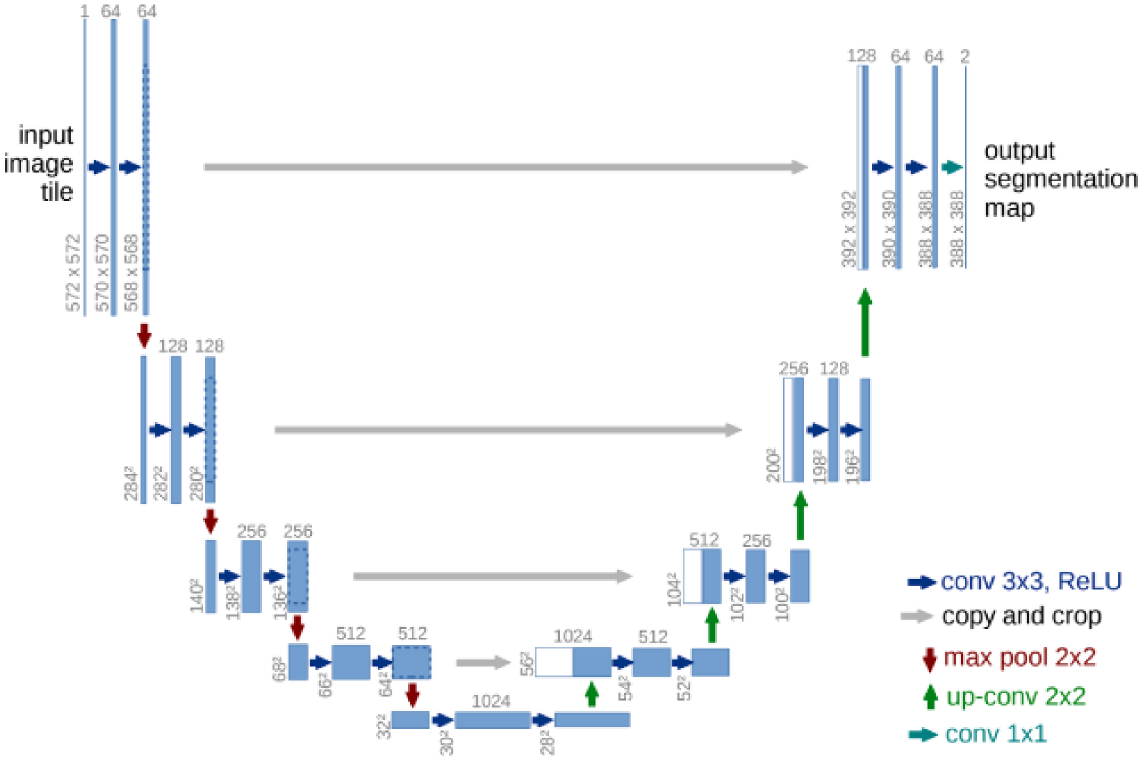
```
7]
feature=feature_map2_r1.predict(input_img)
print(feature.shape)
num_feature_maps = feature.shape[3]

fig= plt.figure(figsize=(100,100))
for i in range(num_feature_maps):
    ax=fig.add_subplot(8,8,i+1)
    ax.imshow(feature[0,:,:,i], cmap='Blues')
```

```
1/1 [==============================] - 0s 29ms/step
(1, 224, 224, 2)
```

- skip connection제거 result



#U-NET


1 64 64
input
image
tile
572 x 572
570 x 570
568 x 568
128 128
284²
282²
280²
256 256
140²
138²
136²
512 512
68²
66²
64²
1024
32²
30²
28²
1024 512
56²
54²
52²
512 256
104²
102²
100²
256 128
200²
198²
196²
128 64 64 2
392 x 392
390 x 390
388 x 388
388 x 388
output
segmentation
map
conv 3x3, ReLU
copy and crop
max pool 2x2
up-conv 2x2
conv 1x1