demo_gradcam

October 27, 2019

1 Applying Grad-CAM to a sample image

1.1 Imports

In [1]:
```
import math
from keras.models import load_model
from keras.preprocessing.image import ImageDataGenerator
import matplotlib.pyplot as plt

from gradcam_utils import *
```

```
Using TensorFlow backend.
```

1.2 Load data

In [2]:
```
DATASET_ROOT = '../mvtec_texture_anomaly_detection/test'
GT_ROOT = '../mvtec_texture_anomaly_detection/ground_truth'
CLASSES = ['carpet', 'grid', 'leather', 'tile', 'wood']

test_datagen = ImageDataGenerator(rescale=1./255)
test_generator = test_datagen.flow_from_directory(DATASET_ROOT,
                                                  class_mode='categorical',
                                                  interpolation='bilinear',
                                                  target_size=(224, 224),
                                                  batch_size=16,
                                                  shuffle=False,
                                                  classes=CLASSES)
unique, counts = np.unique(test_generator.labels, return_counts=True)
print(dict(zip(CLASSES, counts)))
```

```
Found 515 images belonging to 5 classes.
{'carpet': 117, 'grid': 78, 'leather': 124, 'tile': 117, 'wood': 79}
```

1.3 Load model

```
In [3]: model = load_model('save/texturenet.h5')
        for l in model.layers:
            print(l.name)
```

```
conv2d_1
conv2d_2
max_pooling2d_1
conv2d_3
conv2d_4
max_pooling2d_2
conv2d_5
conv2d_6
conv2d_7
global_average_pooling2d_1
dense_1
```

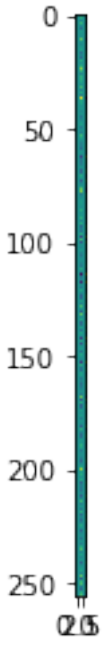
1.4 Get Grad-CAM weights

```
In [4]: alpha, layer = get_grad_cam_weights(model, np.zeros((1, 224, 224, 3)))
```

```
In [5]: print(alpha.shape)
        plt.imshow(alpha)
```

```
(256, 5)
```

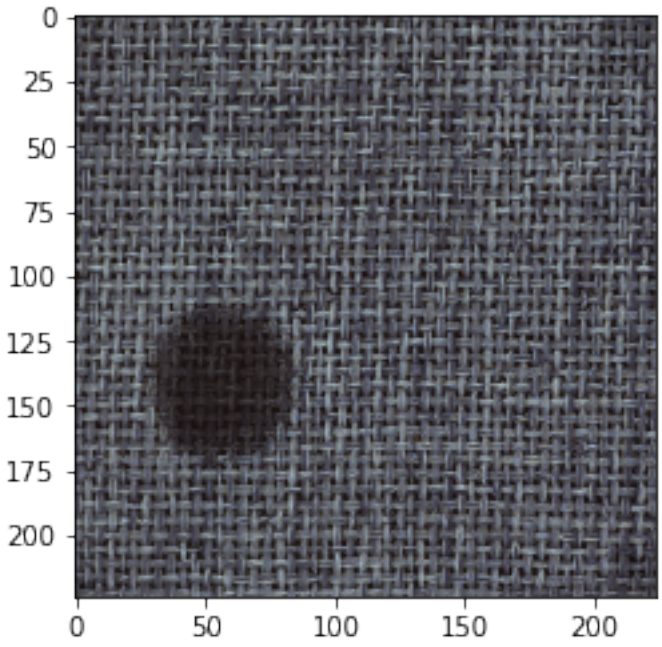
```
Out[5]: <matplotlib.image.AxesImage at 0x28a13691d68>
```



```
In [6]: print(layer)

conv2d_7
```

1.5 Load image batch

```
In [7]: start_idx = 0
        end_idx = 15
        cur_batch_sz = end_idx - start_idx + 1
        img_batch = read_batch(test_generator.directory, test_generator.filenames[start_idx:end
                               (224, 224))

In [8]: plt.imshow(img_batch[0])

Out[8]: <matplotlib.image.AxesImage at 0x28a122612b0>
```




```
In [9]: print(test_generator.filenames[0])

carpet\carpet_test_color_000.png
```

1.6 Predict texture class

```
In [10]: pred_scores = model.predict(img_batch / 255)
         is_pass_threshold = np.zeros((cur_batch_sz, len(CLASSES)))
         is_pass_threshold[np.arange(cur_batch_sz), np.argmax(pred_scores, axis=1)] = 1
         print(is_pass_threshold)
```

```
[[1. 0. 0. 0. 0.]
 [1. 0. 0. 0. 0.]
 [1. 0. 0. 0. 0.]
 [1. 0. 0. 0. 0.]
 [1. 0. 0. 0. 0.]
 [1. 0. 0. 0. 0.]
 [1. 0. 0. 0. 0.]
 [1. 0. 0. 0. 0.]
 [1. 0. 0. 0. 0.]
 [1. 0. 0. 0. 0.]
 [1. 0. 0. 0. 0.]
 [1. 0. 0. 0. 0.]
 [1. 0. 0. 0. 0.]
 [1. 0. 0. 0. 0.]
 [1. 0. 0. 0. 0.]]
```

1.7 Run Grad-CAM

Ignoring non-maximum classes:

```
In [11]: Y = grad_cam(model, alpha, img_batch / 255, is_pass_threshold, layer, pred_scores,
                      orig_sz=img_batch.shape[1:3], should_upsample=True)
```

```
In [12]: plt.figure()
         for i in range(len(CLASSES)):
             plt.subplot(151+i)
             plt.imshow(Y[0, :, :, i])
             plt.title(CLASSES[i])
             plt.xticks([])
             plt.yticks([])
```

Including non-maximum classes:

In [13]:
```
Y = grad_cam(model, alpha, img_batch / 255, np.ones((cur_batch_sz, len(CLASSES))), lay
                    orig_sz=img_batch.shape[1:3], should_upsample=True)
```

In [14]:
```
plt.figure()
for i in range(len(CLASSES)):
    plt.subplot(151+i)
    plt.imshow(Y[0, :, :, i])
    plt.title(CLASSES[i])
    plt.xticks([])
    plt.yticks([])
```