Assignment 2 : Grad-CAM

03137910 Nelson Lefebvre

I choosed to implement the Grad-CAM explainable machine learning methods that we had discussed in class:

In [11]:
```
import keras
import tensorflow as tf
tf.compat.v1.disable_eager_execution()
import vis
import matplotlib.pyplot as plt
%matplotlib inline
import numpy as np
import cv2
from keras.preprocessing import image
from keras import backend as K
from keras.applications.vgg16 import preprocess_input, decode_predictions,VGG16, p
from vis.utils import utils
from tensorflow.keras.preprocessing.image import load_img

model = VGG16(weights='imagenet')
img = load_img('Panda.jpg',target_size=(224,224))
x = tf.keras.utils.img_to_array(img)
x = np.expand_dims(x, axis=0)
x = preprocess_input(x)
```

In [12]:
```
print('The most accurate possibility is :', tf.keras.applications.vgg16.decode_pre
```

```
The most accurate possibility is : [('n02510455', 'giant_panda', 0.9979412)]
```

In [13]:
```
layer_names = ['block1_conv1', 'block1_conv2', 'block2_conv1', 'block2_conv2', 'bl
pred_vector_output = model.output[:, 386]

heatmap = []
for i in layer_names:
    layer = model.get_layer(i)
    gradient = tf.gradients(pred_vector_output, layer.output)[0]
    gradient_boosted = K.mean(gradient, axis=(0, 1, 2))
    iterate = K.function([model.input], [gradient_boosted, layer.output[0]])
    pooled_grads, conv_layer_output = iterate([x])
    for i in range(model.get_layer(i).output_shape[-1]):
        conv_layer_output[:, :, i] *= pooled_grads[i]
    heatmap.append(np.mean(conv_layer_output, axis=-1))
```

In [14]:
```
#Bonus i printed the differents layers with the heatmap
plt.figure(figsize=(18, 18))
for i in range(13):
    plt.figure(figsize=(40,40))
    plt.subplot(5, 3, i+1)
    img_heatmap = np.maximum(heatmap[i], 0)
    img_heatmap /= np.max(img_heatmap)
    plt.imshow(img_heatmap)
    plt.title(layer_names[i])
    plt.show()
```

```
<Figure size 1296x1296 with 0 Axes>
```

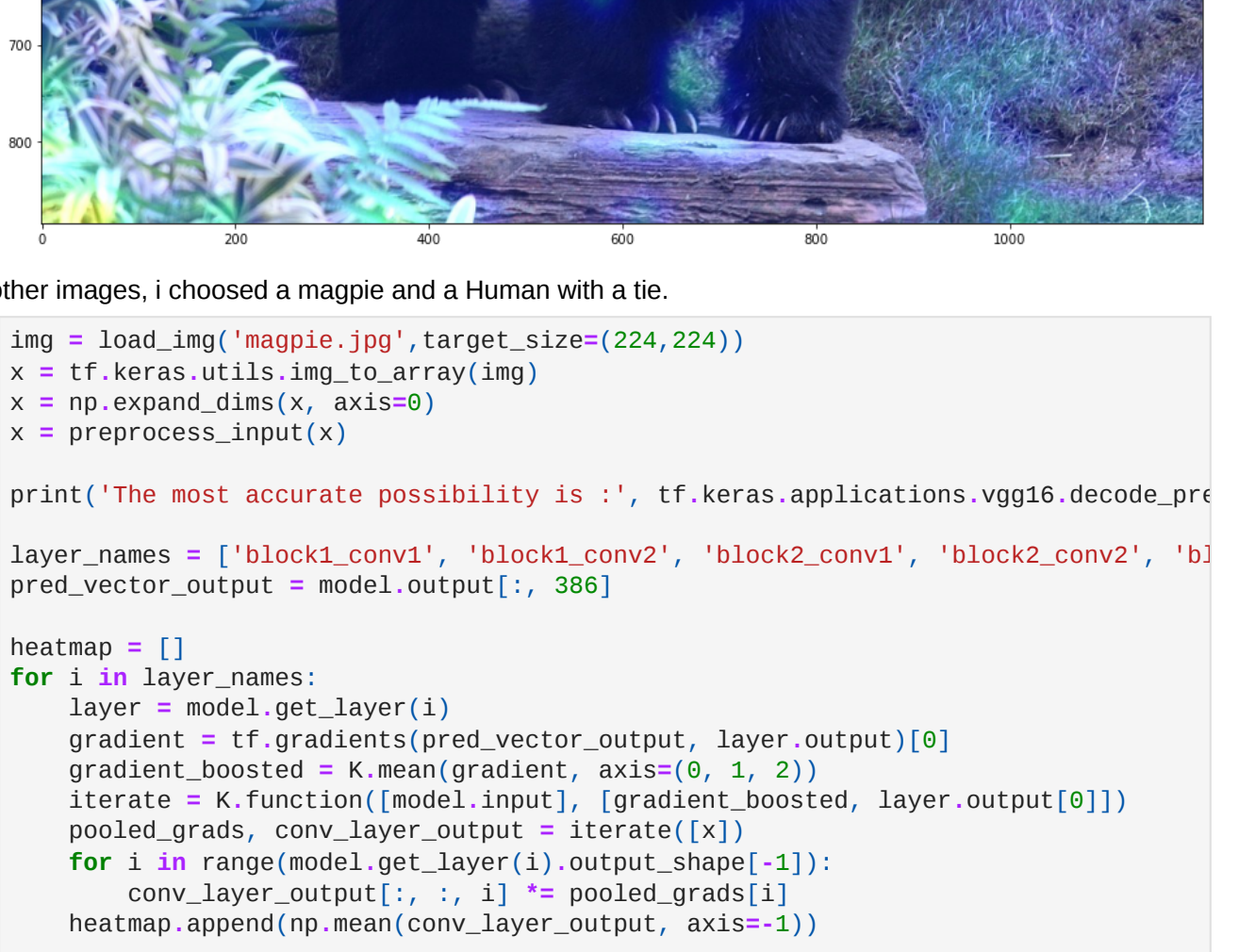
In [15]:
```
img = cv2.imread('Panda.jpg')
for i, hm in enumerate(heatmap):
    img_heatmap = np.maximum(hm, 0)
    img_heatmap /= np.max(img_heatmap)
    img_hm = cv2.resize(img_heatmap, (img.shape[1], img.shape[0]))
    img_hm = np.uint8(255 * img_hm)
    img_hm = cv2.applyColorMap(img_hm, cv2.COLORMAP_JET)
    superposed = img_hm * 0.4 + img
    cv2.imwrite('./Panda_{}.jpg'.format(layer_names[i]), superposed)
```

In [16]:
```
plt.figure(figsize=(16, 16))
img = tf.keras.utils.img_to_array(tf.keras.utils.load_img('./Panda.jpg')) / 255
plt.imshow(img)
plt.title('input_image')
```

Out[16]:
```
Text(0.5, 1.0, 'input_image')
```

In [17]:
```
plt.figure(figsize=(16, 16))
layer_name_exemple = 'block4_conv3'
img = tf.keras.utils.img_to_array(
    tf.keras.utils.load_img('./Panda_{}.jpg'.format(layer_name_exemple))) / 255
plt.imshow(img)
plt.title(layer_name_exemple)
```

Out[17]:
```
Text(0.5, 1.0, 'block4_conv3')
```

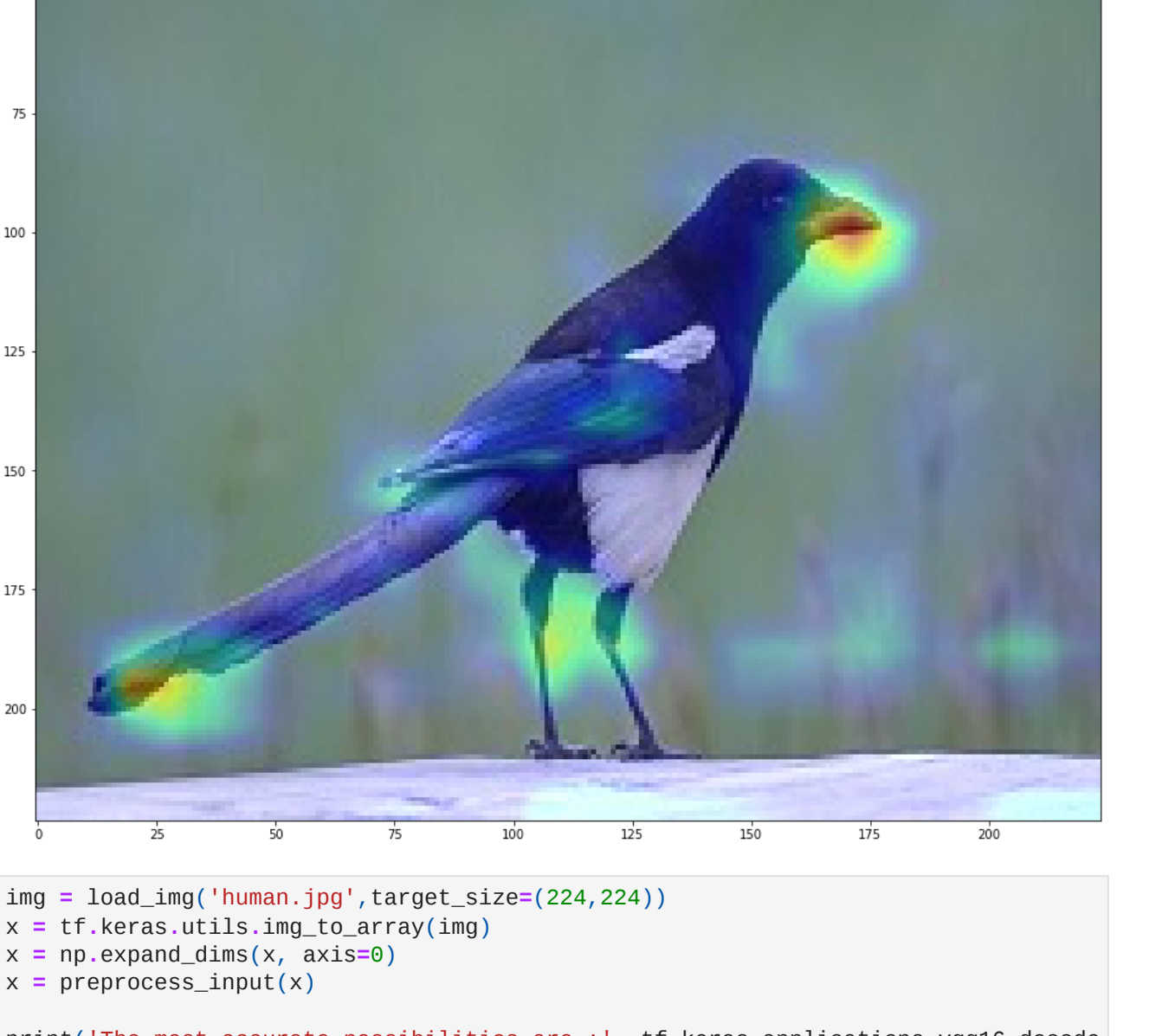


Let's print 2 other images, i choosed a magpie and a Human with a tie.

In [18]:
```
img = load_img('magpie.jpg',target_size=(224,224))
x = tf.keras.utils.img_to_array(img)
x = np.expand_dims(x, axis=0)
x = preprocess_input(x)

print('The most accurate possibility is :', tf.keras.applications.vgg16.decode_pre

layer_names = ['block1_conv1', 'block1_conv2', 'block2_conv1', 'block2_conv2', 'bl
pred_vector_output = model.output[:, 386]

heatmap = []
for i in layer_names:
    layer = model.get_layer(i)
    gradient = tf.gradients(pred_vector_output, layer.output)[0]
    gradient_boosted = K.mean(gradient, axis=(0, 1, 2))
    iterate = K.function([model.input], [gradient_boosted, layer.output[0]])
    pooled_grads, conv_layer_output = iterate([x])
    for i in range(model.get_layer(i).output_shape[-1]):
        conv_layer_output[:, :, i] *= pooled_grads[i]
    heatmap.append(np.mean(conv_layer_output, axis=-1))

img = cv2.imread('magpie.jpg')
for i, hm in enumerate(heatmap):
    img_heatmap = np.maximum(hm, 0)
    img_heatmap /= np.max(img_heatmap)
    img_hm = cv2.resize(img_heatmap, (img.shape[1], img.shape[0]))
    img_hm = np.uint8(255 * img_hm)
    img_hm = cv2.applyColorMap(img_hm, cv2.COLORMAP_JET)
    superposed = img_hm * 0.4 + img
    cv2.imwrite('./magpie_{}.jpg'.format(layer_names[i]), superposed)

plt.figure(figsize=(16, 16))
layer_name = 'input_image'
img = tf.keras.utils.img_to_array(tf.keras.utils.load_img('./magpie.jpg')) / 255
plt.imshow(img)
plt.title(layer_name)
plt.figure(figsize=(16, 16))
layer_name_exemple = 'block4_conv3'
img = tf.keras.utils.img_to_array(
    tf.keras.utils.load_img('./magpie_{}.jpg'.format(layer_name_exemple))) / 255
plt.imshow(img)
plt.title(layer_name_exemple)
```

```
The most accurate possibility is : [('n01582220', 'magpie', 0.9977563)]
```

Out[18]:
```
Text(0.5, 1.0, 'block4_conv3')
```



In [19]:
```
img = load_img('human.jpg',target_size=(224,224))
x = tf.keras.utils.img_to_array(img)
x = np.expand_dims(x, axis=0)
x = preprocess_input(x)

print('The most accurate possibilities are :', tf.keras.applications.vgg16.decode_

layer_names = ['block1_conv1', 'block1_conv2', 'block2_conv1', 'block2_conv2', 'bl
pred_vector_output = model.output[:, 386]

heatmap = []
for i in layer_names:
    layer = model.get_layer(i)
    gradient = tf.gradients(pred_vector_output, layer.output)[0]
    gradient_boosted = K.mean(gradient, axis=(0, 1, 2))
    iterate = K.function([model.input], [gradient_boosted, layer.output[0]])
    pooled_grads, conv_layer_output = iterate([x])
    for i in range(model.get_layer(i).output_shape[-1]):
        conv_layer_output[:, :, i] *= pooled_grads[i]
    heatmap.append(np.mean(conv_layer_output, axis=-1))

img = cv2.imread('human.jpg')
for i, hm in enumerate(heatmap):
    img_heatmap = np.maximum(hm, 0)
    img_heatmap /= np.max(img_heatmap)
    img_hm = cv2.resize(img_heatmap, (img.shape[1], img.shape[0]))
    img_hm = np.uint8(255 * img_hm)
    img_hm = cv2.applyColorMap(img_hm, cv2.COLORMAP_JET)
    superposed = img_hm * 0.4 + img
    cv2.imwrite('./human_{}.jpg'.format(layer_names[i]), superposed)

plt.figure(figsize=(16, 16))
layer_name = 'input_image'
img = tf.keras.utils.img_to_array(tf.keras.utils.load_img('./human.jpg')) / 255
plt.imshow(img)
plt.title(layer_name)
plt.figure(figsize=(16, 16))
layer_name_exemple = 'block2_conv2'
img = tf.keras.utils.img_to_array(
    tf.keras.utils.load_img('./human_{}.jpg'.format(layer_name_exemple))) / 255
plt.imshow(img)
plt.title(layer_name_exemple)
```

```
The most accurate possibilities are : [('n02883205', 'bow_tie', 0.99200773), ('n0
4591157', 'Windsor_tie', 0.0046322476), ('n04350905', 'suit', 0.0018812838)]
```

Out[19]:
```
Text(0.5, 1.0, 'block2_conv2')
```

Discussion of thinking : why does a model think an image is a "y"? The GradCam (a technique from a more general class of processes called class activation map visualization CAM) is a visualization technique as we have observed that allows to understand which parts of a given image led a convnet to its final classification decision. It is useful for debugging the decision process of a convnet, especially in the case of a classification error but also allows to locate specific objects in an image. The principle is to generate a heatmap for the class "X" (parrot, dog, bow tie ...), indicating how much the different parts of the image look like an X. discovery : https://www.youtube.com/watch?v=RNnKtNrsrmg&ab_channel=DenisDmitriev https://keras.io/examples/vision/grad_cam/ discovery in french : https://inside-machinelearning.com/keras-gradcam-tutoriel/