PW-11

by Aurélien Héritier and Jean Nanchen

Exercice 1

Understanding Convolutional neural networks by using filter activation statistics

The objective of this exercise is to train a deep convolutional neural network capable of determining the features (e.g., the convolutional filters) that allow it to properly recognize the digits 0 to 9.

First, we create a CNN with this architecture. 3 convolutional layers

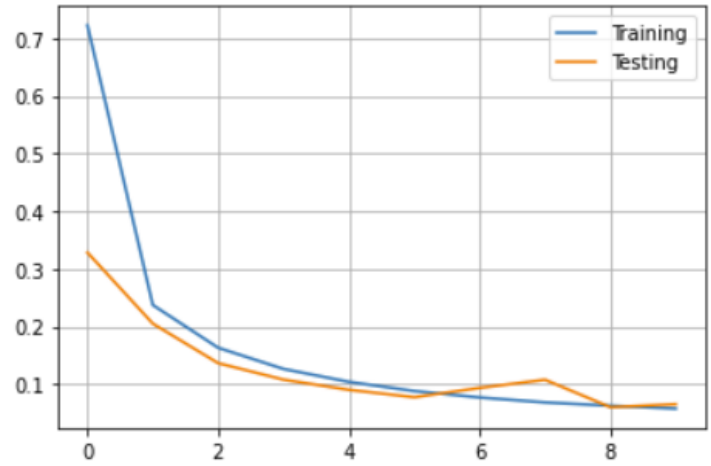
Layer (type)	Output Shape	Param #
l0 (InputLayer)	[(None, 28, 28, 1)]	0
l1 (Conv2D)	(None, 28, 28, 9)	234
l1_mp (MaxPooling2D)	(None, 14, 14, 9)	0
l2 (Conv2D)	(None, 14, 14, 9)	2034
l2_mp (MaxPooling2D)	(None, 7, 7, 9)	0
l3 (Conv2D)	(None, 7, 7, 16)	1312
l3_mp (MaxPooling2D)	(None, 3, 3, 16)	0
flat (Flatten)	(None, 144)	0
l4 (Dense)	(None, 25)	3625
l5 (Dense)	(None, 10)	260

With :

```
batch_size = 256
n_epoch = 10
```

After the set has been trained, we can see the results :

```
Test score: 0.06489972025156021
Test accuracy: 0.979200005531311
```



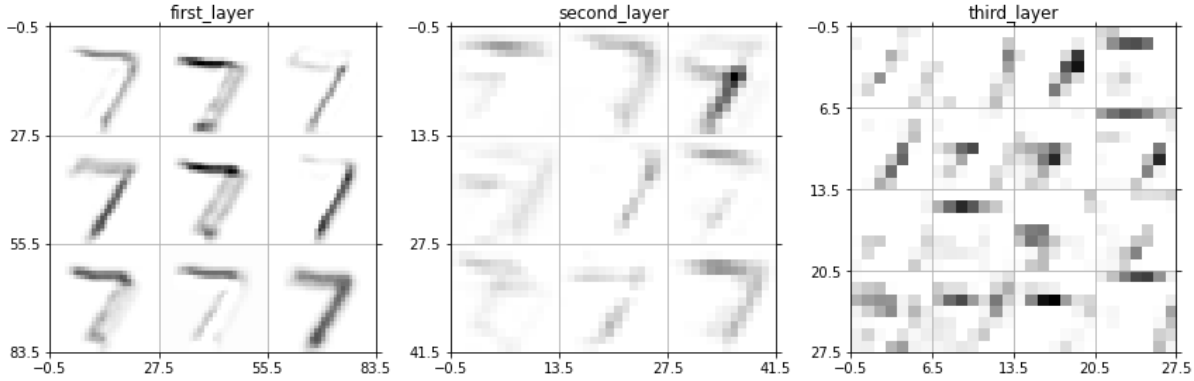
The X axis is the epoch, and the Y axis is the loss on the validation set and on training set. The accuracy is very good !

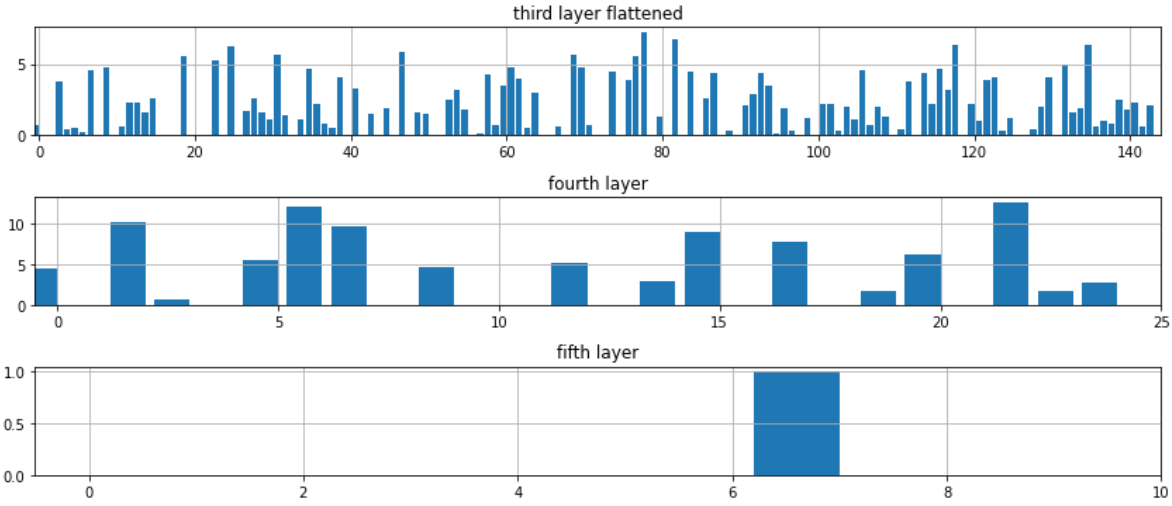
On the confusion matrix, we see that the set is globally well classified, there is no bias !

```
[ 968,    0,    0,    2,    0,    1,    1,    1,    6,    1],
[   0, 1121,    2,    1,    0,    0,    0,    3,    8,    0],
[   1,    0, 1010,   10,    0,    0,    0,    3,    8,    0],
[   0,    0,    0, 1003,    0,    2,    0,    1,    4,    0],
[   0,    0,    1,    2,  966,    0,    2,    0,    4,    7],
[   2,    0,    0,    9,    0,  875,    1,    0,    3,    2],
[   3,    4,    1,    1,    5,   18,  911,    0,   15,    0],
[   0,    0,    7,    4,    1,    0,    0, 1008,    4,    4],
[   1,    0,    3,    2,    1,    1,    0,    3,  960,    3],
[   0,    1,    1,    6,    7,    4,    1,    5,   14,  970]]
```

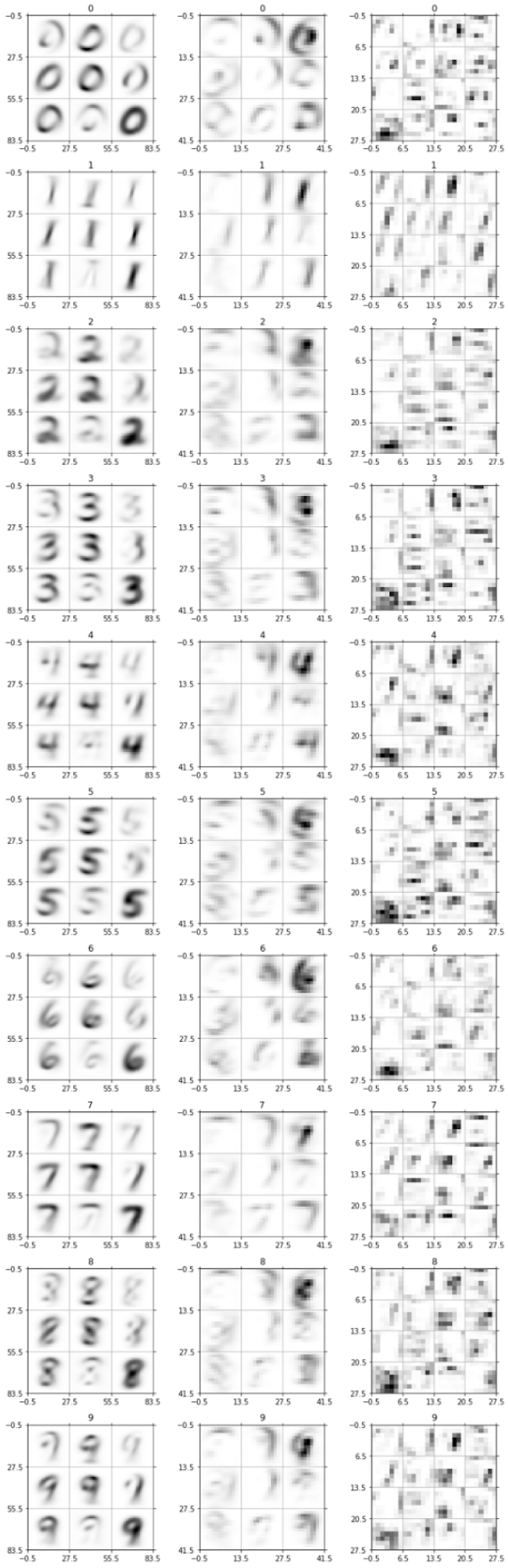
In the next picture, we can see the representation of the model. The three first layers are the convolutional layers. The next layers are flat & dense. The upper layers contains more spatials informations of the input image than the deeper layers. In the fifth layer the model give 7 at his output (which is right !)

We can see the patterns recognized by the 3 layers of convolution. They are mainly diagonal lines and diagonal lines with horizontal line (the shape of 7), for this example (number 7).

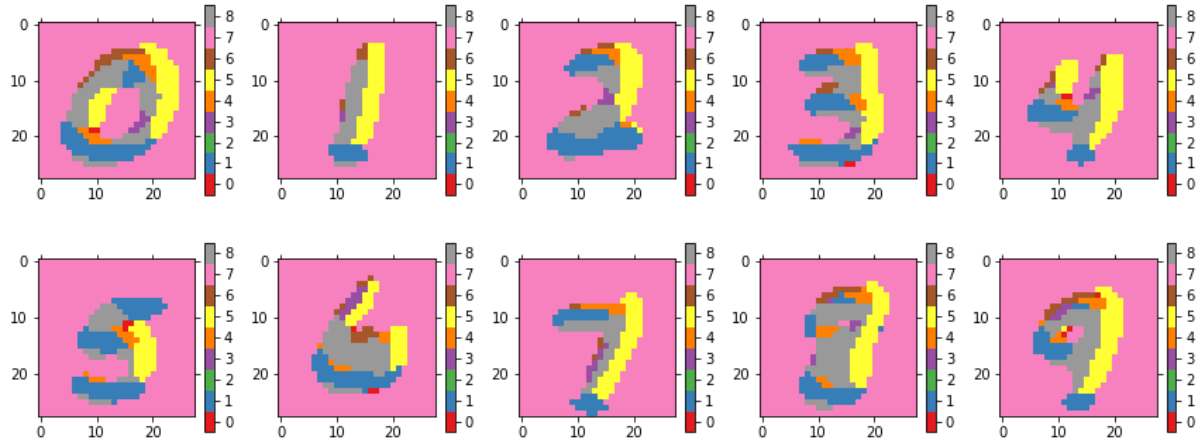




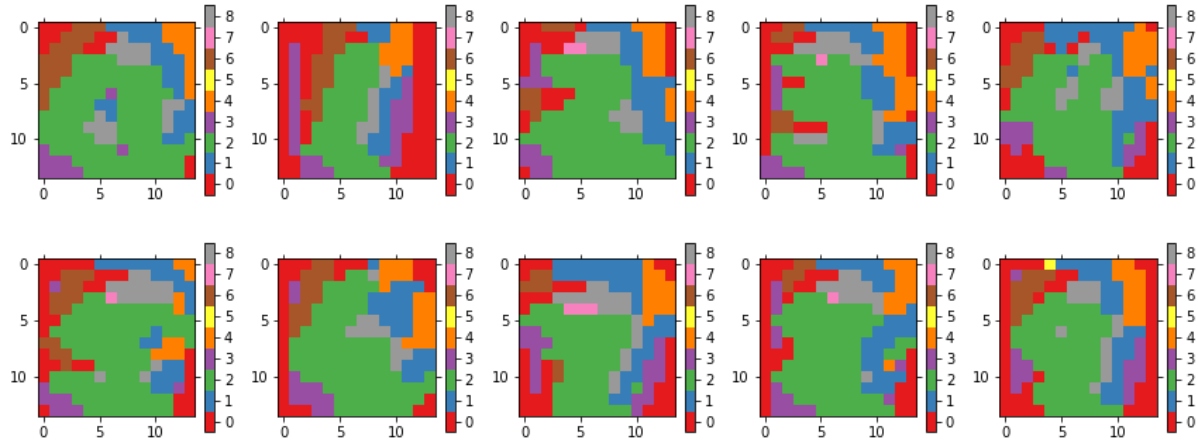
We can see the patterns for all numbers, and for the 3 convolutionnal layers.



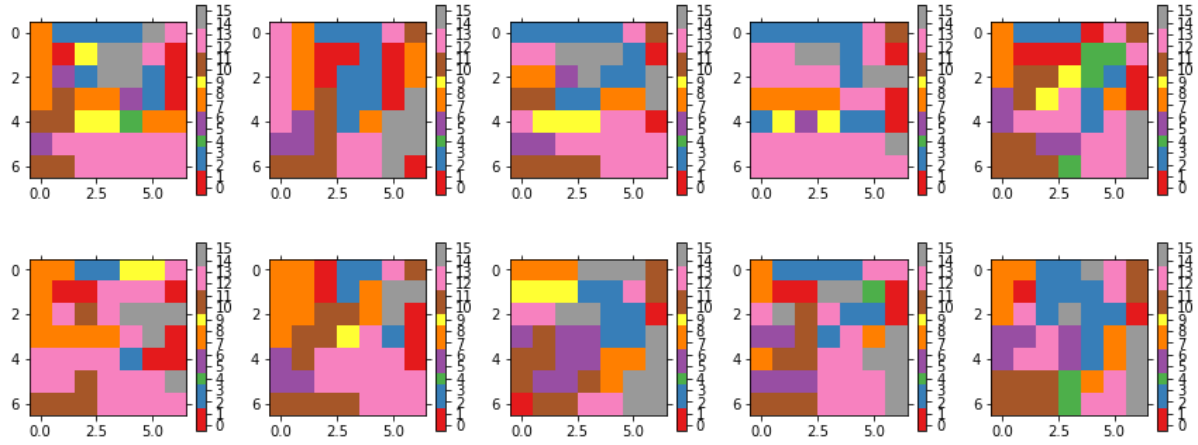
In the next picture, we can observe the “filter activation statistics”. It shows the most important characteristics of the class. For example, blue, correspond to the less essential characteristics that determine the class of the input. Because for this example, the blue parts are horizontal lines and in every number there are horizontal lines. It’s interesting how much the background shape is influential in every number (2nd importance for every number).



In the second layer, we can much hardly determine the number. With the human perspective, the green part is the better characteristic, because our brain can easily find which number it is. But for a computer, others parts of the picture are much determinant !



For the third layer, it’s not human-readable. Everything is coded in the pattern. And only the non-human readable parts of the last layer are still take in count.



Exercice 2

Activation maximization as a means for understanding a CNN model

The objective of this exercise is to try to do an activation maximization using the keras-vis python package. The idea behind activation maximization is simple: generate an input image that maximizes the output activations of a given unit (filter) in the network.

Test different values of *tv_weight*

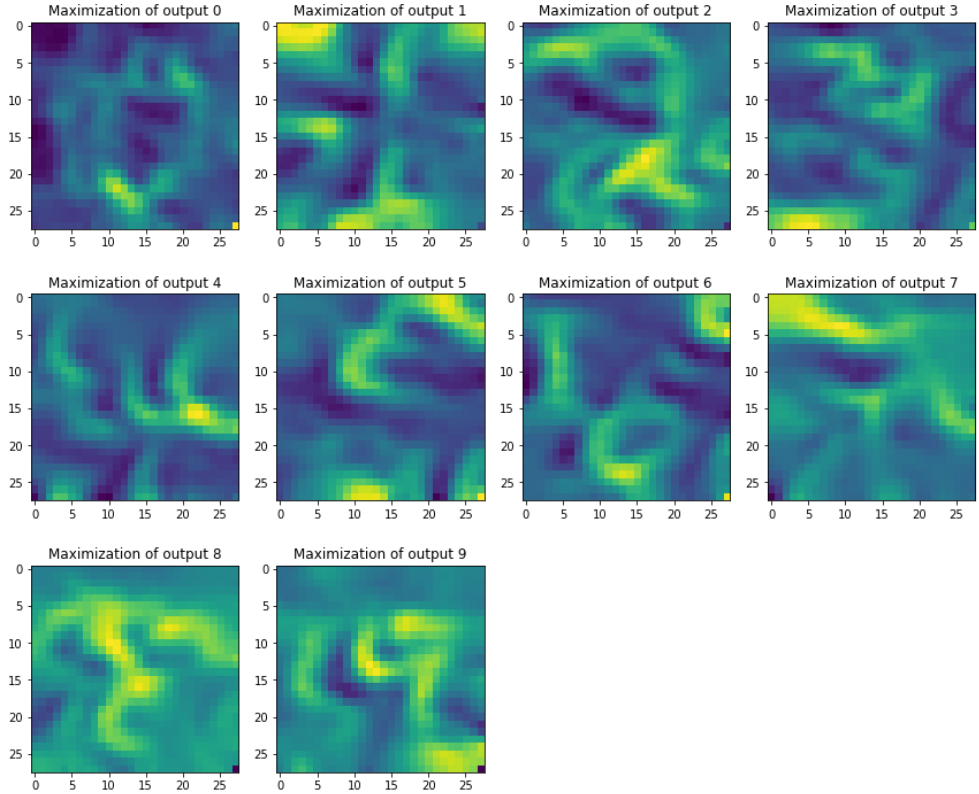
- Try values between 0.1 and 20 (for example: [0.125, 0.25, 0.5, 1, 2, 4, 8, 16])

tv_weight parameter is the weight param for TotalVariation regularization loss.

We tried several values between 0.1 and 20. Some activation maximization are not human-readable, and some yes. That's why we have to test several values.

- Select the regularization parameter that gives the best images (more realistic)
- Show the images that maximize each one of the outputs of the network

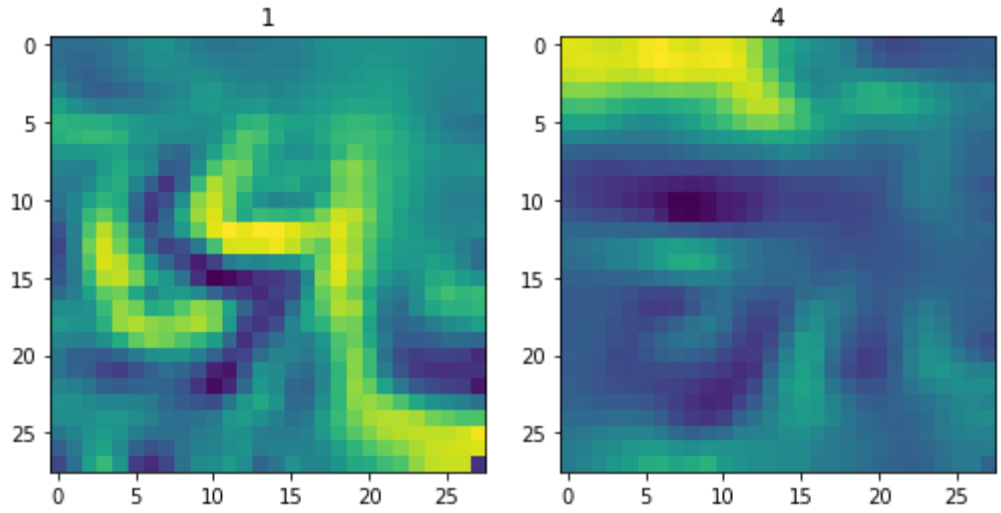
In the following image, it can be observed the activation maximization for the number 0 to number 9. The tv_weight value is equal to 2, which is the most well looking parameter.




```
tv_weight = 2
```

Maximize two outputs at the same time (filter_indices=[f1, f2])

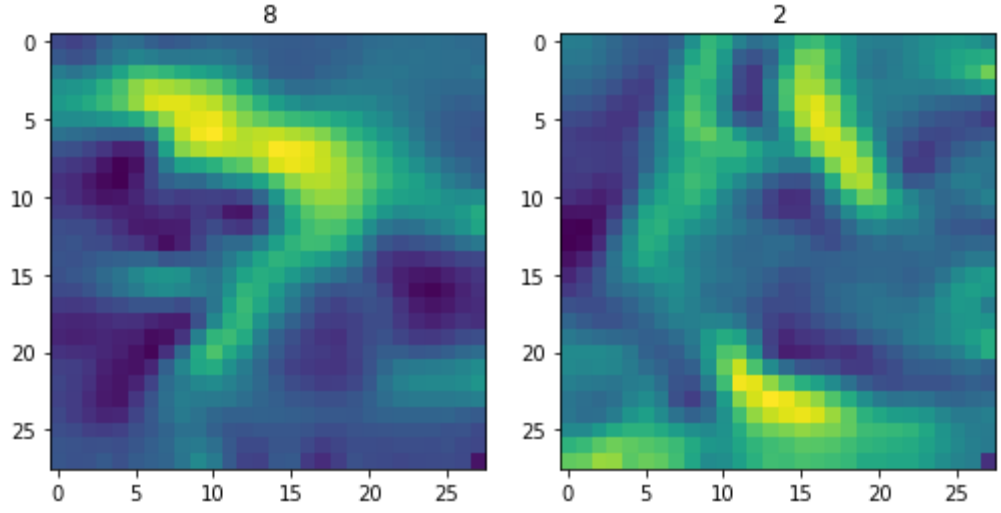
- Try two classes with similar shape like 1 and 7 or 4 and 9



9 & 4, tv_weight = 1 1 & 7, tv_weight = 4

In this case, because the shape of the 2 numbers are close. In the activation maximization, we can see the 2 numbers.

- Try two classes with very different shapes like 0 and 1 or 7 and 8



7 & 8, tv_weight = 8 0 & 1, tv_weight = 2

In this case, because the shape of the 2 numbers are not close at all. In the activation maximization, we can only part of imaginary numbers. That's the common

characteristics between the 2 numbers, but that's not human-readable, because the human brain didn't recognize numbers like this.

- How activation maximization can be useful for understanding a deep neural network? Explain

Activation Maximization detects the patterns that trigger a particular neuron in a deep network **by transforming the pixels of a randomly initialized image to maximize the activation** of the neuron keeping the weights of the network and desired output constant. It permits us to understand what pattern is detected by the neuron layer. And it can be used to avoid bias (like we see in the next exercice).

Exercice 3

Class Activation Maps

The objective of this exercise is to discover another way of visualizing attention over input using Class activation maps and grad-CAM.

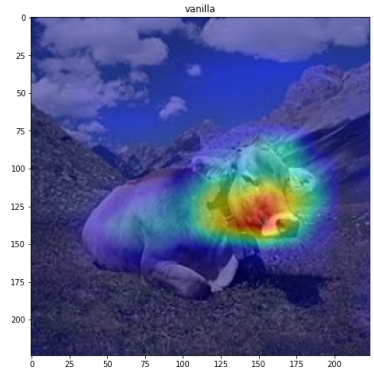
1. Compute the activation map for all the images we give you.
2. Do the maps reflect the class of the pictures?

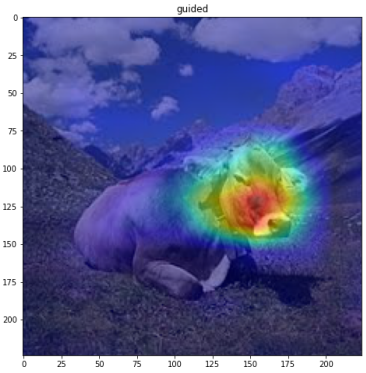
   Result of the predicted classes for all the images.

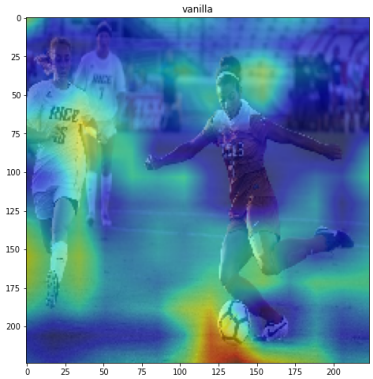
Image	Expected result	result	OK
Cat01.jpg	cat	`Egyptian_cat, 0.92138064`	TRUE
soccer02.jpg	soccer ball	`'soccer_ball', 0.9329395`	TRUE
soccer03.jpg	soccer ball	`'soccer_ball', 0.658205`	TRUE
miniskirt.jpg	miniskirt	`'miniskirt', 0.5854134`	TRUE
cow.jpg	cow	`'ox', 0.43208575`	FALSE

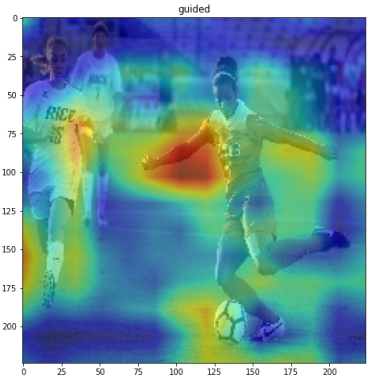
3. Are there images in which the activation maps highlight zones that do not belong to the object? Which ones?

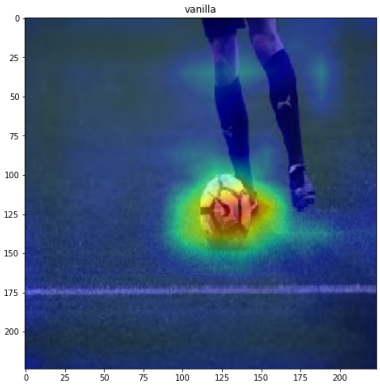
   Yes, for the soccer ball, the algorithm highlight the soccer player. And for the miniskirt, the algorithm highlight the legs of the woman too. (see in images below)

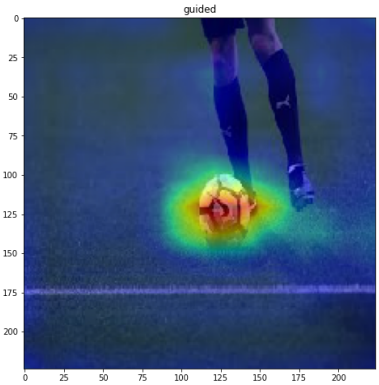

vanilla

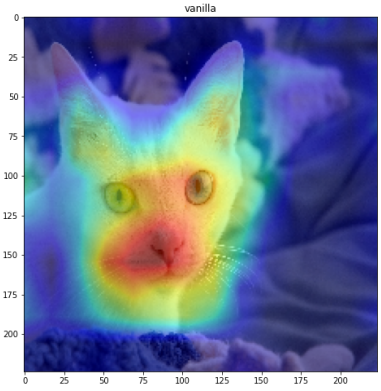

guided

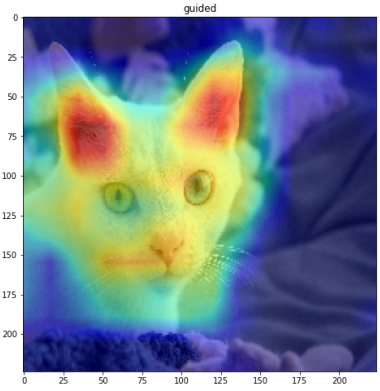

vanilla

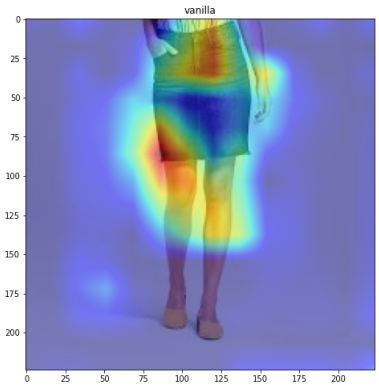

guided

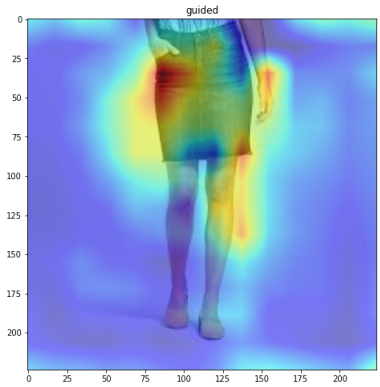

vanilla


guided


vanilla


guided





4. Look at the provided images and observe the images that were used to train the classifier on these specific classes (i.e., soccer ball). Do you think that all images have been labelled appropriately ? explain.

   The training image contains a soccer player and a soccer-ball. It can eventually be renamed as “soccer-player.png”.

5. Hypothesize about the behavior of the network observed in 3.

   The 2 results for one image, represent 2 different ways of recognizing a class in an image. For the first result (left), it use a vanilla Gradients (Vanilla Backpropagation). In the second result, we use the guided Grad CAM who use Guided Backpropagation.

   Vanilla Grad CAM visualizations are class-discriminative and localize relevant image regions but do not highlight the fine-grained pixel importance.

   Guided backpropagation visualizes fine-grained details in the image. Its premise is: neurons act like detectors of particular image features, so when backpropagation, the gradient, negative gradients are set to zero to highlight the pixels that are important in the image.