PROBLEM 4 Yanis Tazi

Q1

In [1]:
```python
from PIL import Image, ImageDraw
Image.open('alexnet_PARAMS.png')
```

Out[1]:

Computations
Yanis Tazi

Layers	Description	Size of Tensor	Number of weights	Number of biases	Number of Parameters
Input	Image 227*227*3	227*227*3	0	0	0
Conv1	96 kernels of size 11*11,stride 4,0 padding	[(227-11+2*0)/4]+1 = 55 55*55*96	11*11*3*96=34848	96	34848+96= 34944
MaxPool1	3*3 stride 2	(55-3)/2 +1=27 27*27*96	0	0	
Conv2	256 kernels of size 5*5,stride 1, padding 2	[(27-5+2*2)/1]+1 =27 27*27*256	5*5*96*256 = 614400	256	614400+256= 614656
MaxPool2	3*3 stride 2	(27-3)/2 +1=13 13*13*256	0	0	
Conv3	384 kernels of size 3*3,stride 1 ,padding 1	[(13-3+2*1)/1]+1= 13 13*13*384	3*3*256*384= 884736	384	884736+384= 885120
Conv4	384 kernels of size 3*3,stride 1 ,padding 1	[(13-3+2*1)/1]+1= 13 13*13*384	3*3*384*384= 1327104	384	1327104+384= 1327488
Conv5	256 kernels of size 3*3,stride 1 ,padding 1	[(13-3+2*1)/1]+1= 13 13*13*256	3*3*384*256= 884736	256	884736+256= 884992
MaxPool3	3*3 stride 2	(13-3)/2 +1 =6 6*6*256	0	0	
FC1	4096 neurons	4096	6*6*256*4096= 37748736	4096	37748736+4096= 37752832
FC2	4096 neurons	4096	4096*4096= 16777216	4096	16777216+4096= 16781312
FC3	1000 neurons	1000	4096*1000= 4096000	1000	4096000+1000= 4097000
Total					**62378344**

Q2

In [2]:
```
Image.open('vgg_PARAMS.png')
```

Out[2]:

Layer	Number of Activations (Memory)	Parameters (Compute)
Input	224*224*3=150K	0
CONV3-64	224*224*64=3.2M	(3*3*3)*64 = 1,728
CONV3-64	224*224*64=3.2M	(3*3*64)*64 = 36,864
POOL2	112*112*64=800K	0
CONV3-128	112*112*128 = 1.6M	(3*3*64)*128=73728
CONV3-128	112*112*128 = 1.6M	(3*3*128)*128=147456
POOL2	56*56*128=400K	0
CONV3-256	56*56*256= 800K	(3*3*128)*256=294912
CONV3-256	56*56*256=800K	(3*3*256)*256 = 589,824
CONV3-256	56*56*256= 800K	(3*3*256)*256=589824
CONV3-256	56*56*256= 800K	(3*3*256)*256=589824
POOL2	28*28*256= 200K	0
CONV3-512	28*28*512=400K	(3*3*256)*512 = 1,179,648
CONV3-512	28*28*512=400K	(3*3*512)*512=2359296
CONV3-512	28*28*512=400K	(3*3*512)*512=2359296
CONV3-512	28*28*512=400K	(3*3*512)*512=2359296
POOL2	14*14*512=100K	0
CONV3-512	14*14*512 = 100K	(3*3*512)*512=2359296
CONV3-512	14*14*512 = 100K	(3*3*512)*512=2359296
CONV3-512	14*14*512 = 100K	(3*3*512)*512=2359296
CONV3-512	14*14*512 = 100K	(3*3*512)*512=2359296
POOL2	7*7*512= 25K	0
FC	4096	25088*4096=102760448
FC	4096	4096*4096 = 16,777,216
FC	1000	4096*1000=4096000
TOTAL	~16.3M	~138M

Table 1: VGG19 memory and weights

Q3

Input size L ,Filter (F*F) , Pad = 0 (P), Stride = 1 (S)

For one filter of size F*F,Output activation map length is L-F+1

For two stacked filters of size F*F, Output activation map length is (L-F+1)-F+1 = L-2F+2.

Therefore, for N stacked filters of size F*F, Output activation map length is $\boxed{L - NF + N}$**.**

For one filter of size (NF – N + 1)*(NF – N + 1), Output activation map lenth is L-(NF-N+1)+1=L-NF+N-1+1= $\boxed{L - NF + N}$**.**

For 3 stacked filters of size 5x5, Output activation map length is $\boxed{L - 3 * 5 + 3 = L - 12}$**. Therefore, the receptive field is 13*13.**

Q4

a)

The idea is to have efficient computation to be able to use very deep neural networks . For that, they have been using stacked 1*1 convolutions and this also reduce the problem of overfitting. This allows us to keep computational constraints while increasing depth of the network. We saw the benefits of using stacked of small filters in the previous question in terms of computational reduction.

In terms of intuition, we can say that usually we do not know the size of the filters that we want because detecting a specific characteristic of an image will vary from images to images so the idea is to have several different filter sizes on the same level.

Also, it preserves local and sparse correlations, parallel convolutional filters of different sizes

b)

For the naive one , it is

$$\boxed{32 * 32 * (128 + 192 + 96 + 256) = 32 * 32 * 672}$$

For the dimensionality reduction one, it is:

$$\boxed{32 * 32 * (128 + 192 + 96 + 64) = 32 * 32 * 480}$$

c)

For the naive version , we have :

$$(1 * 1 * 256 * 128 + 128) + (3 * 3 * 256 * 192 + 192) + (5 * 5 * 256 * 96 + 96)$$

For the dimensionality reduction version , we have :

$$(1 * 1 * 256 * 128 + 128) + (1 * 1 * 256 * 128 + 128) + (1 * 1 * 256 * 32 + 32)$$
$$+ (5 * 5 * 32 * 96 + 96) + (1 * 1 * 256 * 64) = 388672$$

d)

In the dimension reduction architecture, we have added an extra 1*1 convolution before adding the 3*3 and 5*5 filters. Doing so allowed us to reduce the number of input channel and it becomes much cheaper to compute the convolution filters now since the depth has decreased with the 1*1 convolutions.*

Dimension reduction module has reduced computational complexity compared to naïve version by a factor of 2.8!

```
In [ ]:
```