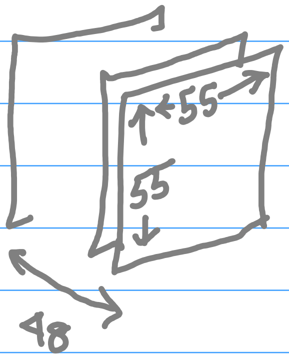
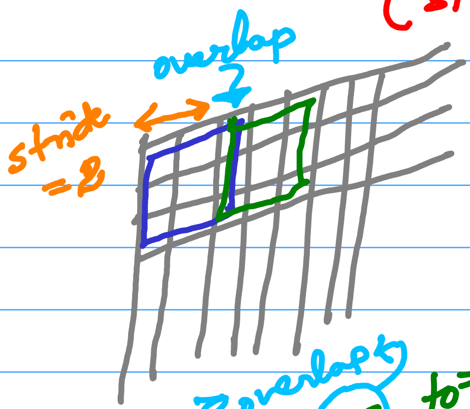
AlexNet (contd.)

(2) Second layer: 3x3 overlapping Max Pooling (stride = 2)





overlaps

(#0 to #2) (#2 to #4) (#4 to #6)

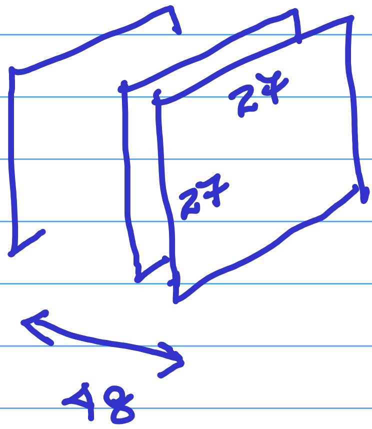
#0 #1 #2

... (#2k to #2k+2) ... (#52 to #54)

#k

$2k = 52 \Rightarrow k = 26$

→ There are 27x27 outputs
48 of these, 2 groups

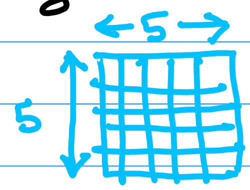


(3) 3-rd layer: Local Response Normalisation

→ only the values change, the size does not change.

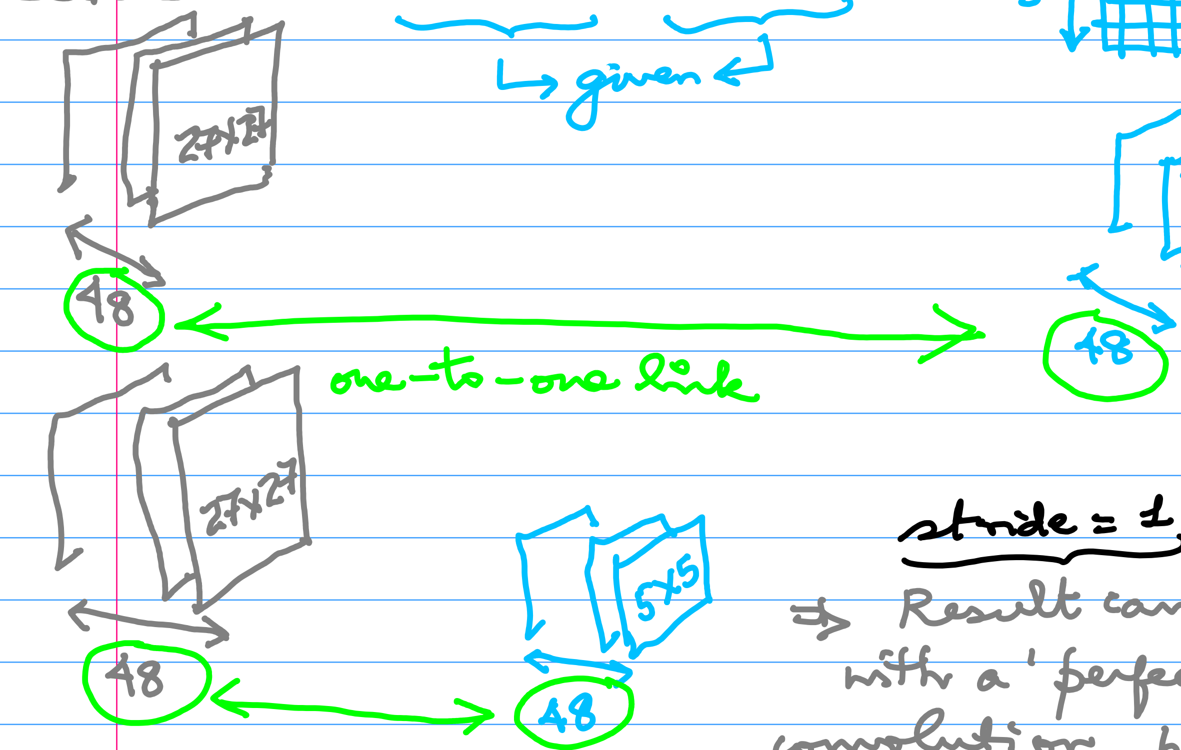
④ Second Convolutional Layer

2 groups of 128 kernels of size $5 \times 5 \times 48$



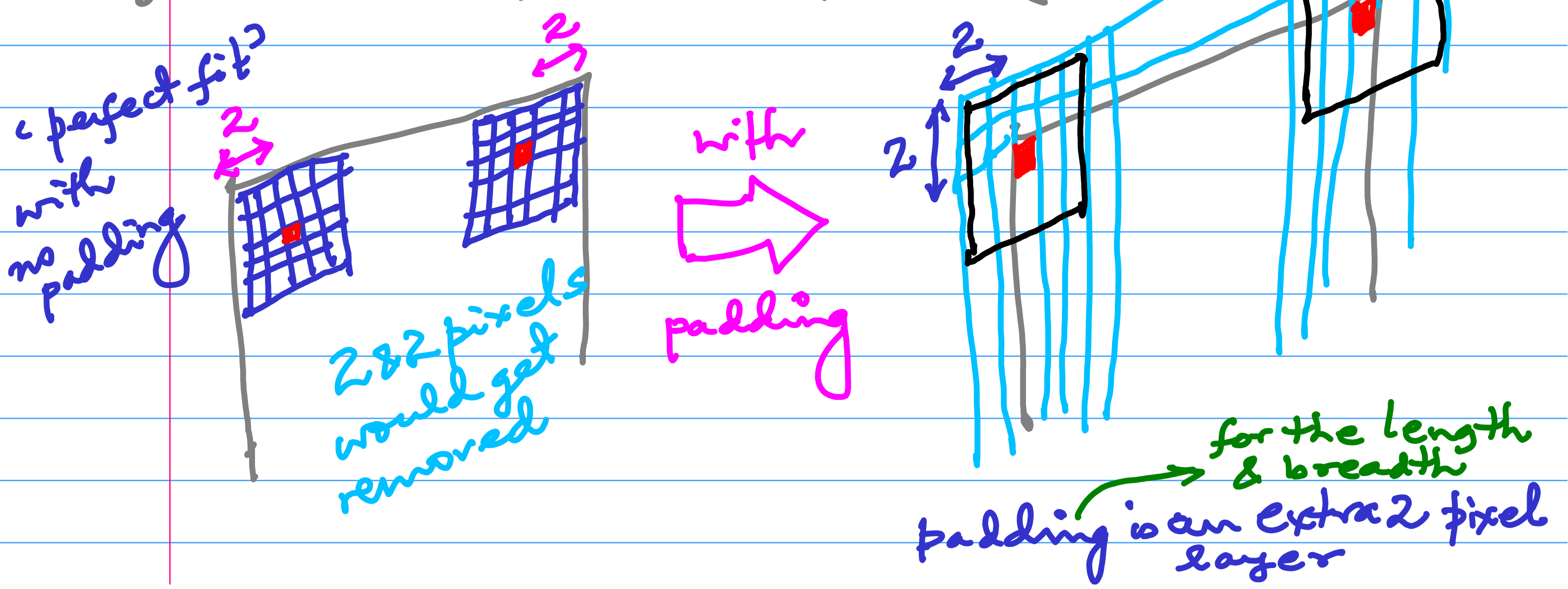
Input:

$\underbrace{\text{stride} = 1}$, $\underbrace{\text{pad} = 2}$ → given



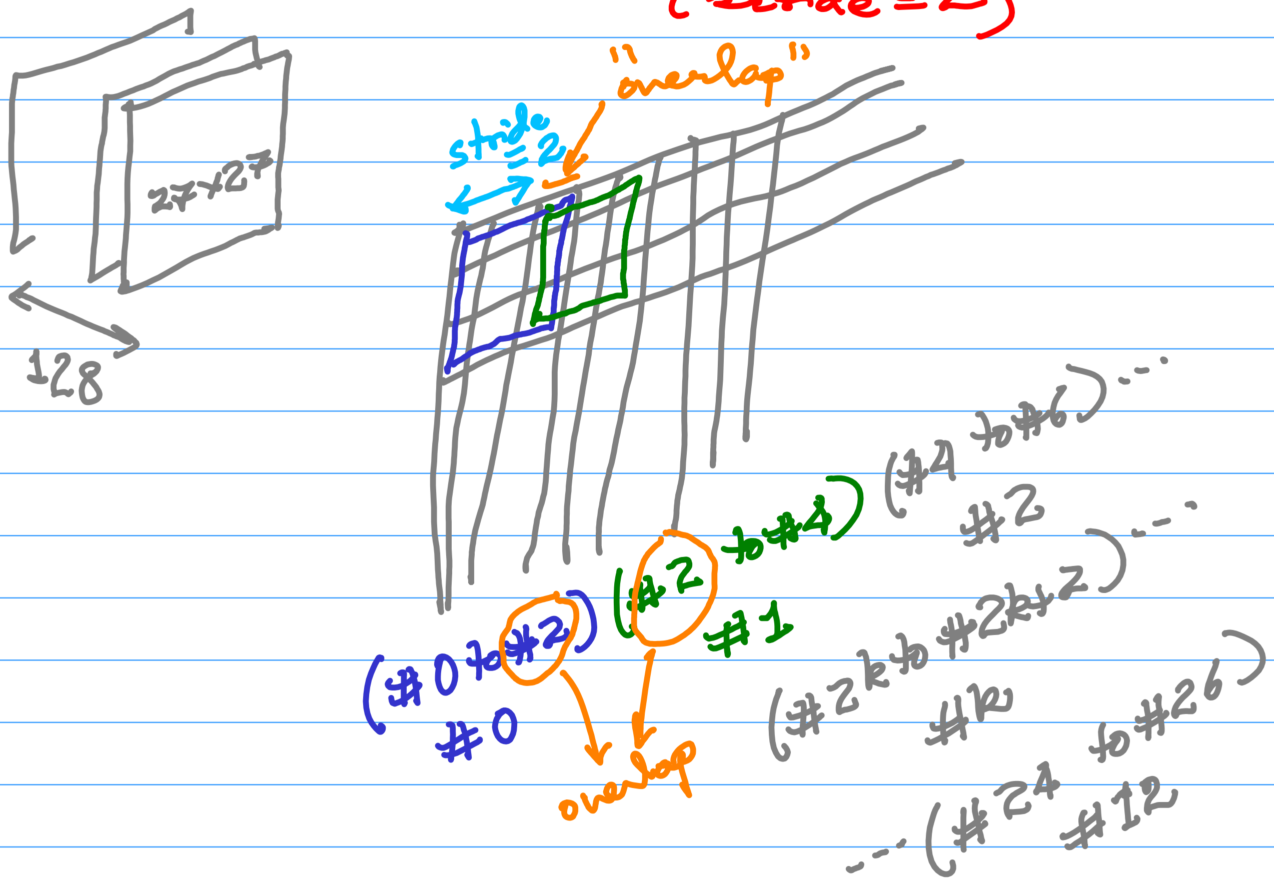
stride = 1 ⇒ Result can get smaller with a 'perfect fit' convolution, but here

pad = 2 → doesn't in this case, since the number of pixels which would otherwise get lost, get made up with the padding



→ The resultant image size remains 27 X 27
128 filters/kernels in 2 groups
→ output 27 x 27 x 128 x 2 groups.

(5) next layer: 3 x 3 overlapping Max Pooling
(stride = 2)



→ there are 13 x 13 x 128 outputs in 2 groups.

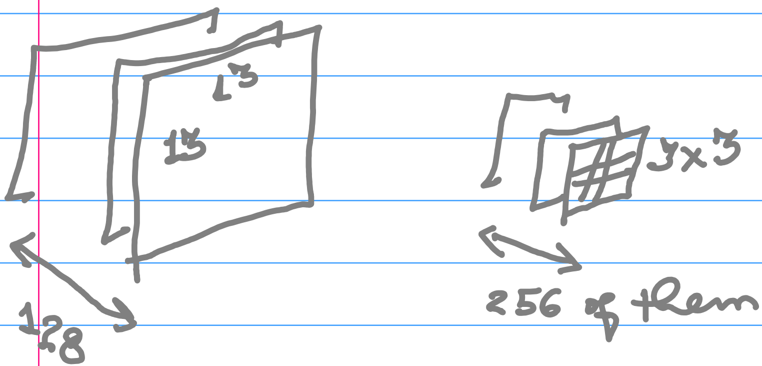
(6) Local Response Normalisation
(This does not change the output size: size-preserving, but changes the values)

AlexNet (contd).

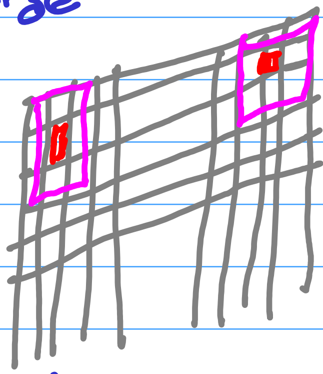
(7) Third Convolutional layer:

2 groups of 192 kernels of size $3\times3\times256$
stride = 1, pad = 1

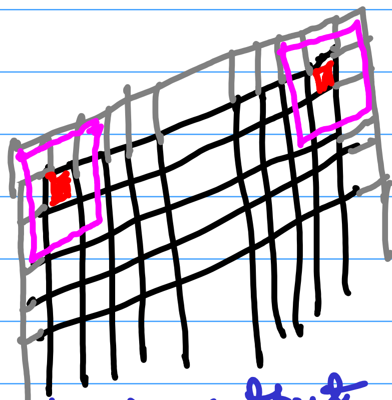
Input: 13×13 images $\times$ 128 in 2 groups.



First, let us consider the size



'just fit' $\rightarrow$ 12×12



pad = 1 $\Rightarrow$ output 13×13

Now, how do we account for the number? $\rightarrow$ heuristic!

To resolve 128, 256 and 192

$$128 \xrightarrow{+128/2} 192 \xrightarrow{+128/2} 256$$

nothing mentioned about pooling, so possibly
2 kernels each for the 128 to give 256 and then

some selection and pooling to give 192.

now that we have an understanding of padding, convolutions, stride & pooling, we will recognise that there are many heuristics to get actual numbers.

→ Try to look for conceptual ideas from different families of successful architectures.

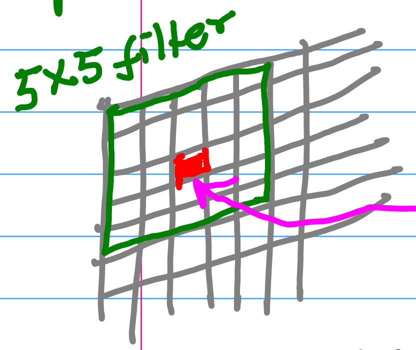
VGG - 16/-19 "Visual Geometry Group" at the University of Oxford

Karen Simonyan, Andrew Zisserman (2014)

Basic Concept:

The use of 3×3 filters/kernels in place of larger 11×11 or 7×7 filters.

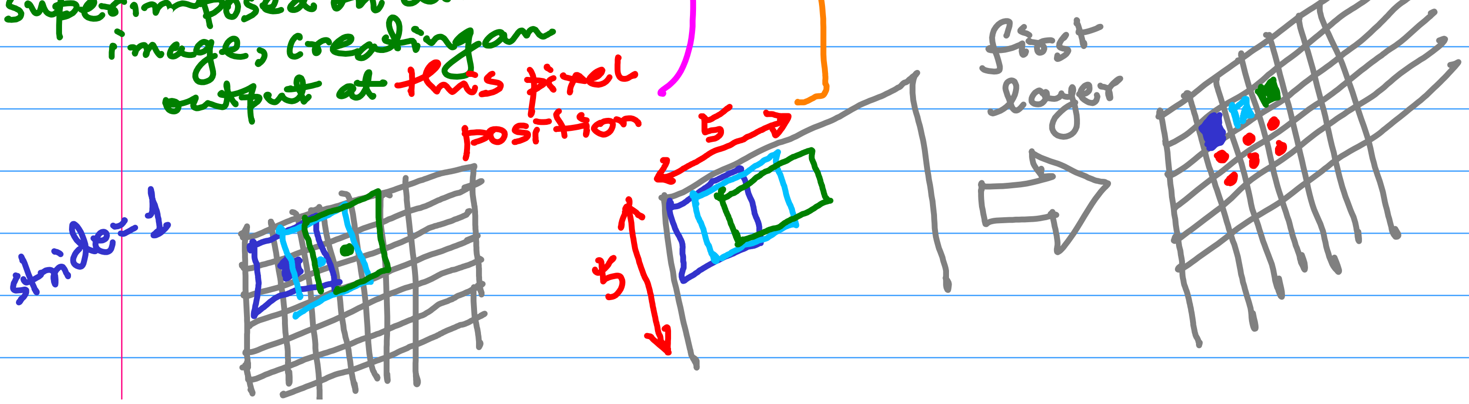
Result: Simpler architecture with a smaller no. of parameters, but an increased depth: (16 - 19 layers)

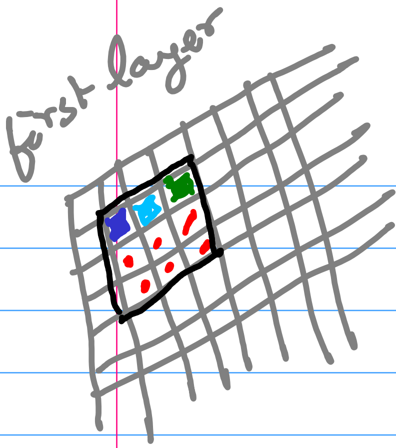


large 5×5 filter superimposed on an image, creating an output at this pixel position

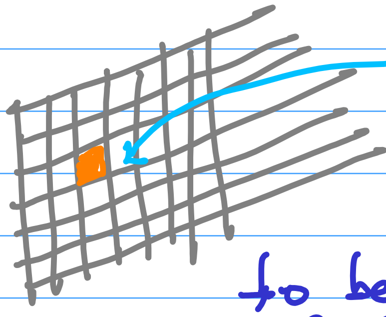
instead of this

2 layers of 3×3 filters: cover the same effective pixel positions as one larger 5×5 mask





we note that the pixel positions in the first layer allow a 3x3 mask/ filter/ kernel at another layer to be put right here



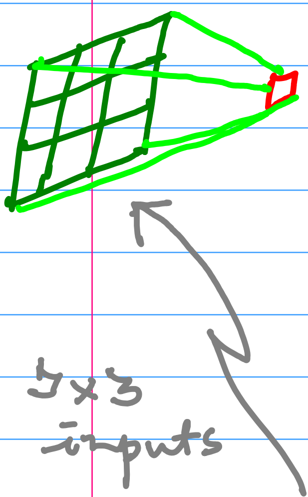
position of the output which we note (happily :-)) to be the position at which we calculated the output of the layer 5x5 mask.

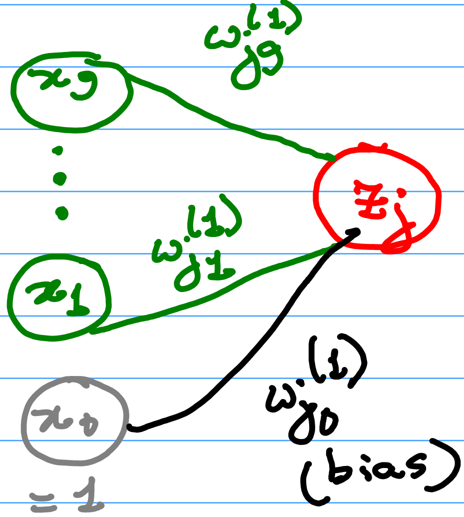
→ Advantage: there is a smaller # of parameters

(*) 5x5 filter : $5 \times 5 + 1$ (bias) $= 26$ parameters

2 layers of 3x3 : $2 \times (3 \times 3 + \underset{\text{bias}}{1}) = 2 \times 10 = 20$ parameters

These weights (parameters): they are the relative-position-invariant weights of the local receptive field





$$z_j = \sum_{i=1}^{9} w_{ji}^{(1)} x_i + w_{j0}$$

(activation)