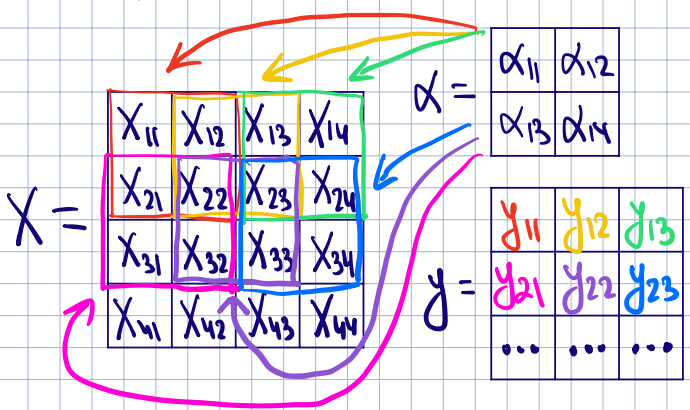
Inductive bias – incorporating our knowledge about data structure into architecture of neural networks

Convolutional layers – inductive bias for images

Convolution operation

RGB images are 3-dim tensors of size $H \times W \times 3$

Let us process each colour channel separately, so that we have a matrix of size $H \times W$



$$y_{kl} = \sum_{\substack{i=0,\dots,1 \\ j=0,\dots,1}} \alpha_{i+1,j+1} X_{k+i,l+j}$$

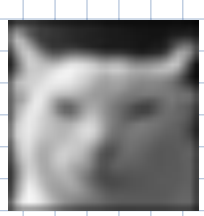
$$y = X * \alpha$$

α – convolution kernel

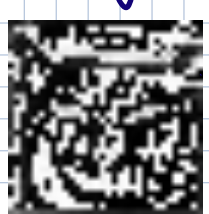
Motivation: processing image with filters:

- Original image

- Blur



- Edges



$$\alpha_{blur} = \frac{1}{9} \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix} \qquad \alpha_{edges} = \frac{1}{8} \begin{pmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{pmatrix}$$

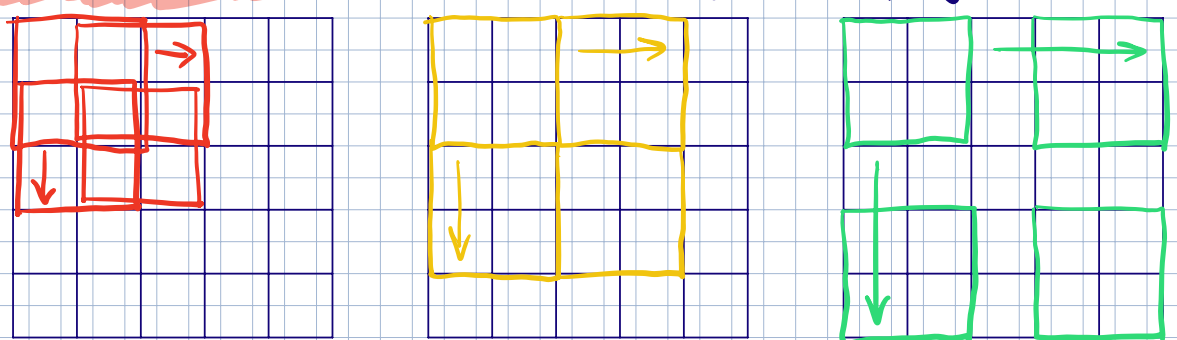
In convolutional layer, matrices α are trainable
Convolution is a specific type of linear transform, so we still need activation functions between them

Convolution hyperparameters

- **Kernel size (k):** $\alpha \in \mathbb{R}^{K_1 \times K_2}$

  Usually $K_1 = K_2 = K$, one matrix has $k_1 k_2 = K^2$ parameters

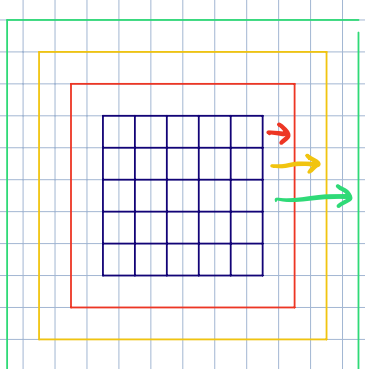
- **Stride (s):** how frequently we apply the kernel



$s = 1$ $s = 2$ $s = 3$

Output size reduces for higher strides: $H_{out} = \left\lfloor \frac{H_{in} - K}{s} + 1 \right\rfloor$

- **Padding (p):** used to control output size



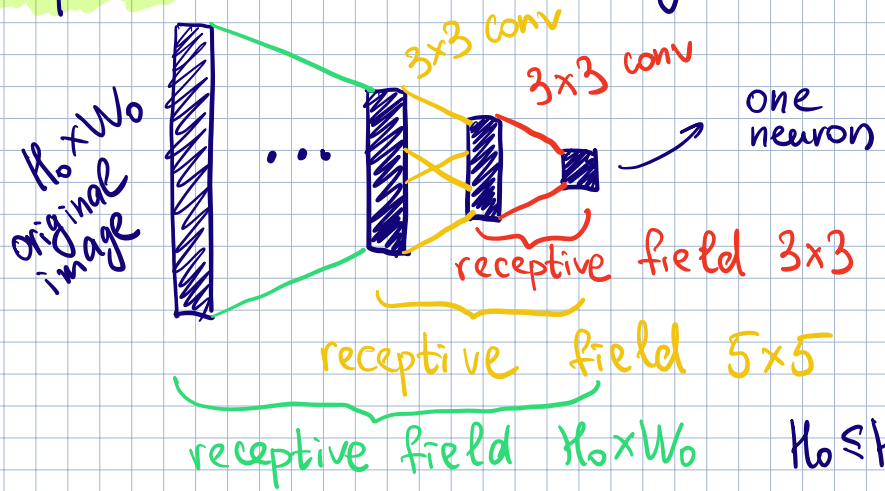
$p = 1$
$p = 2$
$p = 3$

Usually pad with zeros
(other options - reflective, circular)

for odd k and $s = 1$
we can take $p = \frac{k-1}{2}$ so that
$H_{out} = H_{in}$

Receptive field is the size of the patch of the original image, which affects a neuron after several convolutions

Example: let the whole image have size $H \times W$



$H_0 \le H,\ W_0 \le W$

More convolution layers, higher strides of convolutions

⇓

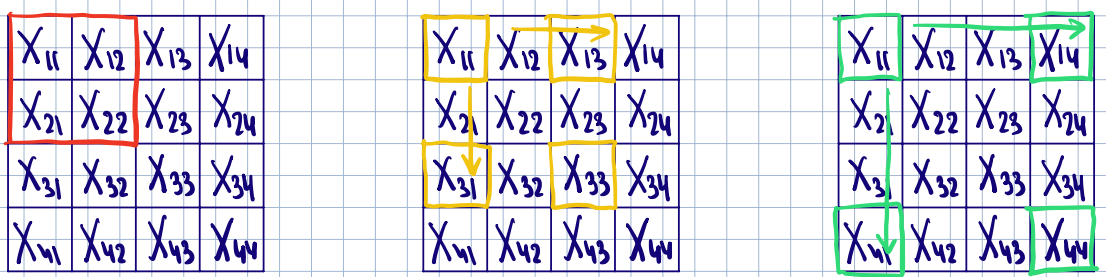
Larger the receptive field

⇓

Encoding more global features of the image

One more option to increase receptive field is to use **dilated convolutions**

- **Dilation (d):** convolution kernels "with holes"

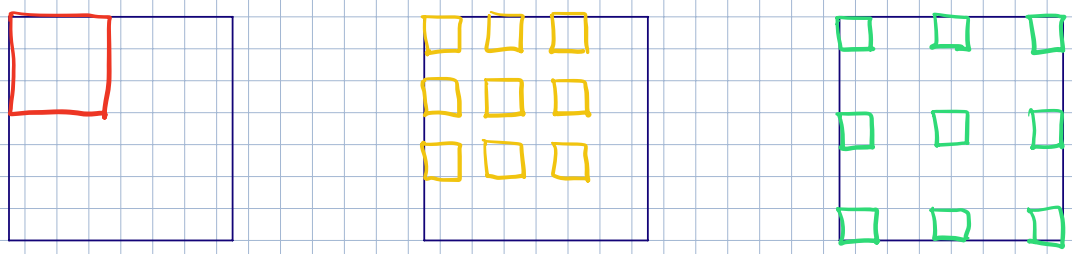


$y_{11} = \alpha_{11} x_{11} + \alpha_{12} x_{12} + \alpha_{21} x_{21} + \alpha_{22} x_{22}$

$y_{11} = \alpha_{11} x_{11} + \alpha_{12} x_{13} + \alpha_{21} x_{31} + \alpha_{22} x_{33}$

$y_{11} = \alpha_{11} x_{11} + \alpha_{12} x_{14} + \alpha_{21} x_{41} + \alpha_{22} x_{44}$

For 3×3 convolution kernel:



One convolution with kernel size k and dilation d has receptive field:

$k + (k-1)(d-1) = \cancel{k} + kd - \cancel{k} - d + 1 = d(k-1) + 1$

Convolutional layer

How to process several colour channels?
(Recall that RGB images have 3 channels and thus are 3-dim tensors, not matrices)

x: $C_{in} \times H_{in} \times W_{in}$ — input feature map
y: $C_{out} \times H_{out} \times W_{out}$ — output feature map
w: $C_{in} \times C_{out} \times K_h \times K_w$ — convolution kernel
b: C_{out} — bias vector

$1 \le i \le C_{out}$
$1 \le j \le C_{in}$

$$y_i = \sum_{j=1}^{C_{in}} x_j * w_{ji} + b_i$$

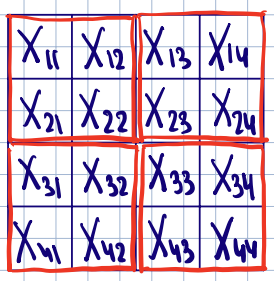
$x_j * w_{ji}$: convolution operation on matrices
b_i: bias (similar to linear layer)

Thus, we have one kernel matrix for each pair of input and output channels

- Number of channels is similar to hidden size in MLPs (i.e., width of the network), it is usually increased through the network
- Height and width of feature maps is usually decreased through the network to encode more global features. One option to do so is strided convolutions, another - pooling layer

Pooling

Layer for downsampling feature maps, legacy from older architectures, in more recent architectures is rare (while strided convolutions are more common)



- Average pooling: $y = \frac{1}{k^2} \sum_{ij} x_{ij}$
- Max pooling: $y = \max_{i,j} x_{ij}$

No trainable parameters!

Usually stride = kernel size, so kernels do not overlap