MACHINE INTELLIGENCE

UNIT - 5

CNNs

feedback/corrections: vibha@pesu.pes.edu

VIBHA MASTI

1. Convolution Operation

- Suppose we are tracking position of aeroplane using RADAR
- Measurements noisy
- Weighted average of k previous pos until time t

$$s_t = \sum_{a=0}^{k} x_{t-a} \, w_{-a}$$

revised estimate of x_t

$$= (x * w)_t$$

input, convolution, filter

- Calculating s_6

	w_{-6}	w_{-5}	w_{-4}	w_{-3}	w_{-2}	w_{-1}	w_0
W	0.01	0.01	0.02	0.02	0.04	0.4	0.5

X	1.00	1.10	1.20	1.40	1.70	1.80	1.90	2.10	2.20	2.40	2.50	2.70

S							1.80					
	0	1	2	3	4	5	6	7	8	9	10	11

$$s_6 = x_6 w_0 + x_5 w_{-1} + x_4 w_{-2} + x_3 w_{-3} + x_2 w_{-4} + x_1 w_{-5} + x_0 w_{-6}$$

- Similarly s_7

	w_{-6}	w_{-5}	w_{-4}	w_{-3}	w_{-2}	w_{-1}	w_0
W	0.01	0.01	0.02	0.02	0.04	0.4	0.5

X	1.00	1.10	1.20	1.40	1.70	1.80	1.90	2.10	2.20	2.40	2.50	2.70

S							1.80	1.96				
	0	1	2	3	4	5	6	7	8	9	10	11

1.1 For 2D inputs

- Images (black & white)
- Use mxn 2D filter
- Revised estimate S_{ij} of pixel I_{ij} (use mxn filter)

$$S_{ij} = (I * K)_{ij}$$

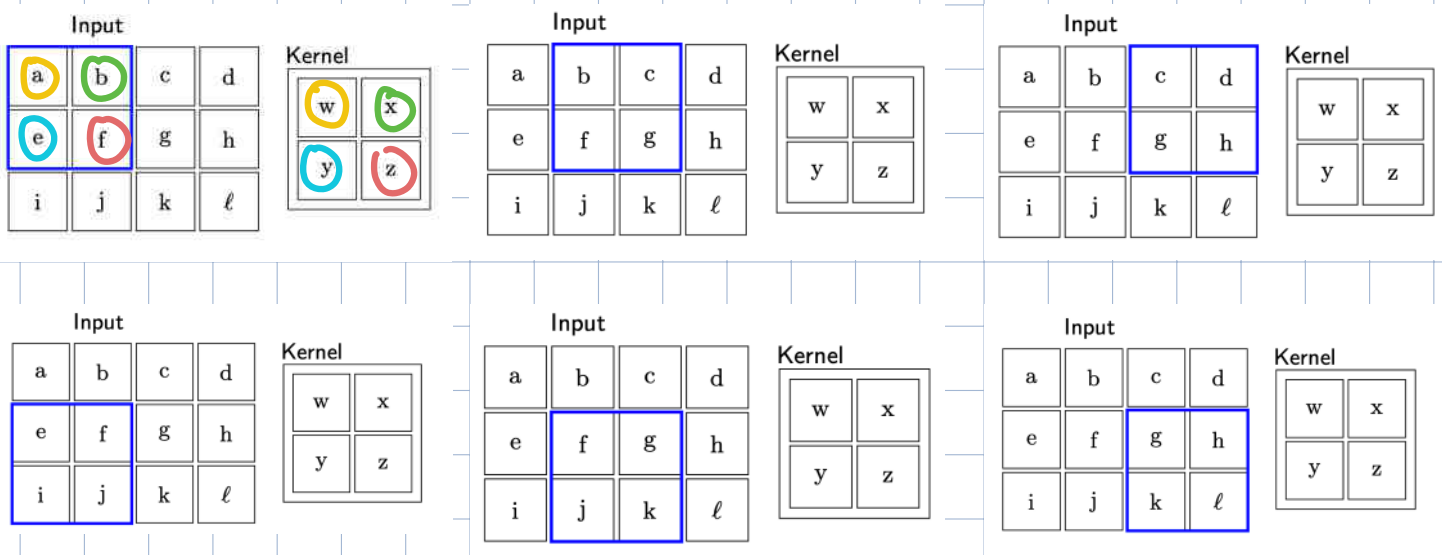
filter (kernel)

$$= \sum_{a=0}^{m-1} \sum_{b=0}^{n-1} I_{i-a,j-b} \; K_{a,b}$$

- Can choose to look ahead by mxn steps

$$S_{ij} = \sum_{a=0}^{m-1} \sum_{b=0}^{n-1} I_{i+a, j+b} \, K_{a,b}$$

- Example

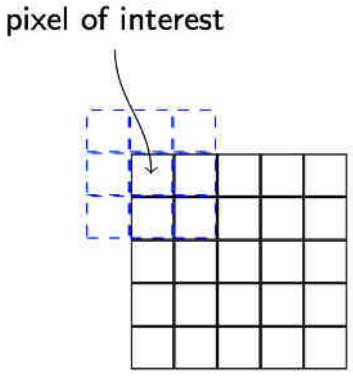


Output

aw+bx+ey+fz	bw+cx+fy+gz	cw+dx+gy+hz
ew+fx+iy+jz	fw+gx+jy+kz	gw+hx+ky+ℓz

- Most intuitive sense: estimate S_{ij} with average of surrounding cells

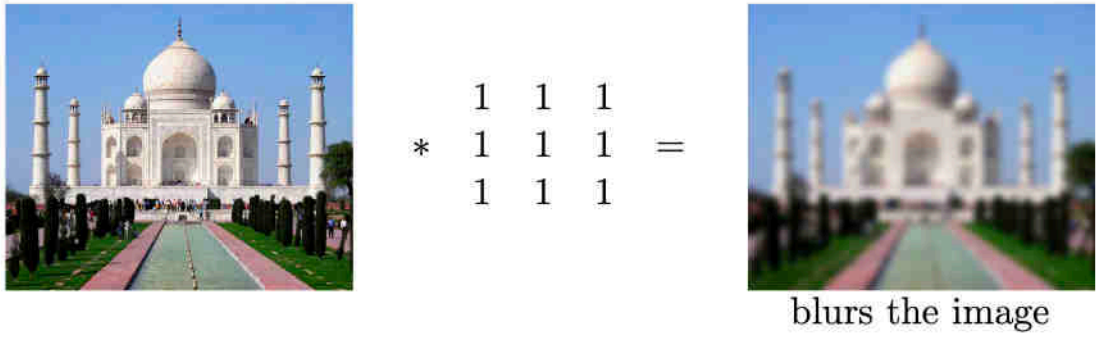
$$S_{ij} = \sum_{a=\lfloor -\frac{m}{2} \rfloor}^{\lfloor \frac{m}{2} \rfloor} \sum_{b=\lfloor -\frac{n}{2} \rfloor}^{\lfloor \frac{n}{2} \rfloor} I_{i-a, j-b} \; k_{\frac{m}{2}+a, \frac{n}{2}+b}$$



$m = 3$
$n = 3$

Examples of convoluting images

1. $\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$



blurs the image

2. $\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$

$$* \begin{matrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{matrix} =$$

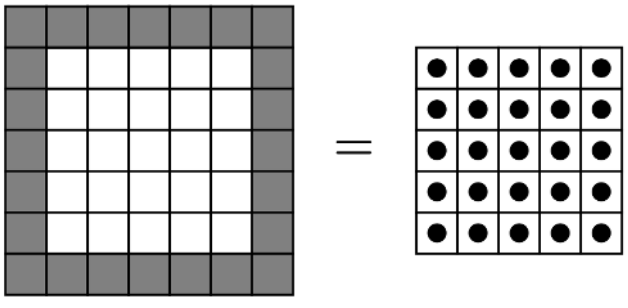
sharpens the image

3. $\begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix}$

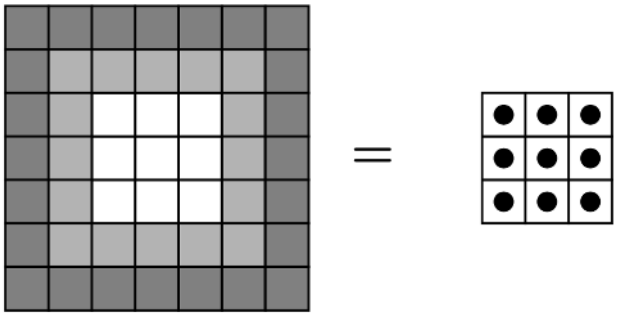
$$* \begin{matrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{matrix} =$$

detects the edges

- Output dimension: kernel not placed at boundaries



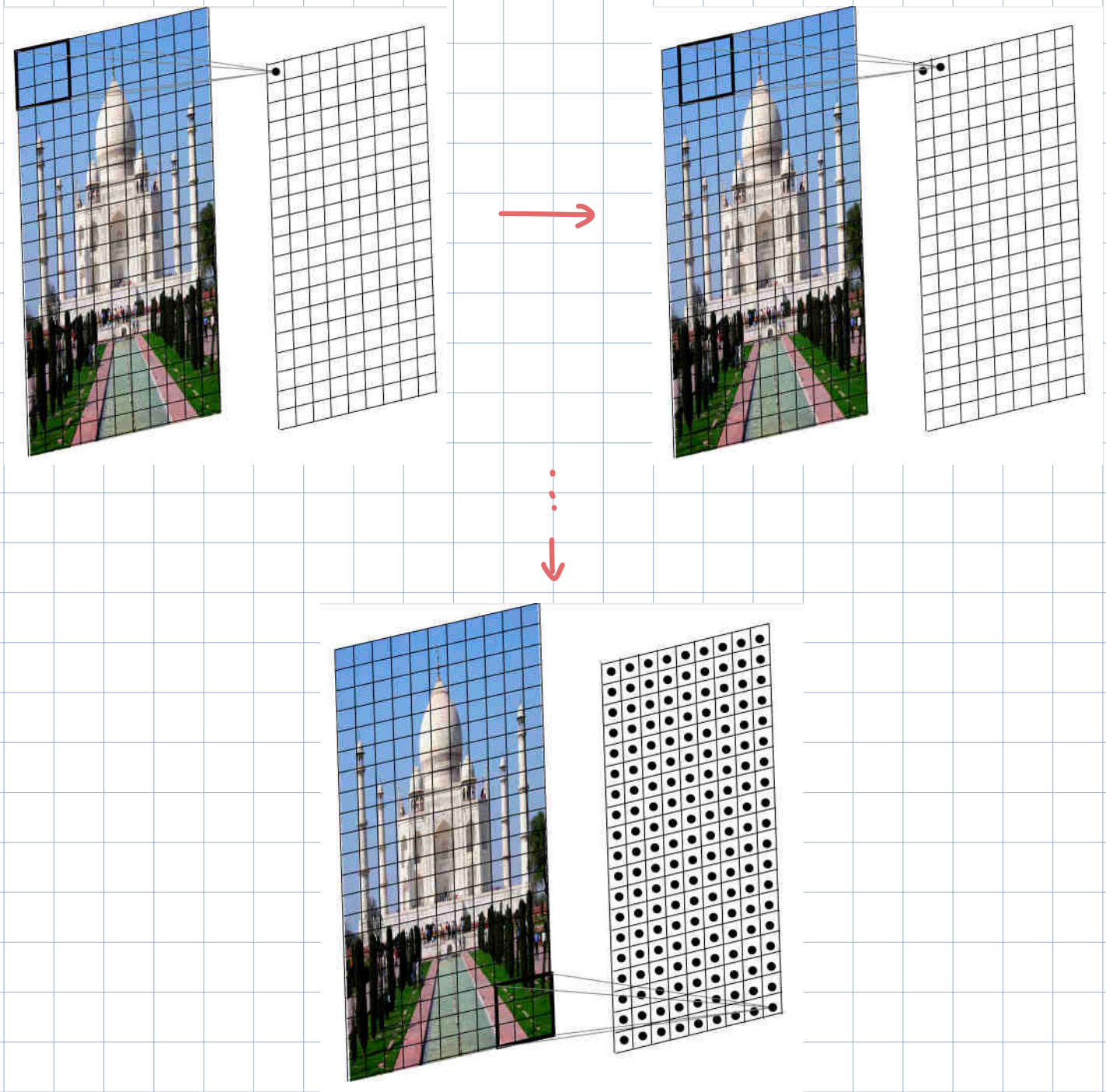
3x3 filter on 7x7 input



5x5 filter on 7x7 input

1.2 Applying a filter to an image

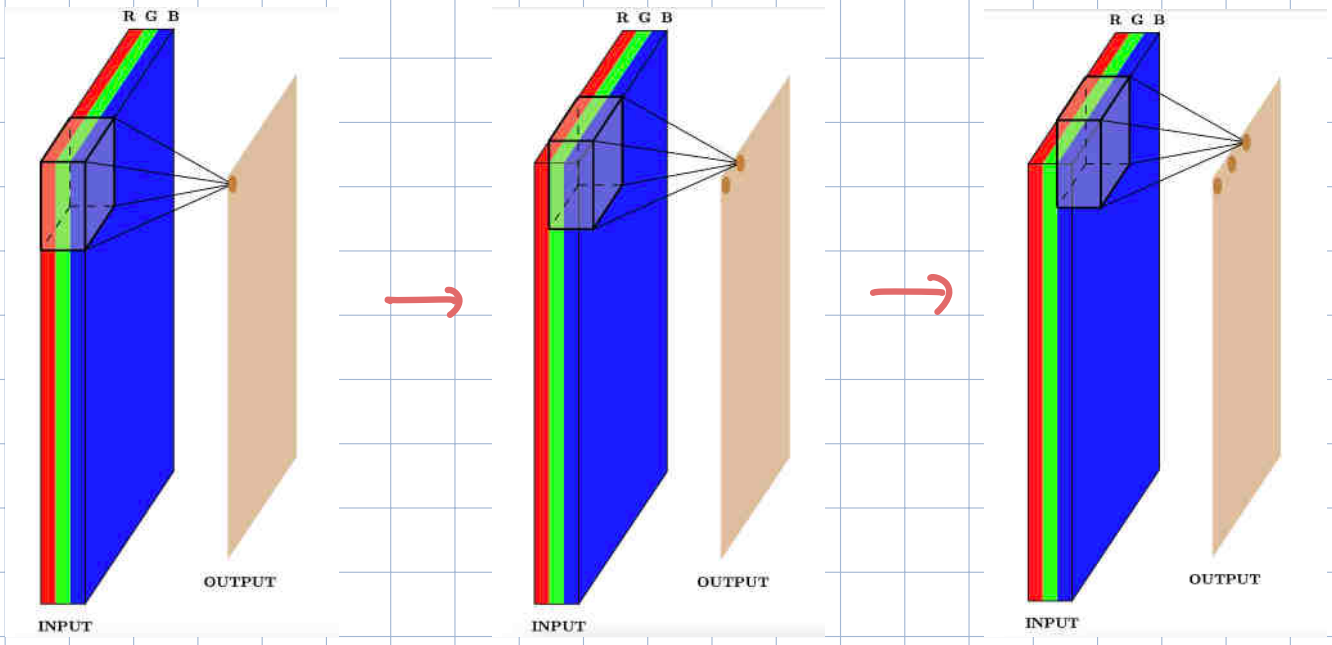
- Slide kernel over input image



- Result: feature map

1.2 3D Inputs

- Images are 3D (RGB)
- Slide 3D filter (volume) over the 3D input and compute convolution



- 2D convolution on 3D input (filter moves along height & width, but not depth)

Definition of Terms

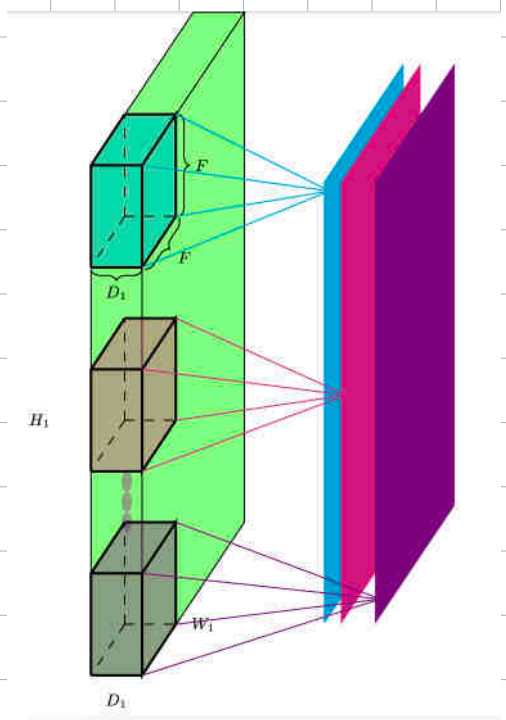
1. W_1 : width of original input
2. H_1 : height of original input
3. D_1 : depth of original input

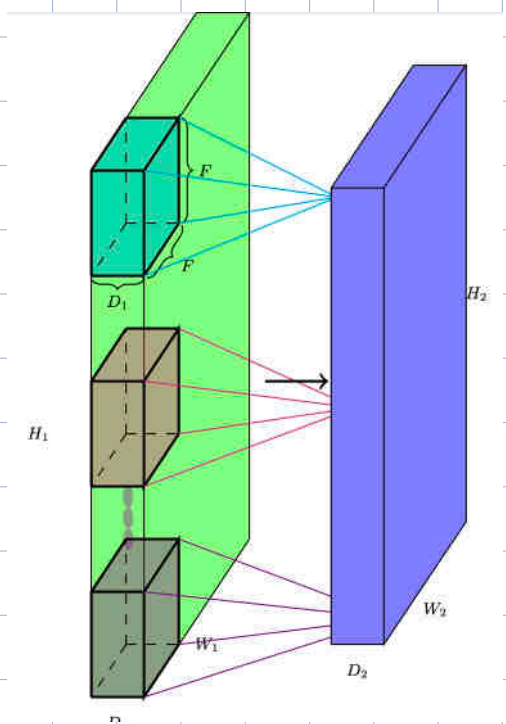
3. S : stride (intervals at which to apply filter)

4. K : number of filters

5. F : spatial extent of each filter (depth is same as input's)

6. $W_2 \times H_2 \times D_2$: output dimensions



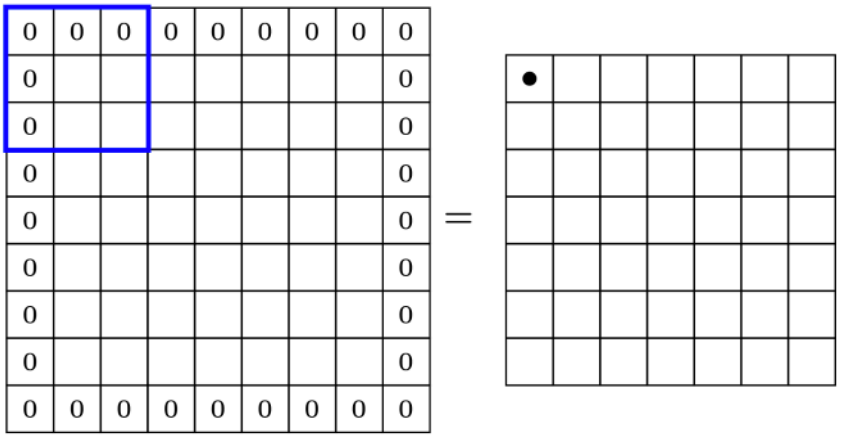


$$W_2 = W_1 - F + 1$$

$$H_2 = H_1 - F + 1$$

- To get same sized output as input, padding used

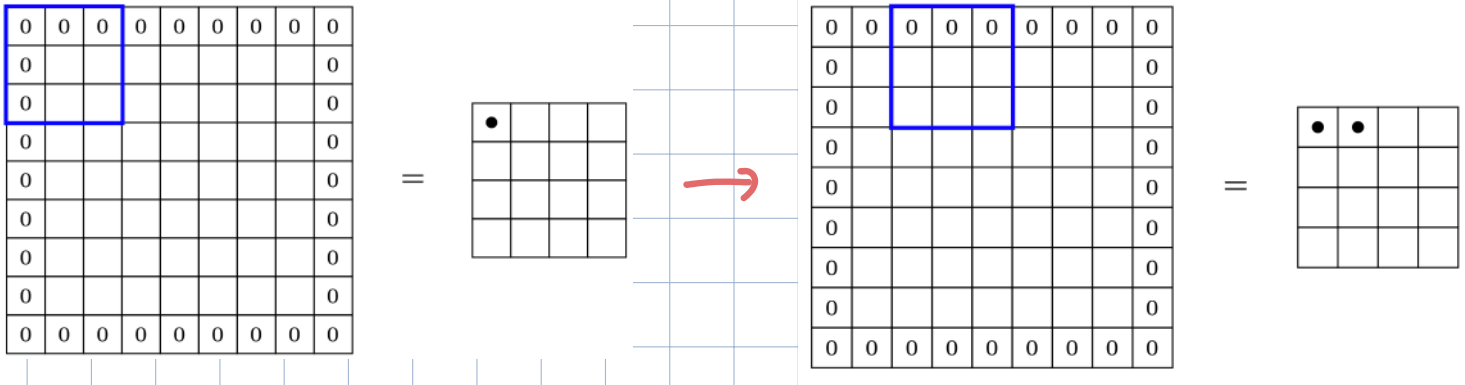
- Padding of P=1 for 3x3 kernel



$$W_2 = W_1 - F + 2P + 1$$

$$H_2 = H_1 - F + 2P + 1$$

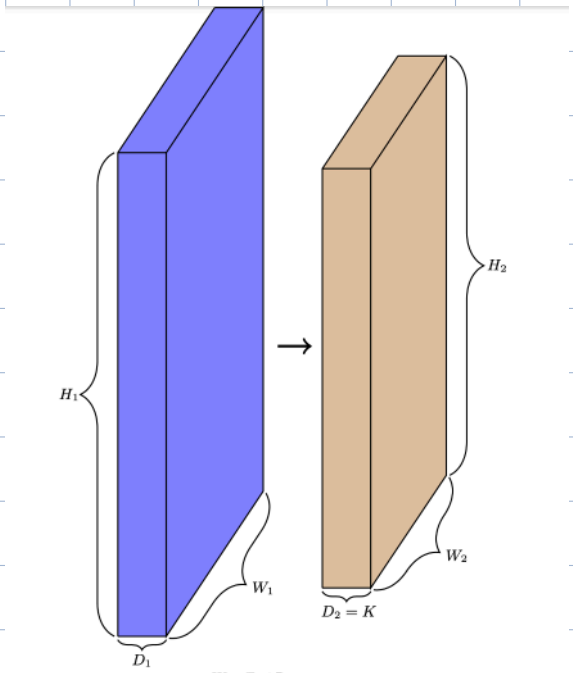
- Stride S: no of pixels to skip before filter applied again
- S=2 on 3x3 filter and P=1



$$W_2 = \frac{W_1 - F + 2P}{S} + 1 \qquad H_2 = \frac{H_1 - F + 2P}{S} + 1$$

- D_2 : K filters give output of depth k

$$D_2 = K$$

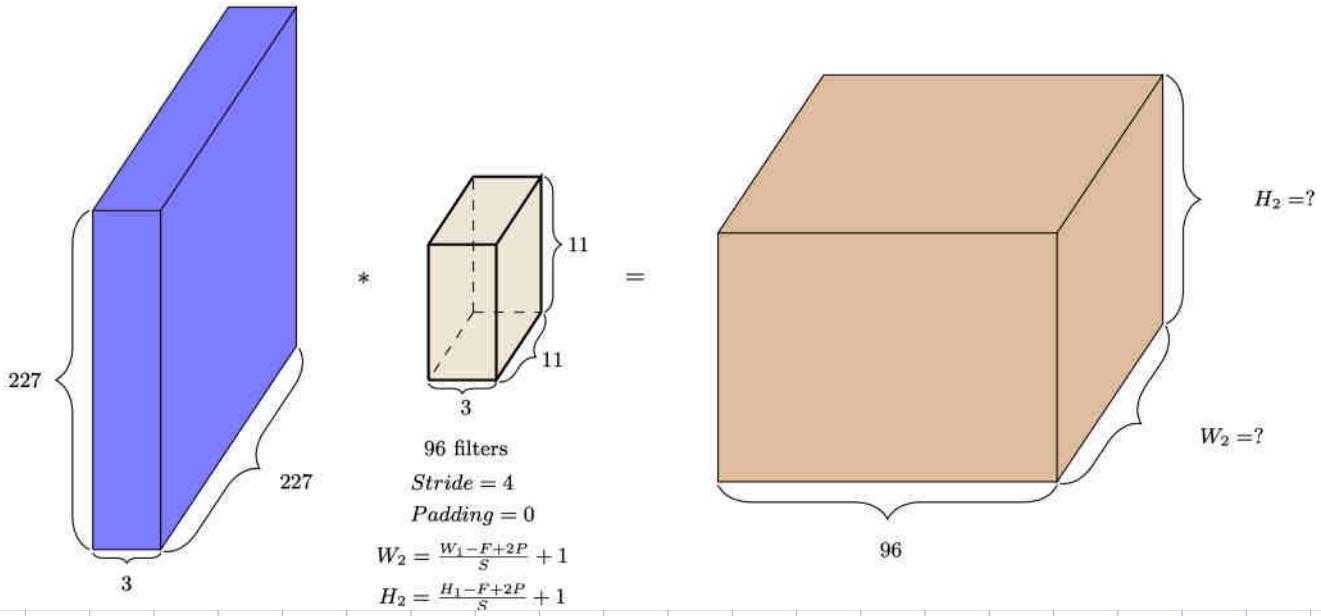


$$W_2 = \frac{W_1 - F + 2P}{S} + 1$$

$$H_2 = \frac{H_1 - F + 2P}{S} + 1$$

$$D_2 = K$$

Q: Exercise



$$W_2 = \frac{W_1 - F + 2P}{S} + 1$$

$$H_2 = \frac{H_1 - F + 2P}{S} + 1$$

$$H_2 = \frac{H_1 - F + 2P}{S} + 1 = \frac{227 - 11 + 0}{4} + 1$$

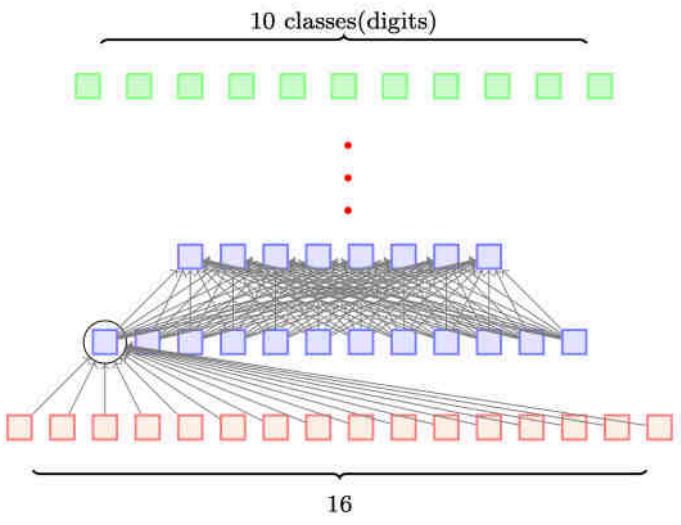
$$= 54 + 1 = 55$$

$$W_2 = \frac{W_1 - F + 2P}{S} + 1 = \frac{227 - 11 + 0}{4} + 1$$

$$= 54 + 1 = 55$$

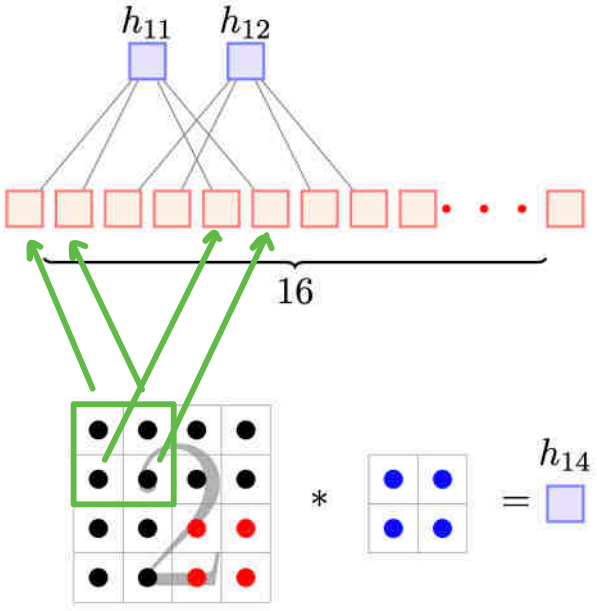
2. CNN

- In CNNs, kernels are learnt, so the features that they detect are unknown to us
- Trained like a FF NN with sparse connections



regular dense (feed-forward) NN

all 16 inputs contribute to h_{11}



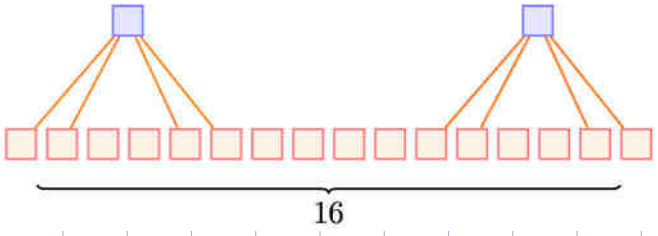
only few neurons participate in the computation of h_{11} (1, 2, 5, 6)

sparser connections than fully connected NN (dense)

2.1 Features of CNN

(a) Parameter / weight sharing

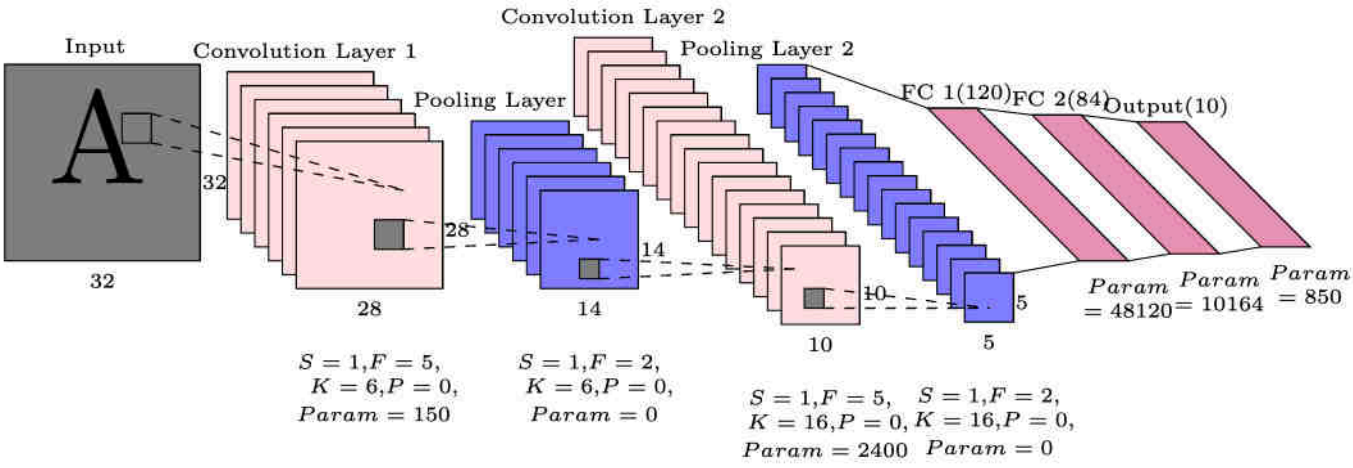
- kernel same as it passes over image (tries to detect same features over entire image)



(b) Sparsity of connections

- unlike fully connected NN, only a few input pixels contribute to every hidden node

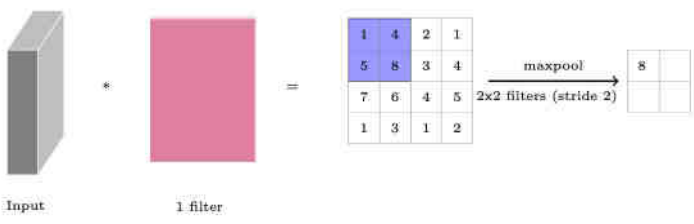
2.1 Full CNN

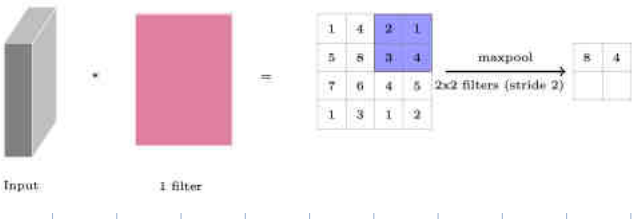


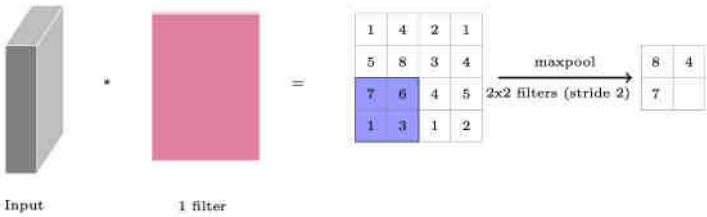
- Alternate convolution and pooling layers

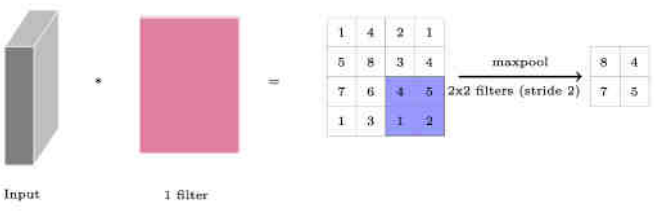
2.2 Pooling Layer

- Pool function applied on a sub-matrix and returns a single value
- Eg: maxpool, minpool, average pooling

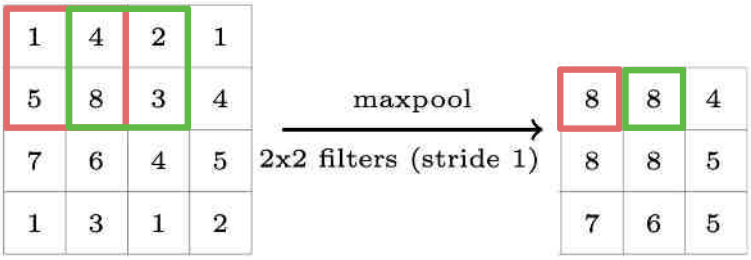








- with pool stride $S = 1$ and $P_s = 2 \times 2$



- Output size $= \frac{I - P_s}{S} + 1$

  (I: input size (4×4); P_s: pool size (2×2))

- Pooling gives translation invariance (small changes in location of a feature are detected in pooled feature map)
- Pooling layers are not trained

Q: Given CNN layer with 6 filters, $S = 1$, $P = 0$, $W_1 = 32$, $H_1 = 32$, $D_1 = 1$, $F = 5$

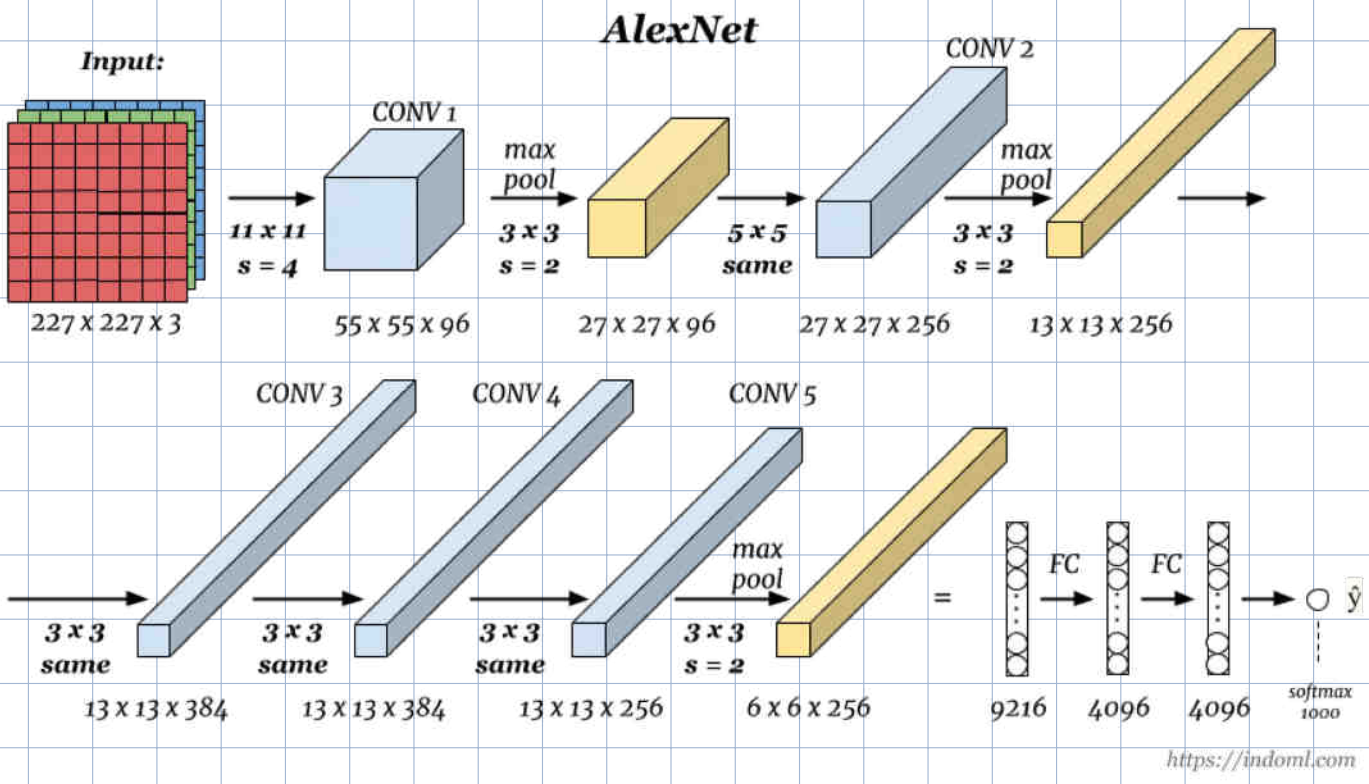
Find W_2, H_2, D_2.

$$W_2 = \frac{W_1 - F + 2P}{S} + 1 = 27 + 1 = 28$$

$$H_2 = \frac{H_1 - F + 2P}{S} + 1 = 27 + 1 = 28$$

$$D_2 = K = 6$$

3. AlexNet



- Exercise: work out dimensions of output from each layer and verify

4. Calculate no. of Parameters

W_c = no. of weights in conv layer

B_c = no. of biases in conv layer

$K \times K \times D$ = size of kernel / filter

N = no. of filters / kernels

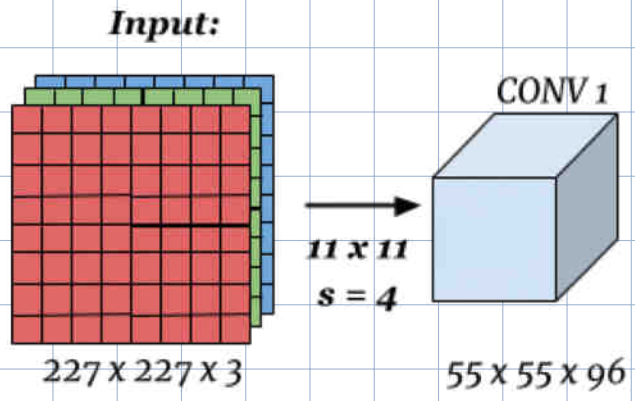
$$W_c = K^2 \times N \times D$$

$$B_c = N$$

$\therefore$ total no. of parameters

$$P_c = W_c + B_c$$

Q: Calculate no. of parameters in CONV1



$$W_c = 11 \times 11 \times 3 \times 96 = 34848$$

$$B_c = 96$$

$$P_c = 34944 \text{ parameters}$$

AlexNet no. of Parameters

Layer Name	Tensor Size	Weights	Biases	Parameters
Input Image	227x227x3	0	0	0
Conv-1	55x55x96	34,848	96	34,944
MaxPool-1	27x27x96	0	0	0
Conv-2	27x27x256	614,400	256	614,656
MaxPool-2	13x13x256	0	0	0
Conv-3	13x13x384	884,736	384	885,120
Conv-4	13x13x384	1,327,104	384	1,327,488
Conv-5	13x13x256	884,736	256	884,992
MaxPool-3	6x6x256	0	0	0
FC-1	4096×1	37,748,736	4,096	37,752,832
FC-2	4096×1	16,777,216	4,096	16,781,312
FC-3	1000×1	4,096,000	1,000	4,097,000
Output	1000×1	0	0	0
Total				**62,378,344**

5. Batch Normalization

- Normalization (over entire dataset)

$$z^N = \frac{z - \mu}{\sigma}$$

- Batch normalization between layers of NN

$$z^N = \frac{z - \mu_z}{\sigma_z}$$

μ_z : mean of neuron's output

σ_z : standard dev of neuron's output

- With batch norm, no bias (mean subtracted)

Without Normalization

$z = g(w, x) + b$ ← linear function

$a = f(z)$ ← activation

With normalization

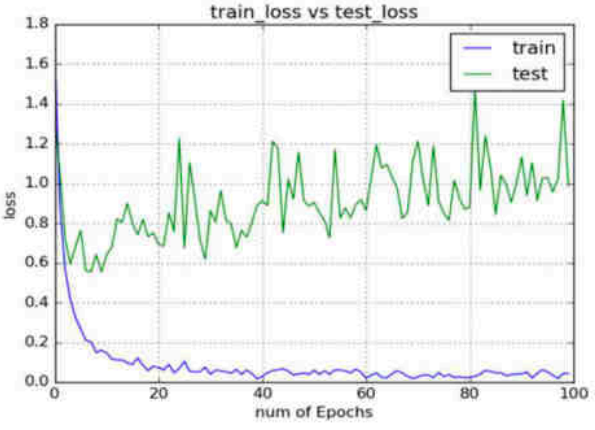
$$z = g(w, x)$$

$$z^N = \left(\frac{z - \mu_z}{\sigma_z}\right) \cdot \gamma + \beta$$

$$a = f(z^N)$$

- γ, β are learning parameters
  - γ: std dev
  - β: mean
- Loss with & without BN

without BN



with BN