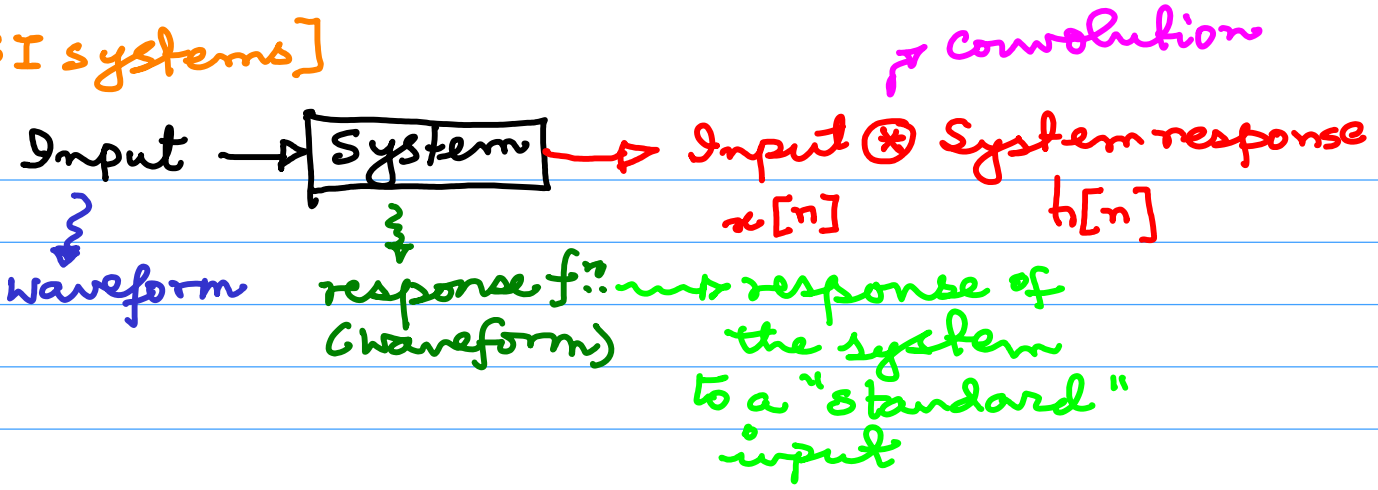
[LSI systems]



Motivation for CNNs (Simple Deep NNs)

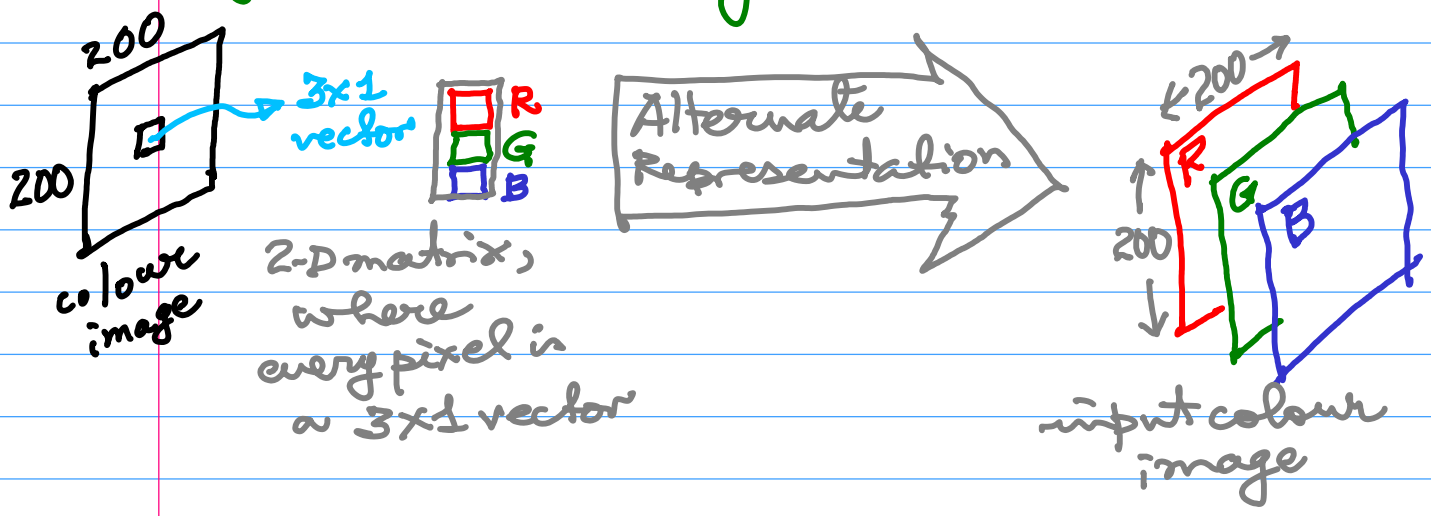
⊛ Feedforward MLP: Too many connections (each neuron to every neuron: between layers)

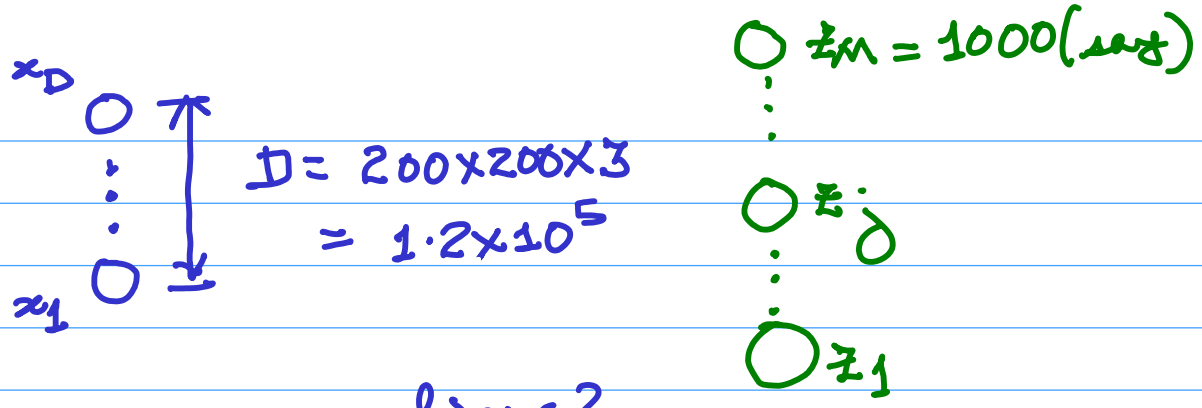
⊛ Images/Videos (2-D/3-D): inputs

→ vectorise them → discards spatial connectivity/neighbourhood information

→ Need for a structure which can naturally account for peculiarities of 2-D inputs/3-D inputs/... with a 'reasonable' complexity.

Images (2-D) are very common → LeNet (1989)





How many connections?

Each hidden layer neuron z_i is connected to 1.2×10^5 neurons

$\equiv 1.2 \times 10^5$ weights + 1 bias term

Even if we forget the bias term,

# of parameters $\simeq 1.2 \times 10^5 \times 10^3$ weights

$= 1.2 \times 10^8$ weights.

→ Too many parameters, chances of overfitting

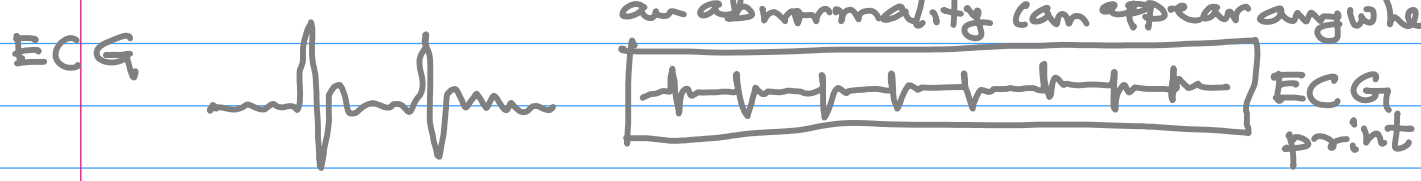
(*) **Invariance Issues**: we often look for certain patterns in 1-D inputs (e.g., doctors looking at ECGs) or 2-D inputs (e.g., doctors looking at X-ray images or USG)

These patterns can be anywhere in the 1-D waveform / input / signal, or a 2-D image

Simplest Invariance issue: **translation**

Others: rotations, scaled, shear, projective effects (2-D)

["Shift Invariance"] → comes naturally



Euclidean transform (Simplest possible)

2-D transformations

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

$$\underline{x}' = R\underline{x} + \underline{t}$$

R → rotation

$\underline{t}$ → translation

⇩

resilience to translated patterns of interest in 1-D / 2-D / ... inputs is usually built into the CNN structure itself

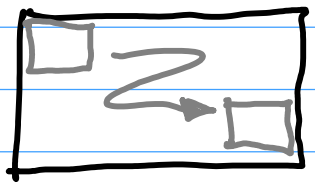
(for other transformations → heuristics e.g.

- feed transformed variants into the NN & hope that it learns them
- Regularisation (penalise a NN which does'nt account for these)
- Pre-processing the input, & feeding the processed input to the NN)

2-D example: "Where's Wally / Where's Waldo"

☐ "Wally/waldo detector" → sweep the entire image with this detector (correlation score over a square patch of the input image)

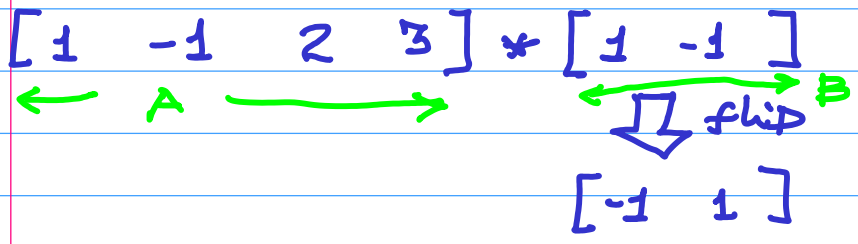
→ how likely is Wally/Waldo present in this part of the image



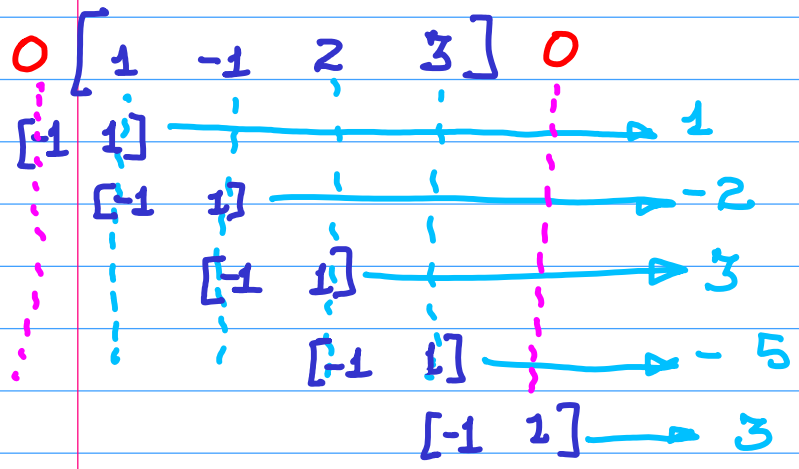
→ Convolution / shift invariance

Usually, we consider two forms of convolution:

(1) "Wherever fits"



Also called "Zero padding"



longer than the

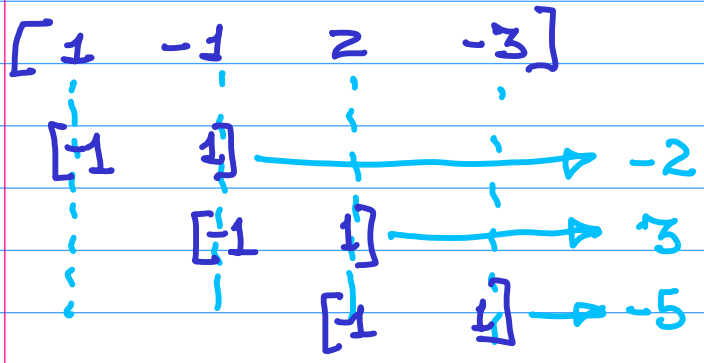
[1 -2 3 -5 3]

larger array (input)

size = A + B - 1

= 4 + 2 - 1 = 5

(2) "Perfect fit" (Generally used in NNs / image processing)



Ans: A - B + 1

[-2 3 -5]

shorter than the larger array

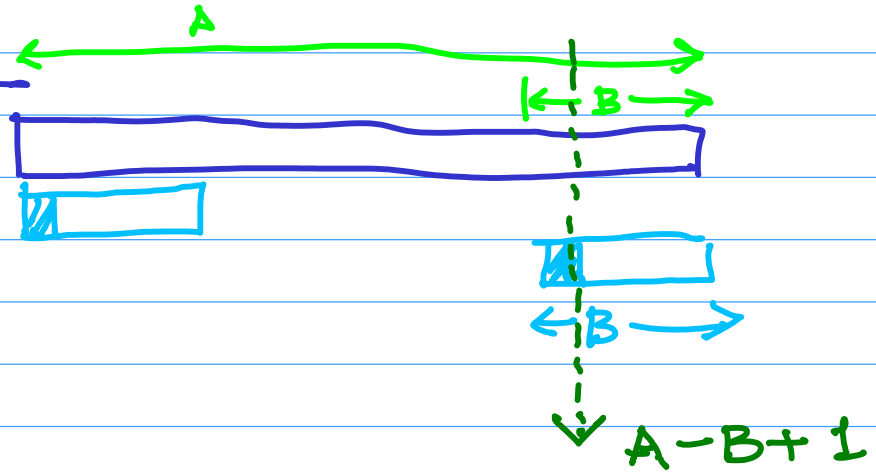
logic → the previous 'wherever fits' strategy has (B-1) more overlaps to the left & (B-1) more to the right

perfect fit + (B-1) + (B-1) = A + B - 1

$\Rightarrow$ perfect fit = A + B - 1 - 2B + 2

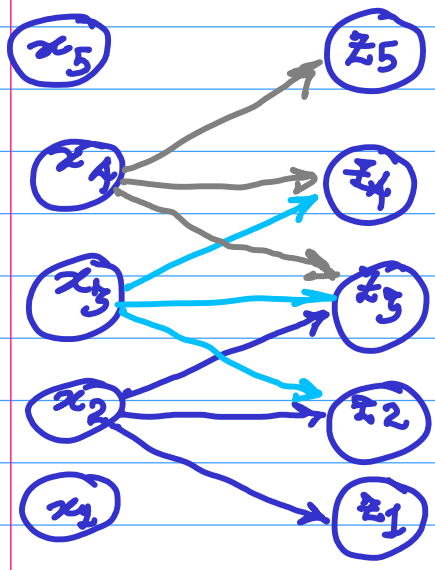
= A - B + 1

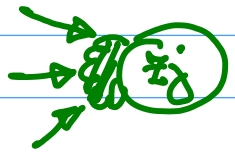
Alter: —



Characteristics of Deep Networks

(1) Local Receptive fields (Sparse connections)





"receptive field of z_i"

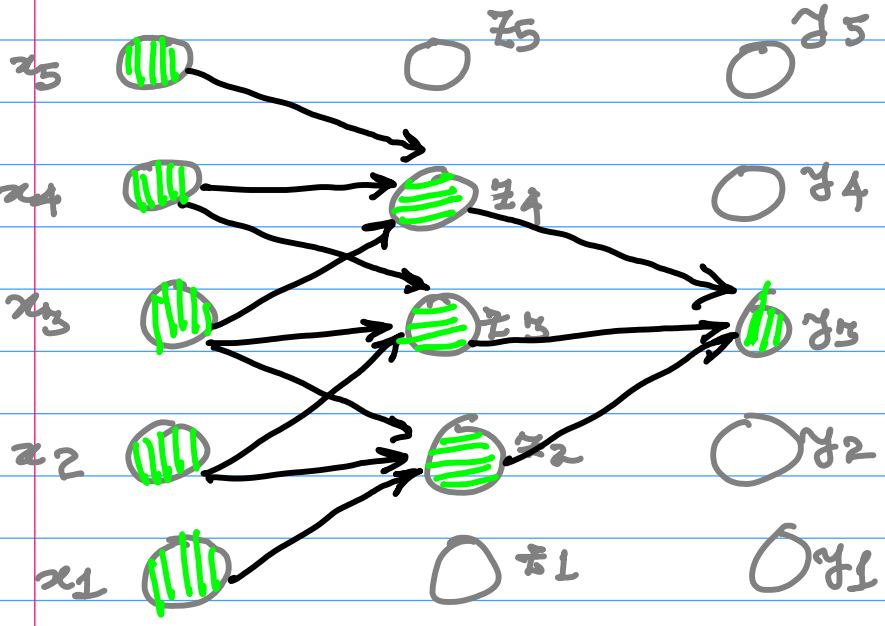
Complexity: connectivity is now $\mathcal{O}(M)$

M: # of neurons in the hidden layer

in place of the general MLP case of $\mathcal{O}(DM)$

D: # of neurons in the input layer

* The sparsity is not really that restrictive!

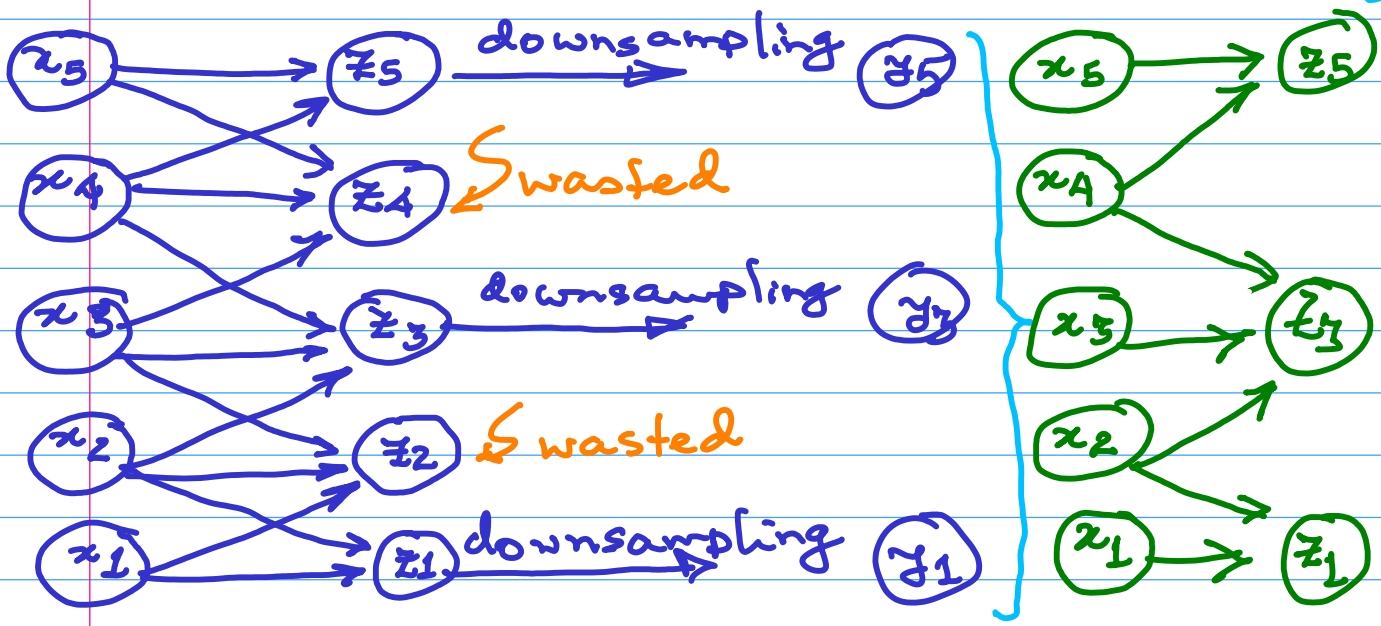


y_3 interacts indirectly with 5 in the input layer!

→ Units in deeper layers interact indirectly with a larger portion of the input

* Efficient representation of complicated interactions among many variables, by using simple building blocks which have sparse interactions.

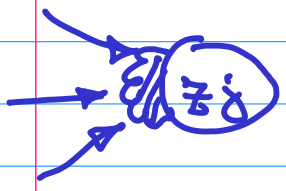
(2) 'Strided' Convolutions

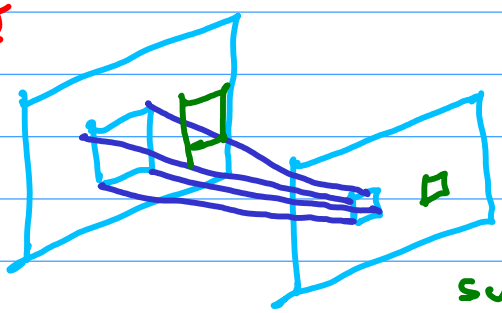


'stride' = 2 in this case

mathematically equivalent but less wasteful

(3) Parameter Sharing

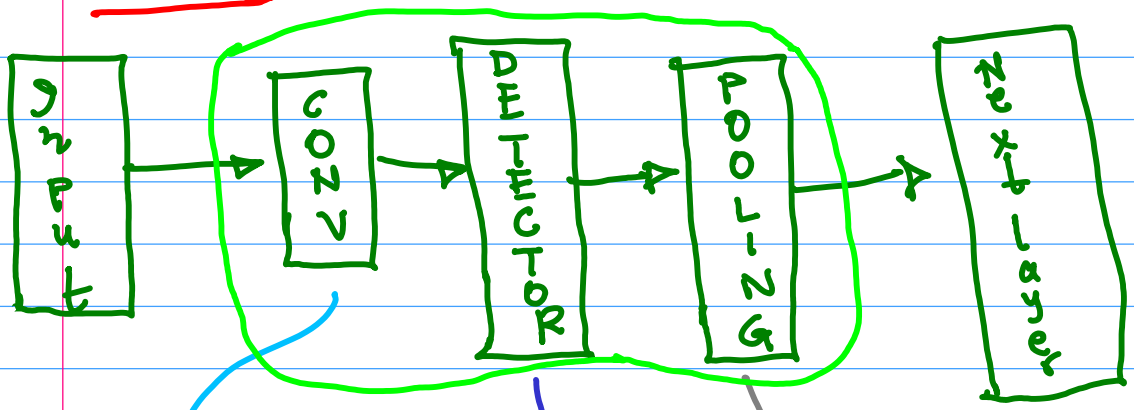




"tied weights"
→ use them for all mappings of substructures in the first layer to the second layer

(4) Pooling

"one convolutional layer"



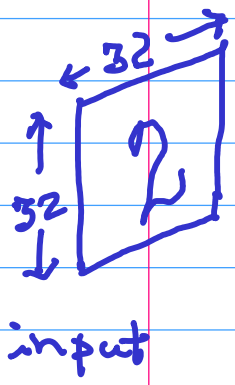
convolutional stage with linear activations

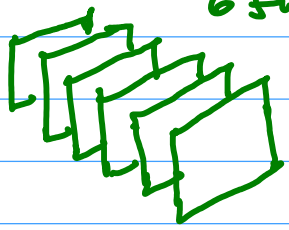
non-linearity: activation f'ns e.g., ReLU

to modify the output of this layer, further (some form of subsampling — MaxPooling / Average pooling (historical)

LeNet-5 (1989) LeCun, Bottou, Bengio, Haffner

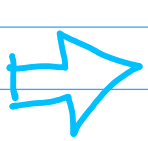
(handwritten & machine printed digit recognition)

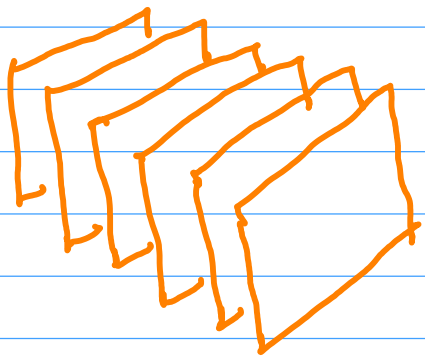




6 filters/ kernels/ masks, each of size 5x5

Use the 'perfect fit' model of convolution





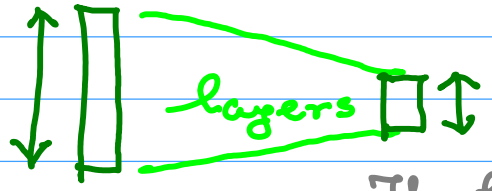
each is a 28x28 image

Important philosophy: this is one part of a 'contractive structure' (the other being pooling)

Why contractive?

large-sized input

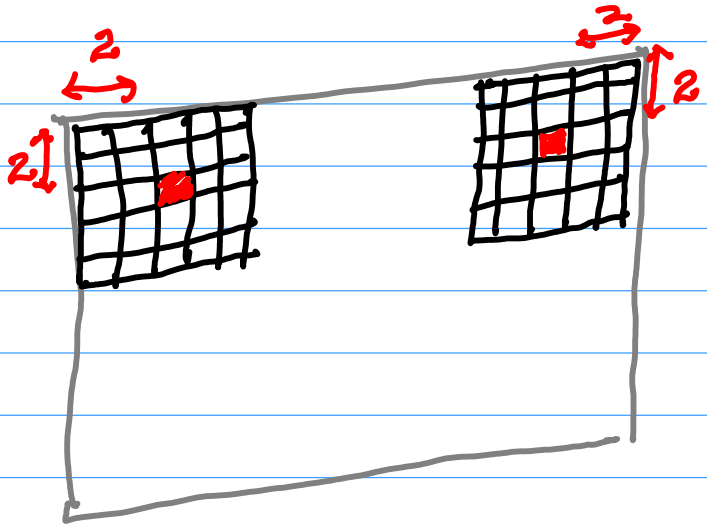
e.g., 32×32 image = 1024×1 input

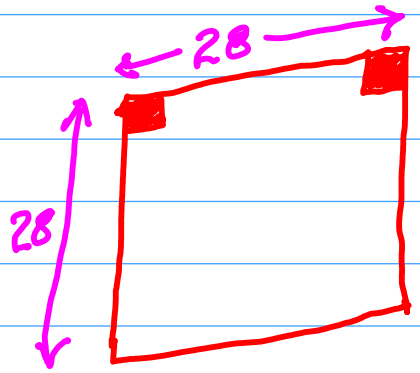


The final classifier is a 10-class classifier

The 'perfect fit' convolution, as opposed to the 'full' convolution: → A−B+1

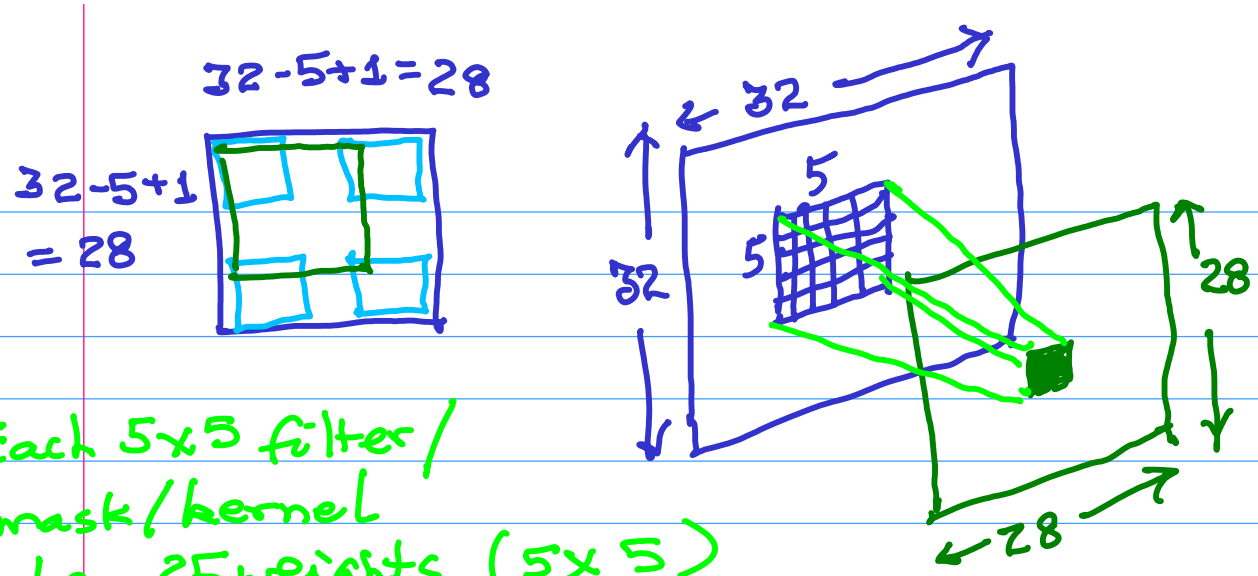
→ A+B−1



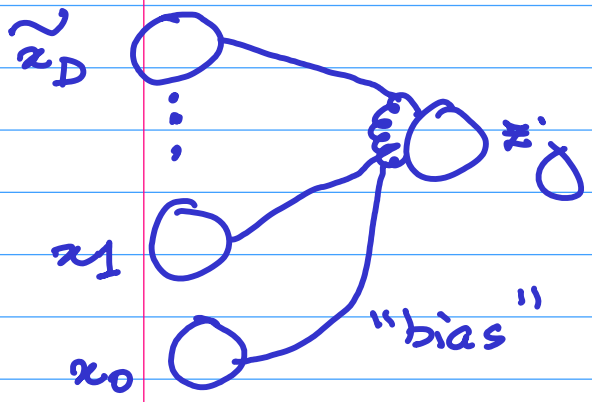


why 28? 32−2−2 = 28 (32−5+1)

The resultant of the 'perfect fit' convolution is of a smaller size.



Each 5x5 filter/ mask/kernel has 25 weights (5x5) + 1 bias term (also counted as a connection)



$$a_j = \sum_{i=1}^{\tilde{x}_D} w_{ji} x_i + w_{j0} \text{ or } b_j$$

25 weights + bias

$\tilde{x}_D$ is a small fraction of the total x_D

parameter: in each of the repeating structures

One for each of the 6 filters

parameters = $6 \times (5 \times 5 + 1) = 6 \times 26 = 156$

connections: $6 \times (28 \times 28) \times (5 \times 5 + 1) = 122,304$