Problems : - Too many weights

- Need features

Solution : Convolutional Neural Networks

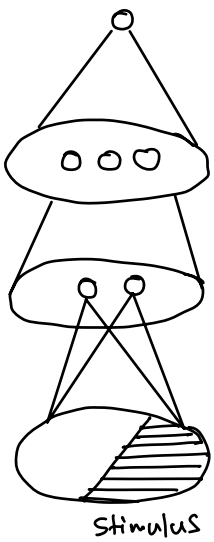
① 1959 - Hubel and Wiesel - [Cats vision]

* Simple and complex cells in the visual/primary cortex a cascaded model for 2 types of cells.

② 1979 - Kunihiko Fukushima - [Neo cognition for digital recognition]

③ 1989 - LeCun et.al - [CNN]

① Hubel and wiesel

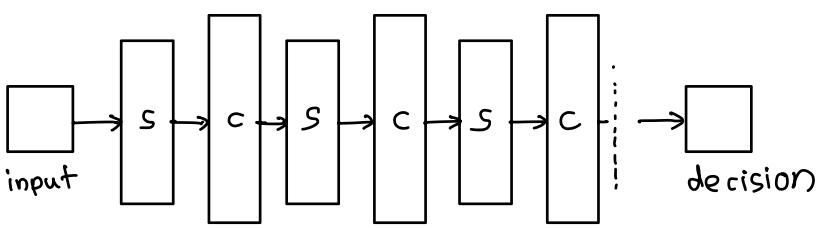


Highly complex cells — square — need low level info

complex cells — line

simple cells — white black — rough features

② Fukushima



	MLPs	CNNs
input	feature vectors	Data(raw)
connection	dense	sparse
weights	independent	share
learning	backprop	backprop
filtering	outside	inside
scheme	directly learn a non linearly separable problem via features of input	learn a (quasi) linear seperable problem via divide and conquering the raw input

Observations / Motivations

— MLPs/AEs have too many weights

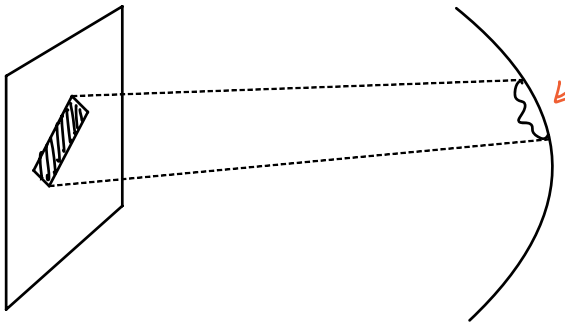
digit recognition: 30 × 30 = 900 inputs ⟶ first layer has 80,000 weights

face recog.: 256 × 256 = 65536 inputs ⟶ 1st layer has 600,000 weights

lots of inputs because full/dense connectivity

Idea — weight sharing

How should we then get enough information (features) from the input?



Hubel-Wiesel discovered "receptive fields"

Specific sensivity towards specific patterns in small fields of (cat's) visual context.

↓

simulate in convolution

What is convolution ?

In a physical system : An input acts on a linear system to produce an output.

$$y(t) = x(t) * h(t) = \int_{-\infty}^{+\infty} x(\tau)\, h(t-\tau)\, d\tau$$

$y(t)$: output; $x(t)$: input; $h(t)$: unit impulse-response of the system; τ: dummy variable

Convolutional sum : $y(n) = \sum_{k \in N} x(k) \cdot h(n-k)$ ← specific field

$$h = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}_{3\times3}$$ Convolutional kernal (mask)

vertical impulse

$$X = \begin{bmatrix} \overset{3}{\boxed{A}}_3 & \overset{3}{\boxed{B}}_3 \\ & \end{bmatrix}_{256\times256}$$

$$A = \begin{bmatrix} 10 & 50 & 100 \\ 10 & 50 & 100 \\ 10 & 50 & 100 \end{bmatrix} * h$$

response of filter → 270

$$B = \begin{bmatrix} 10 & 10 & 10 \\ 50 & 50 & 50 \\ 100 & 100 & 100 \end{bmatrix} * h$$

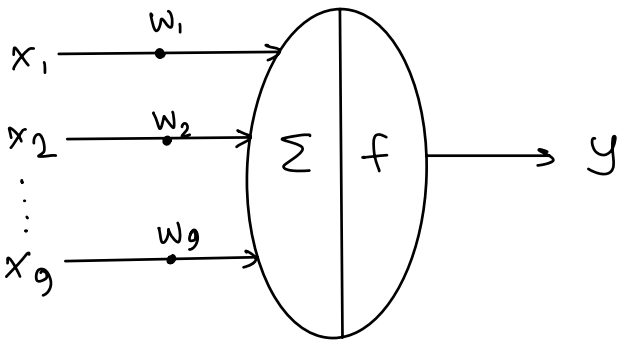
I see something

rof = 0

I dont see anything

Idea : weights as filter coefficients

$$\begin{bmatrix} x_1 & x_4 & x_7 \\ x_2 & x_5 & x_8 \\ x_3 & x_6 & x_9 \end{bmatrix} * \begin{bmatrix} w_1 & w_4 & w_7 \\ w_2 & w_5 & w_8 \\ w_3 & w_6 & w_9 \end{bmatrix}$$

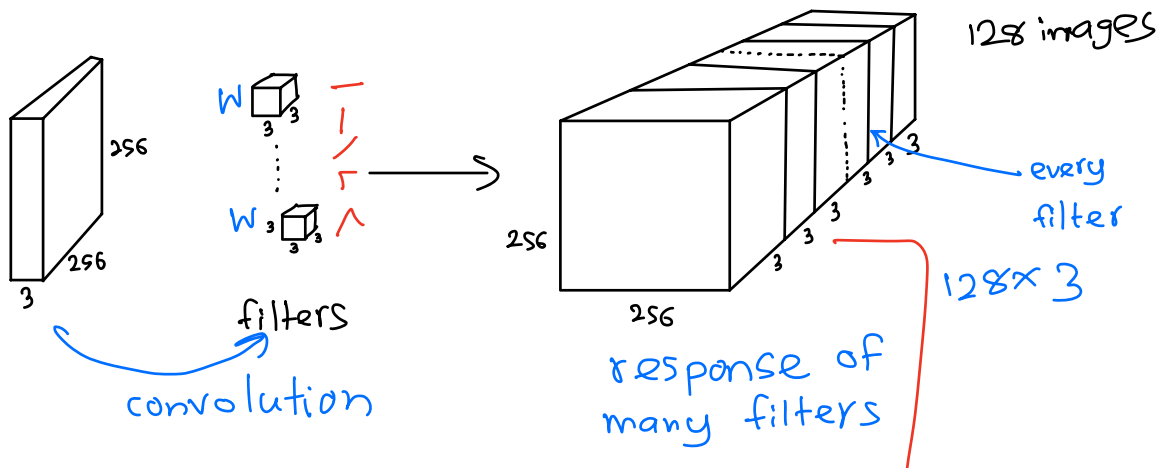


eg: (1000x1000)

Weight sharing : One filter for all pixels

Problem : But we need many features

Solution : Learn several filters (32, 64, 128....)

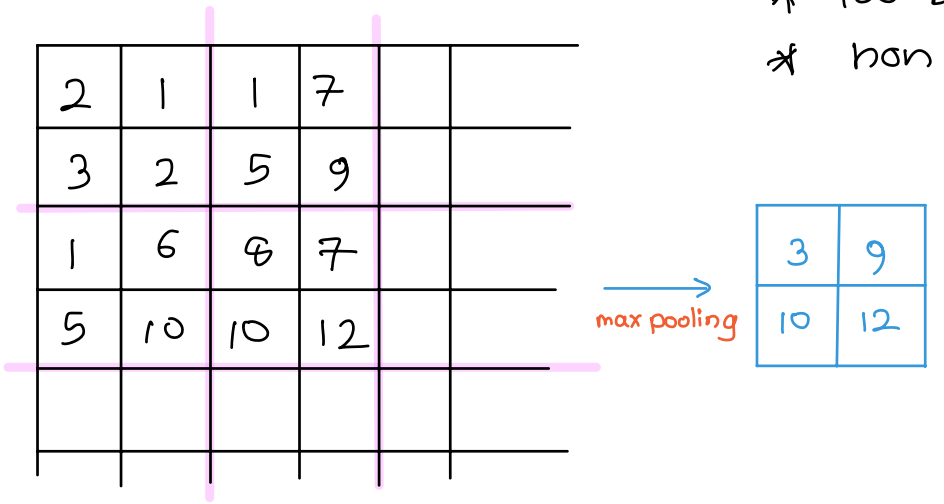


→ Downsampling (pooling)

Convolution is not exactly divide to enable "conquer". So divide it

↓ MLP (still difficult)

* too big
* non linear

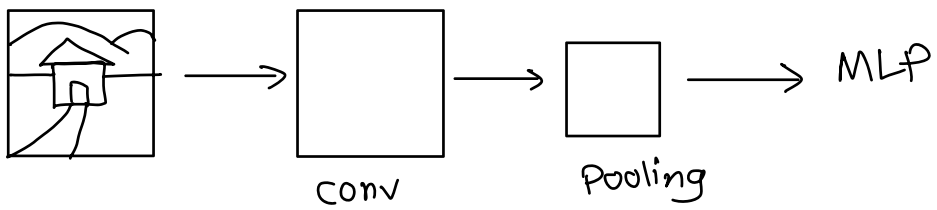


convolution ⟶ get features

pooling ⟶ reduce dimentionality

MLP ⟶ classify

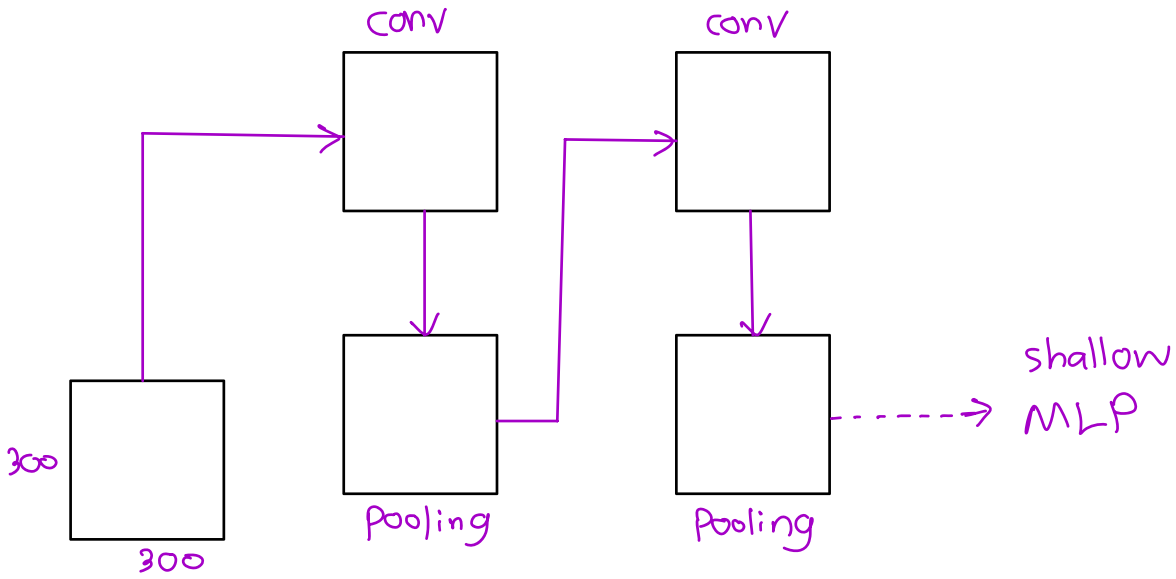
So far we have:



Problem: still non linear hence impossible for shallow MLPs.

Solution: Hubel/Wiesel ⟶ Cascade it

[Neocognitron did it]



Some common Nets :

AlexNet (2012)

ZF Net (2013)

Google Net (2014)

VGG (2014)

Resnet (2015)