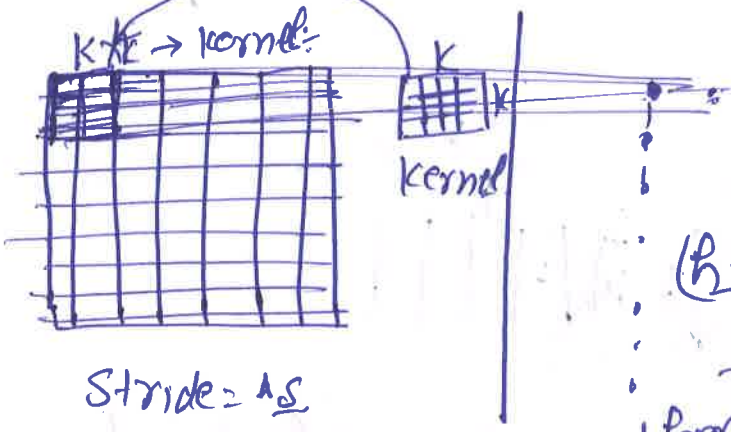
COL865
Deep Learning
Convolutional Neural Networks (CNNs)

First Real Architecture -

At the heart :- convolutional layer

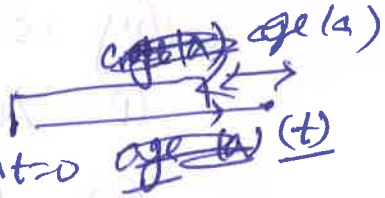


What is a kernel?:-

Inspired by the convolution operation:-

$$s(t) = \int_a x(a)\, w(t-a)\, da$$

x → input
w → kernel

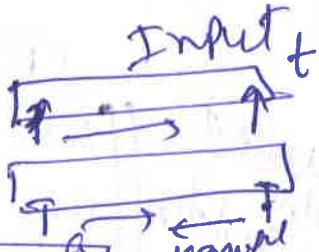


:- Two Real valued functions:-

$x: R \to R$

$w: R \to R$

Feature map



$$s(t) = (x * w)(t) = (w * x)(t)$$

For discrete case:- → Range:- all possible values where x is defined

$$s(t) = \sum_{a=-\infty}^{\infty} x(a)\, w(t-a)$$

$X: I \to R$
$W: I \to R$

$$S(t) = \sum_a X(t-a)\, K(a)$$

Flipped kernel

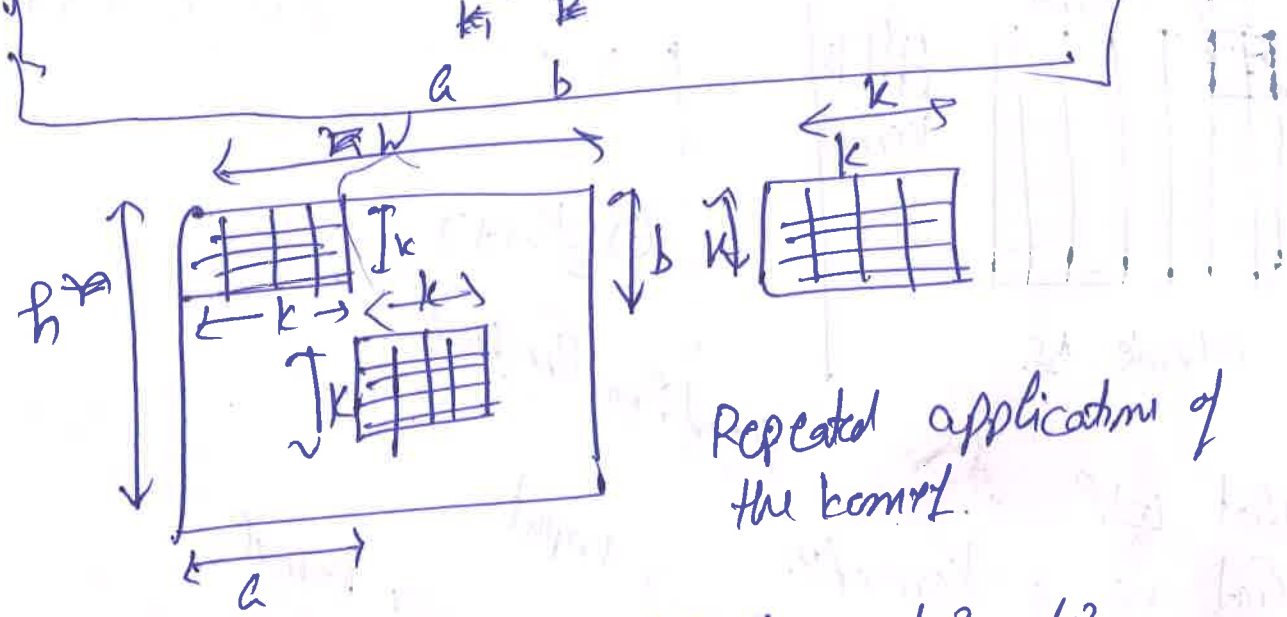
Flipped kernel

$\Rightarrow$

$$x(i) = \sum_{a} x(i+a)\, w(a)$$

Two-dimensional:-

$$x^{(d+1)}(i,j) = \sum_{a}\sum_{b} x^{(d)}(i+a, j+b)\, w(a,b)$$



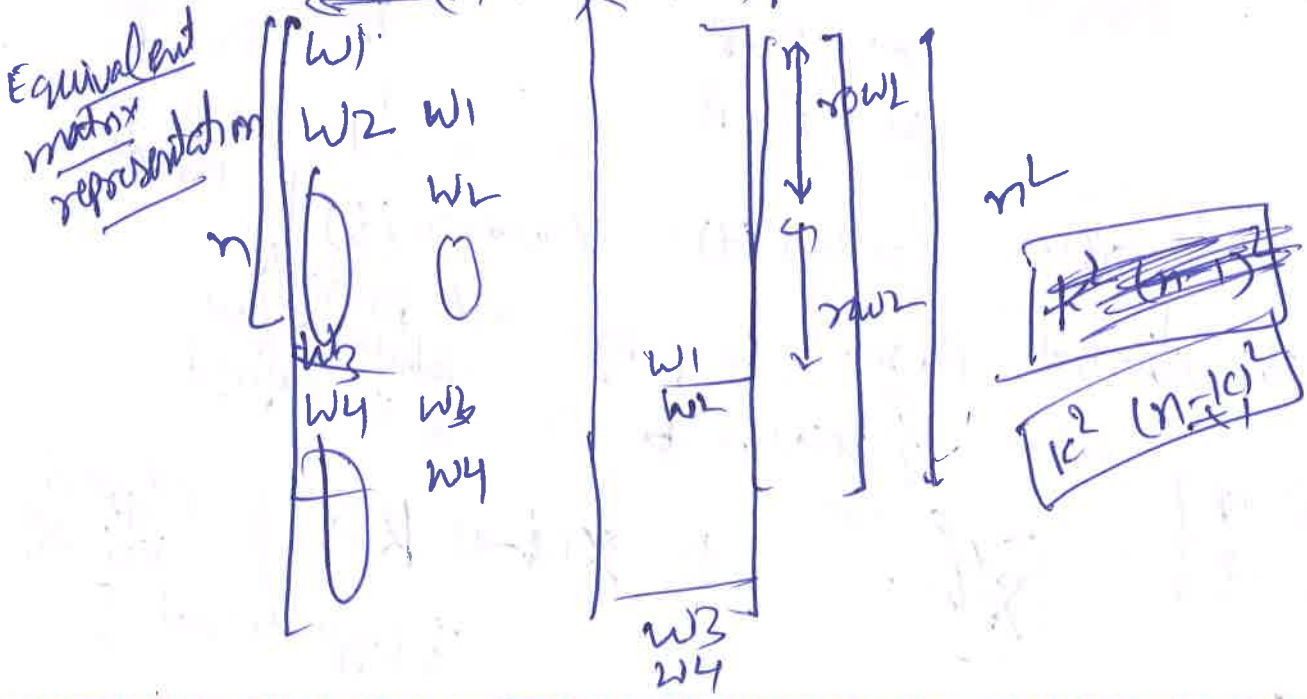
Repeated application of the kernel.

Why is this a useful thing to do?

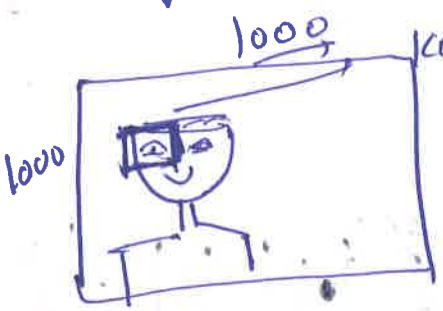
① Localized operation

② Significantly less number of parameters (parameter tying).

$$\begin{bmatrix} w_1 & w_2 \\ w_3 & w_4 \end{bmatrix}$$



② Why this Operation? Exploit spatial locality



connect to recognize eyes

MOTIVATION

CNNs

1000 hidden units

1 billion parameters

∴ To much to handle

Exploit:- ① Spatial arrangement (locality)

② Compositionality

→ Recognize Face

↳ Recognize Parts

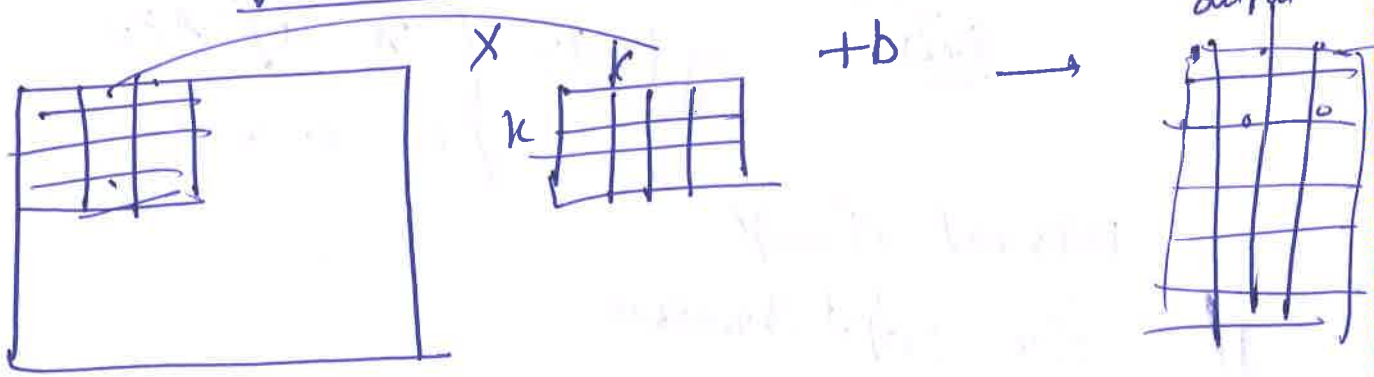
↳ Recognize contours

↳ Recognize edges

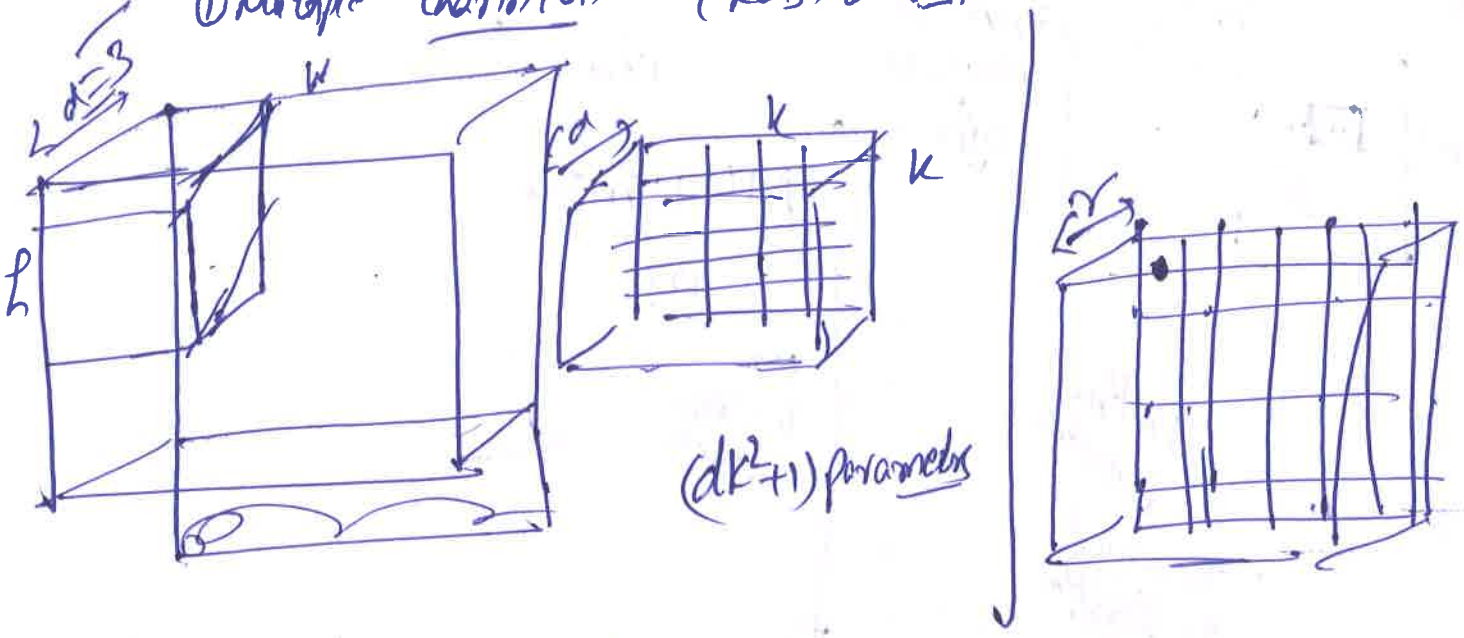
Multiple Levels of hierarchy

Instead of full matrix multiplication, use sparse representation

Idea of kernels:-



① Multiple kernels:- (RGB) values.



But- we need to recognize multiple such "objects"

⇒ multiple kernels: γ

Kernel "Matrix" :- Tensor

$$X'_{i,j,l'} = \left[\sum_{a,b,l} X_{i+a,j+b,l} \; K_{a,b,l,l'}\right] + b_{l'}$$

Tensor

Followed by non-linearity.

} Derive the formula when using strides $s \neq 1$

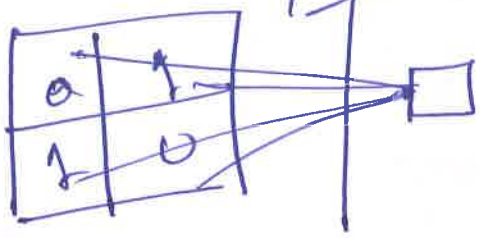
ReLU: $g(x) = \begin{cases} x & \text{if } x \geq 0 \\ 0 & \text{o.w} \end{cases}$

Does not change the output dimension

③ (Max):- Pooling Operation

:- Applied after convolution:-

Pool:-



{ Helps to reduce the size of the representation

$x'(i,j) =$ $x(i,j) \Rightarrow$ [Avg]

Max $x(i+a, j+b)$
a,b
$a \in \{0,1\}$
$b \in \{0,1\}$

Three steps:-

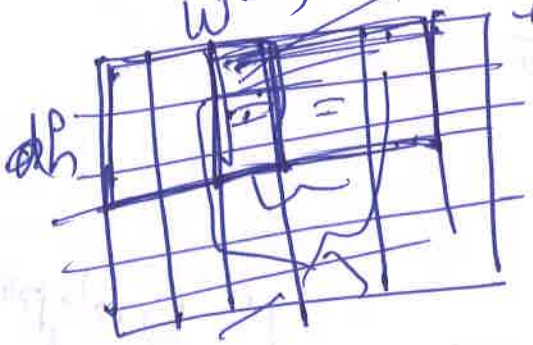
① Linear kernel application

② Non-linear transformation

③ Pooling

why

W → → Do not really care about exact location



⇒ can use with a stride $S \neq 1$.

Another important idea:- zero-padding

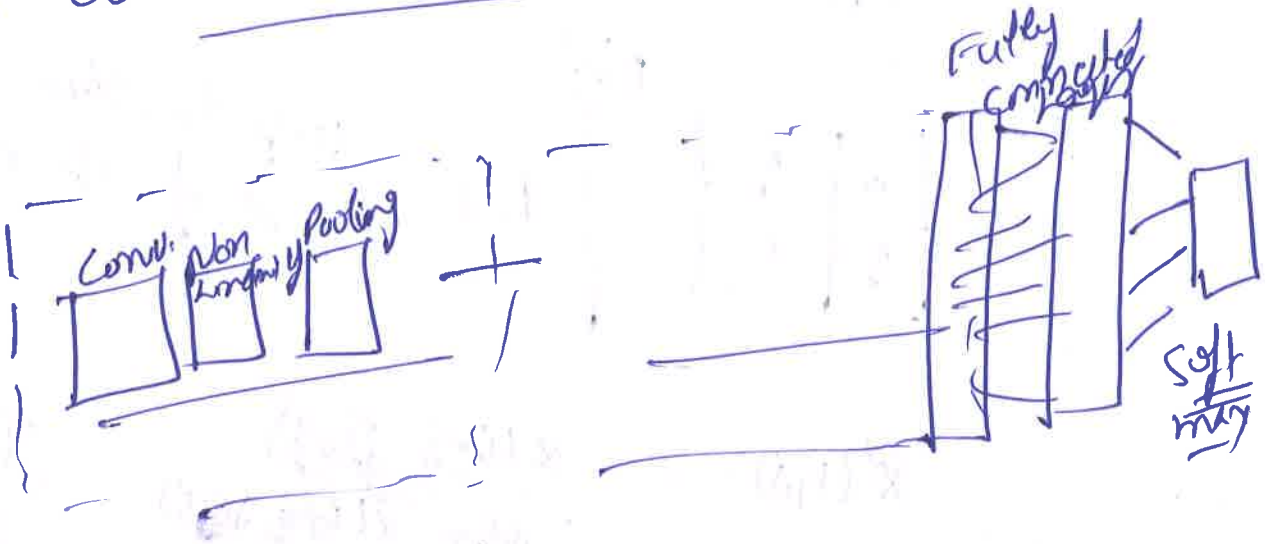
Output Layer (shrinks) $(W-K+1) * (H-K+1)$

⇒ Pad zeroes:- $(K-1)$ zeroes on each side.

Add more zeroes than $(K-1)$ - "Fuller" convolution

Convolutions:-

Overall Architecture:-



Alexnet. (2012)

ILSVRC Imagenet challenge

Train:- 12M

Test:- 150,000

1000 labels

Top 5. error rate:-

26.2% to 15.4%

Backpropagation read from the book

Notion of equivariance:

But not to scaling & rotation

f is equivariant w.r.t g

if f(g(x)) = g(f(x))

convolution (f) is equivariant w.r.t translation (g).