Edge Detection

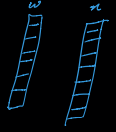
Convolution

→ can be applied on an image to get specific effects on image.

↳ element wise * & then +

(same stuff happens here.

for simple neuron

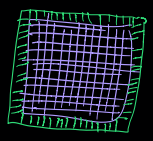


This can be seen as a special case of convolution.

input $N \times N$ → $\xrightarrow{k \times k \text{ conv}}$ output $(n-k+1) \times (n-k+1)$

$N \times N$ → Padding = 1 → $(N+2) \times (N \times 2)$

that means



Zero padding → fill padding with zero

Same value padding → not used that much.

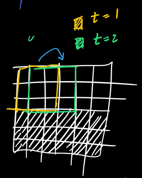
↳ take the value of neighbouring pixel.

↳ there is confusion sometimes → which value to take from surrounding 6 pixels ?

$$N \times N \xrightarrow[\text{pad} = P]{K \times K \text{ conv}} (N - K + 1 + 2P) \times (N - K + 1 + 2P)$$

→ kernels are typically square matrices for CV apps.

Strides



stride is 1 over here. it can vary depending on the application.

→ not just horizontal, stride also applies for vertical.

→ They help us reduce the size of the matrix.

$$N \times N \xrightarrow[K \times K \text{ conv}]{\text{stride} = S} \left(\frac{N-K}{S} + 1\right) \times \left(\frac{N-K}{S} + 1\right)$$

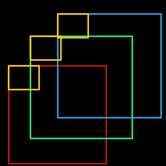
→ if fraction, take floor fⁿ

$$N \times N \xrightarrow[K \times K \text{ conv}]{\text{pad} = P,\ \text{stride} = S} \left(\frac{N-K+2P}{S} + 1\right) \times \left(\frac{N-K+2P}{S} + 1\right)$$

conv, pad, stride

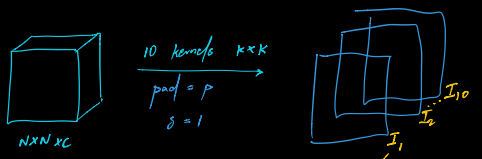
→ used to manipulate images

Convolution on RGB

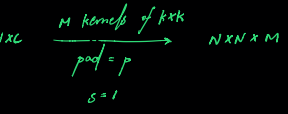


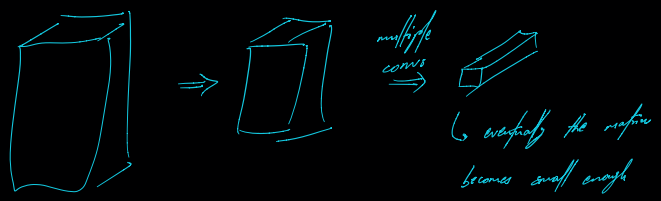
The same concept applies, like a 2D conv.

channels 3 k ... etc.



stayed same because of sufficient padding.





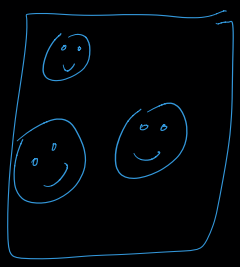
↳ eventually the region becomes small enough for dense connections. So we flatten it.

location invariance

Scale invariance

← Rotation invariance

CNN cannot do this.



Detect something even if the location is different or the scale or rotation

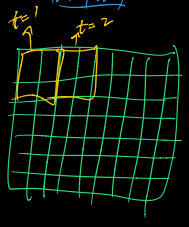
is different

max pooling

provides a way of

achieving some location invariance

→ take max →



if max pool
k=2
then stride also 2

high activation value.

if we have an edge kernel

it can be in ① ② ③ or ④

it will still be detected.

some location invariance.

And if some kernel has learnt

to activate after seeing an "ice cream",

That kernel is going to run over

the entire image anyway. If ice

cream is present in image, it will

be found!!

→ CNN by definition are location invariant this way.

They also have average pooling, in case we want to preserve some data. max pooling takes highest and throws the

2 0 1 2 ...

way of data away.

↳ In medical applications we want to preserve as much data as possible

Training a CNN

$x \xrightarrow{w} y \rightarrow L(y, \hat{y})$

As long as $\frac{\partial L}{\partial w} \rightarrow$ anything is possible

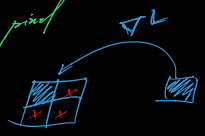
element wise $\odot$ then $+$

⇓

just like dot product

Derivative of max pooling

↳ compute derivative but only pass it to the max pixel that was chosen



for max value → derivative = 1 → derivative with itself is one.

else 0

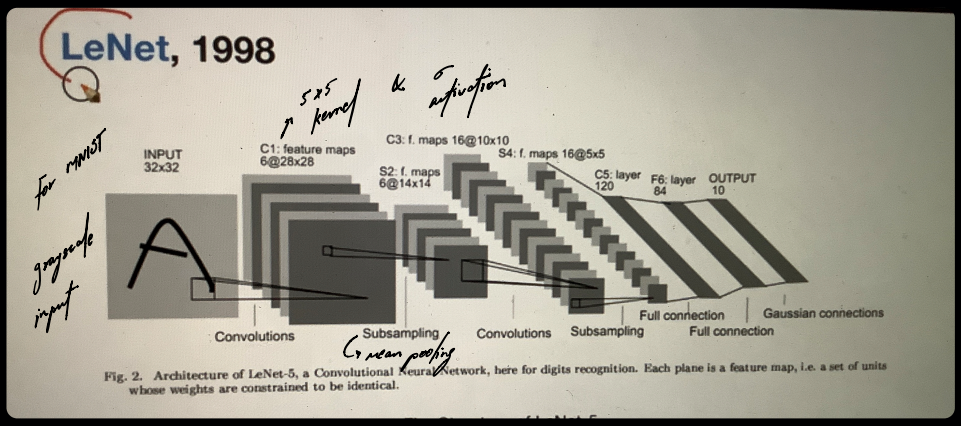


Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.