CNN

↳ allows deep networks to learn func on structured spatial data - image, video, text

↳ CNN learns to exploit the natural covariance structure in order to learn effectively.

Local Receptive field.

↳ receptive field of a neuron is part of body's sensory perception that affects neuron's firing.

↳ Neurons have a certain 'field of view' as they process sensory i/p that the brain sees. → local receptive field.

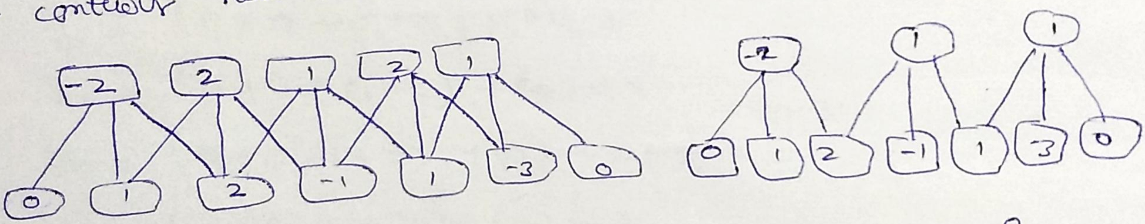
↳ a layer of such neuron → convolutional layer

↳ this layer can be viewed as transformation of one spatial region into another

↳ C.L applies non-linear func to a local receptive field in its input.

What if we want to specify that local receptive fields should not overlap? → stride size

It controls how the receptive field is moved over the i/p.



what if we want more than one grid of no. o/p?

↳ we add more convolutional kernels

collection of convolutional kernels → convolutional layer

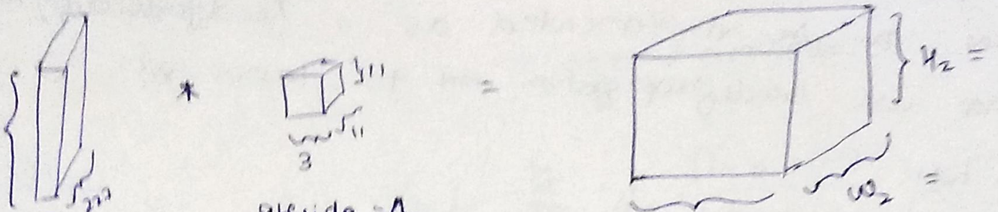
Pooling Layer

↳ Instead of learnable transformation, it's possible to instead use a fixed nonlinear transformation.

↳ to reduce computational cost

• we can have many such kernels, but the kernels will be shared by all locations in the image

Wt. sharing



stride = 4
padding = 0
filters = 96

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

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

P = padding
S = stride
F = filter spatial extent (11 × 11)

D_2 = depth of o/p = no. of filters = 96 = K

$$W_2 = \frac{227 - 96}{4} + 1 = 55$$

$$H_2 = \frac{227 - 96}{4} + 1 = 55$$

~~parameter = wt of the filter~~
~~= (No. of filter) × (filter height × filter width)~~
~~= (K) × (F × F)~~
~~= (K~~
~~No parameter in pooling layer.~~

- parameters = (spatial extent) × (kernels) × (depth of i/p)

↳ there is no parameter in pooling layer

↳ fully connected layer has largest no. of parameters

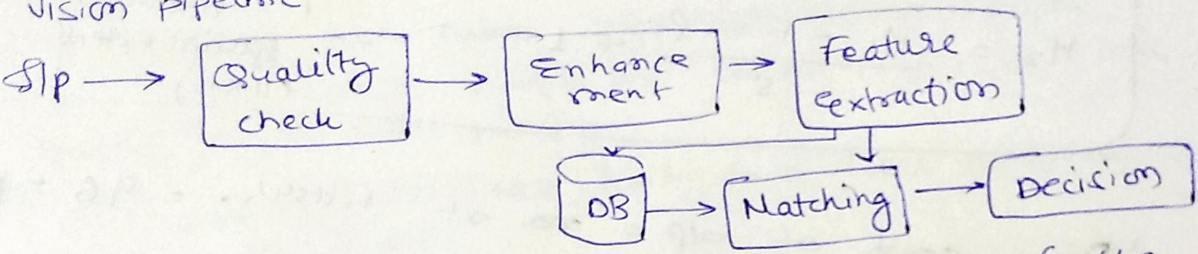
How do we train CNN.

↳ A CNN can be implemented as a feedforward NN

We also use backpropagation NN to learn wt

- Shallow NN:
  - ↳ which have small no. of layers
  - ↳ usually a single hidden layer
  - ↳ optimization is difficult for Non-convex, non-linear
  - ↳ freezing some deep layers makes it shallow

- Vision pipeline



Higher (deeper) layer represents abstraction of the features

- $$S_t = \sum_{i=0}^{\infty} x_{t-i} w_i$$

$$S_{ij} = (I * K)_{ij} = \sum_{a=\lfloor -m/2 \rfloor}^{\lfloor m/2 \rfloor} \sum_{b=\lfloor -n/2 \rfloor}^{\lfloor n/2 \rfloor} I_{i-a, j-b} K_{m/2+a, n/2+a}$$

- Cross-entropy loss func: classification problem

MSE works better for regression problem*

$$\boxed{L = -\frac{1}{C} \sum_{i=1}^{C} y_i \log \hat{y}_i}$$

binary case

$$= -y_i \log \hat{y}_i + (1-y_i) \log (1-\hat{y}_i)$$

which measures probability distribution on the o/p ~~layer~~ labels in the ground truth vs i/p on the o/p by NN.

$\hat{y}_i$ = predicted value = $\sigma(z)$

$\sigma(x) = \dfrac{1}{1+e^{-x}}$ → sigmoid activation

$$\frac{\partial L}{\partial w_j} = -\frac{1}{n} \sum_x \left(\frac{y}{\sigma(z)} + \frac{(1-y)(-1)}{1-\sigma(z)} \right) \underbrace{\frac{\partial \sigma}{\partial w_j}}_{\sigma'(z) x_j}$$

$$=$$

$$\left(\frac{\partial \sigma}{\partial z}\right) \frac{\partial z}{\partial w_j} \Rightarrow \sigma'(z) x_j$$

$$= \frac{1}{n} \sum_x \frac{\cancel{\sigma'(z)} x_j \quad (\sigma(z) - y)}{\cancel{\sigma(z)(1-\sigma(z))}}$$

w.r.t $\sigma'(z) = \sigma(z)(1-\sigma(z))$

$y' = y(1-y)$ $\qquad \sigma(z) = \hat{y}$

$$\boxed{\frac{\partial L}{\partial w_j} = \frac{1}{n} \sum x_j \underbrace{(\sigma(z) - y)}}$$

similar to MSE

$\frac{1}{2}(\hat{y} - y)^2 \xRightarrow{\frac{\partial}{\partial y}} (\hat{y} - y)$

softmax activation func.

$$\boxed{a_j = \frac{e^{z_j}}{\sum_k e^{z_k}}}$$

o/p layer

probability score

o/p layer	Softmax A. func	Probabilities
$\begin{bmatrix} 1.3 \\ 5.1 \\ 2.2 \end{bmatrix}$	$\dfrac{e^{z_j}}{\sum_k e^{z_k}}$	$\begin{bmatrix} 0.02 \\ 0.90 \\ 0.05 \end{bmatrix}$