Feb 9th, 2017 Convolutional Neural Networks

Motivation: MLPs and AEs have too many parameters. (= weights)
Eg.
* a tiny pic of 30 x 30 resolution: for a neural network, it is 900 inputs.
first layer will have 80,000 weights!!
* 224 x 224 images of faces: 65,000 pixels. ∴ 65,000 inputs for MLP.
AE is not designed for this kind of task. 1st layer = 600,000 weights

we combined 3 ideas:

1. Receptive fields (neighborhoods) in the vision system (of cats)
2. weight sharing. (for a bunch of inputs)
   (eg. share 9 weights for 230,000 numbers)
3. down-sampling (idea from A.E.)
   = Pooling

] Convolution

feature 1 feature 2 ...

So far, feature extraction was manually engineered (SIFT)
(manually handrafted features)

(Network shud itself extract features. Not rely on anybody else outside)

Breakthrough.

What is Convolution?

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

$x(t)$ = input

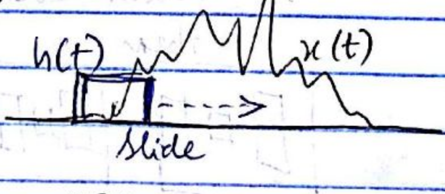
$y(t)$ = output

$h(t)$ = unit impulse response of system

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

τ = dummy variable

$$y(n) = \sum_{-\infty}^{\infty} x(k)\, h(n-k)$$



$$h = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \qquad X = \begin{bmatrix} & & \\ & & \end{bmatrix}_{200 \times 200}$$

$$\sum \begin{bmatrix} 10 & 10 & 10 \\ 50 & 50 & 50 \\ 100 & 100 & 100 \end{bmatrix} * \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} = \sum \begin{bmatrix} -10 & 0 & 10 \\ -50 & 0 & 50 \\ -100 & 0 & 100 \end{bmatrix} = 0$$

vertical edge detector

no vertical edges ∴ 0
(has horizontal edge)

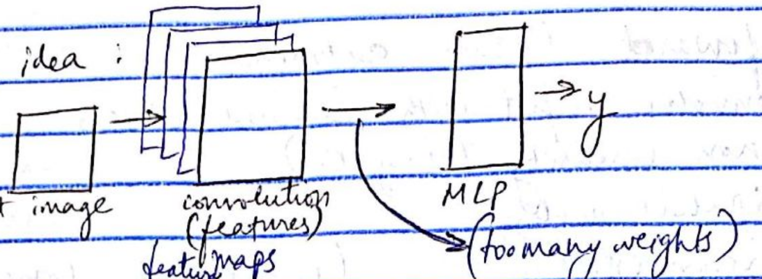
Weights as filters?

$$\begin{matrix} x_1 \xrightarrow{w_1} \\ x_2 \xrightarrow{w_2} \\ x_3 \xrightarrow{w_3} \end{matrix} \bigcirc \rightarrow y = w_1 x_1 + w_2 x_2 + w_3 x_3.$$

Scanned by CamScanner

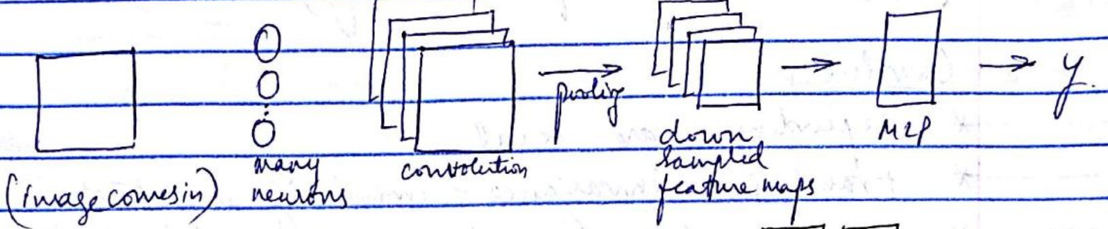
Draft idea :

version ①

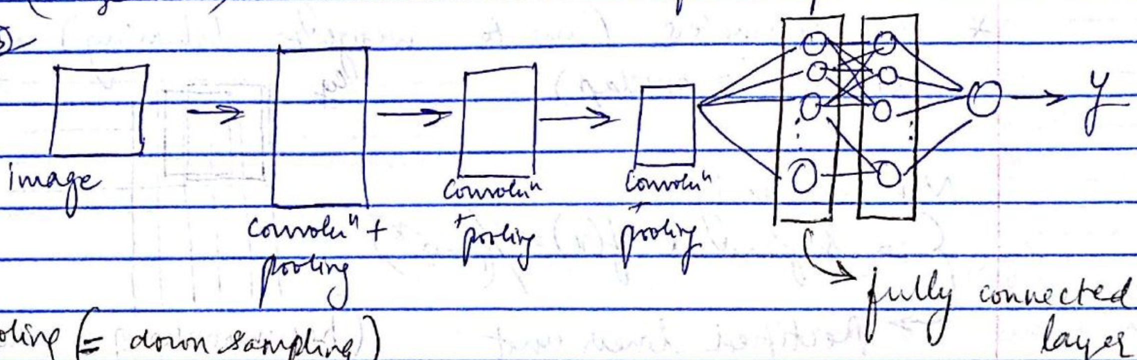


if : many images. way more ip than MLP can handle.

version ②



version ③



Pooling (= down sampling)

1	2	2	1
3	4	3	2
5	6	5	7
10	8	9	8

2x2

4	3
10	9

max pooling
(maximum in each 2x2 block)

Convolution = getting features.

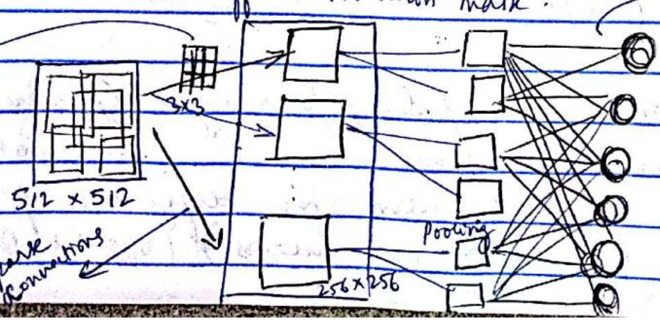
Pooling = Reduce dimensionality

MLP layer = classification

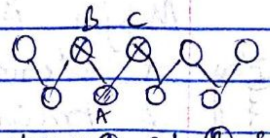
MLP : can never handle raw data.
∴ it needs features. Shallow structure

CNN • Sampling maks it invarient to translation. ∴ many samples with diff. convolution mask.

earlier, this filter was manually decided. But in CNN, the filter is a part of neural network & automatically figured out.



dense connections.



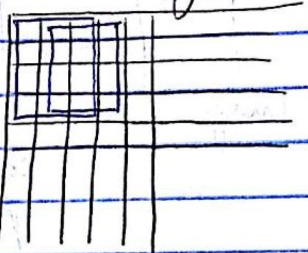
If i change Ⓐ, only Ⓑ & Ⓒ are changed. = weight sharing.
Local Sparsity

Scanned by CamScanner

Feed foward Feature extraction
* convolve input with learned filters.
* non-linearity (logistic)
* spatial pooling
* normalization (usualy you convolve ⇒ Normalize ⇒ ~~pat~~ pool

(weight sharing = filter sharing)

Convolution
* dependencies are local.
* translation invariance - immensely important!
* Few parameters (due to weight sharing)
* Stride (= overlap) eg.



Non-linearity
→ Sigmoid $f(x) = 1/(1+e^{-x})$
→ Rectified linear unit (ReLU)
$f(x) = \max(0, x)$
This simplifies Backprop.

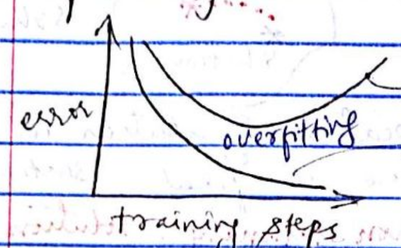
(has an overlap of 1 column. (= stride = 1) if required, we can have a stride of 2 or more. If it is 1, then ~~lengthy~~ probably computationally expensive)

Common Models
- AlexNet (2012) winner on imageNet
- ZFNet (2013) " "
- GooLeNet (2014) " "
- VGG (2014 runner up)
- ResNet (2015) winner on imageNet.

Major issues with CNN / Deep learning in General
difficulty
* design requires some care (but not THAT bad)
* training is still expensive for difficult problems.
* generalization (inspite of all success) (opposite of overfitting)
biggest problem

Regularization

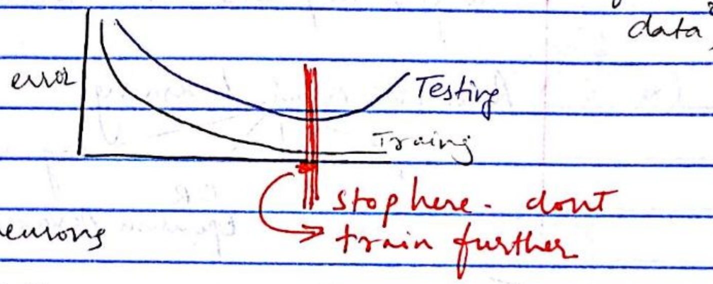
Any change to increase generalization capability (but not necessarily reducing training error), It is about the error of testing. Not training



Types of Regularization:

(1) Data Augmentation
- translation, scaling, rotation
(increase the # of training data)

(2) Early Stopping



(3) Dropout

- randomly remove neurons
- neurons become more insensitive to other neurons. (avoids overfitting)
  (learn local dependencies. They become more independent)
- equivalent of averaging several models. (ensemble).
  (like training multiple N.N.)

(4) Weight Penalties (L2 & L1 regularization)

Total error $E = \text{error} + \|w\|^2$ (size of weights)

$E = \text{error} + \|w\|$ reduce the error, but do it using very small weights.

(already minimized). force it. to choose small weights.

to reduce E, the only option is to reduce $\|w\|$.