DLP - Week-9

L1 Introduc^n to Computer Vision

- we want to extract the meaning from pixels. Computer thinks like a matrix
  - → semantic info. → how many objects are
    Eg. object categorizaⁿ, scene & context categorizaⁿ
  - → vision as measurement device
    - → stereo, structure from motion, reconstrucⁿ from photos.
- Object recogniⁿ → labelling each image with the dominant object.
  - → + localizaⁿ with labelling
- Face detecⁿ + localizaⁿ
- Number plate reader
- Image segmentaⁿ → works like clustering of the pixels as per the objects.
- Object tracking in videos, activity recognition.
- Image captioning → image to text translaⁿ.

- depth from stereo → recover depth by find coordinate x' to correspond x.
- depth from a single image, structure from motion (by rotating your camera)
- photometric stereo (camera fixed, shadow changes)
- Challenges
  1. viewpoint variation Eg. portrait
     ↳ solve by augmentation.
  2. illumination → too much Δ in lightening.
  3. scale → bias issues
  4. deformation for animal poses.
  5. occlusion & background clutter
  6. motion blur & object intra-class variation.

L2 Deep learning for Computer Vision

- how does comp. recognise an image —

 → classifier

shallow network

→ edges → classifier

SIFT / HOG

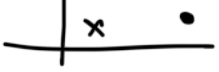
→ edges → histogram → classifier

→ edges → histogram → k means sparse code → classifier ←

deep network

- we started deep learning features, it provides a richer solution space, thus, we can train it end-to-end learning by BackProp.
- Perceptron → any func^n that is linearly seperable.

But what abt. XOR func^n —



we can use more than 1 line. But 1 line can not separate.

- then came, multi layer perceptrons.
  ↳ hidden layers, 'increase abstrac^n. hence, better to have more hidden layers than a single layer with large no. of neurons

L3 CNN (Convolutional Neural Networks)

- if you use a MLP to classify 3^* 224 x 224. typical a imagenet image ⇒ parameter explosion.
  ↳ it doesn't use to exploit local spatial information.

- CNN has more leeeeeverage :—
  1. local connectivity (LC)
  2. parameter sharing (PS)
  3. pooling / subsampling
  4. ReLu (rectifier) non-linearity
- CNN are MLP with 2 constraints
  → LC & PS.
- MLNN (7×3 21 params)

MLNN-LC (3 × 3 = 9 params)

hidden layer (3 nodes)

↑

input layer (7 nodes)

m inputs, n output nodes & CNN local connectivity of k nodes ($k < m$):

MLNN –
1. $m \times n$ params
2. $O(m \times n)$ runtime

MLNN-LC –
1. $k \times n$ params
2. $O(k \times n)$ runtime

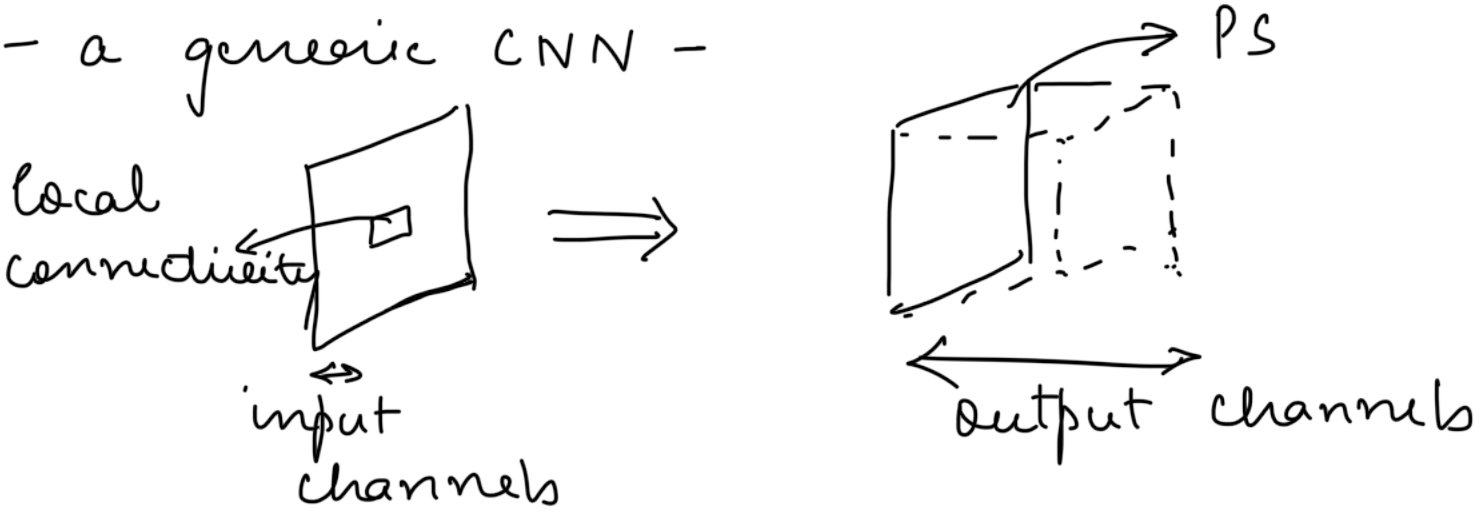
2.3X runtime & storage efficient.

MLNN-LC-PS (3 params)

↳ 2.3X Faster & 7X storage efficient.

1. k params to store.
2. $O(k \times n)$ runtime.

- CNN with multiple input channels, output channels
- a generic CNN -



- Convolutional layer.

1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

image

→

4	3	4
2	4	3
2	3	4

1. to reduce the no. of wts. (LC)
2. to provide spatial invariance (PS)

- hyperparams for CNN.
  1. 0 padding (to control input size spatially).
  2. stride (to produce smaller output volumes spatially)
  3. both padding & stride.
  4. single depth slice of max pool layer.

- max pool layer -

1. to reduce the spatial size of the representation to reduce the amt. of param. and computation in the network.
2. avg. pooling or L2 pooling can be used, but not popular like max pooling.

- Activation function → sigmoid function.
  ↳ It saturates & kills gradients (when the neuron's activation saturates at either tail of 0 or 1).

→ ReLu (Rectified Linear Unit)
1. eliminates saturation & killing of gradients for +ve inputs.
2. it accelerates convergence of SGD.
3. tanh / sigmoid neurons involves expensive operations. (ReLu can be implemented by simply thresholding activation at 0.)

- Flattening → vectorization (converting MXNXD tensor into MND x 1)
- FC Layer → MLP, generally used in final layers, works as a classifier
- Softmax Layer → $Z_n = \frac{e^{x_n}}{\sum_{i=1}^{n} e^{x_i}}$

L4 Intro. to Object Recogni^n

- object recog^n (image classifica^n - a core problem in CV.

 → classify it as an object.

- challenges -
  1. viewpoint varia^n → all the pixel value Δs when the camera moves.
  2. illumina^n → too much contrast
  3. deforma^n → diff. angles.
  4. occlusion & background clutter
  5. intraclass variation
- Now, we will study the CNN architectures
- LeNet 5

  conv filters 6×5, stride = 1

  pooling layers were 2×2, stride = 2

  CONV - POOL - CONV - POOL - FC - FC.

L5 Alex Net & VGGNet

- had 8 layers

  CONV 1
  MaxPool 1
  Norm 1
  CONV 2
  MaxPool 2
  Norm 2

  details
  1. first use of ReLu
  2. used Norm layers (now not in use)
  3. heavy data augmenta^n

CONV3
CONV4
CONV5
Max Pool 3
FC6
FC7
FC8

4. dropout = 0.5
5. batch size = 128
6. SGD momentum = 0.9
7. $\eta = 0.01$
8. L2 weight decay = $5e^{-4}$
9. 7 CNN ensemble.

- uses filters of 11×11. It increases the no. of training parameters. Now, these days CNN uses cascade of filters of size $3*3$ or $5*5$.
- VGGNet → small filters, deep networks

8 in Alex, 16 or 19 in VGG

11.7% error ⟹ 7.3%.

→ stack of 3 3×3 conv. (stride 1) layers has same effective receptive field as 1 7×7 conv layer. But now CNN becomes more deeper, more non-linearities. And lesser params to compute.

$3 * 3^2(C^2)$ vs $7^2 C^2$

if C is no. of channels per layer.

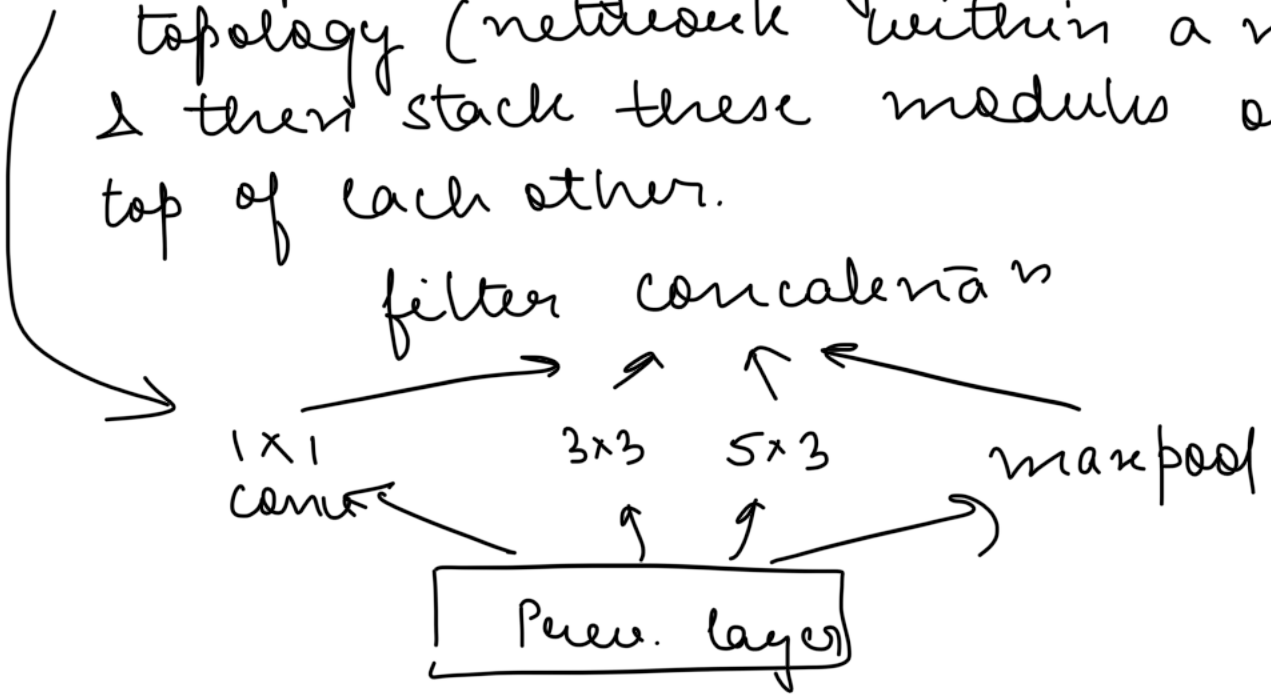
→ details -

1. no norm layers
2. uses ensembles for better results.
3. FC7 features generalizes well to other

tasks as well.

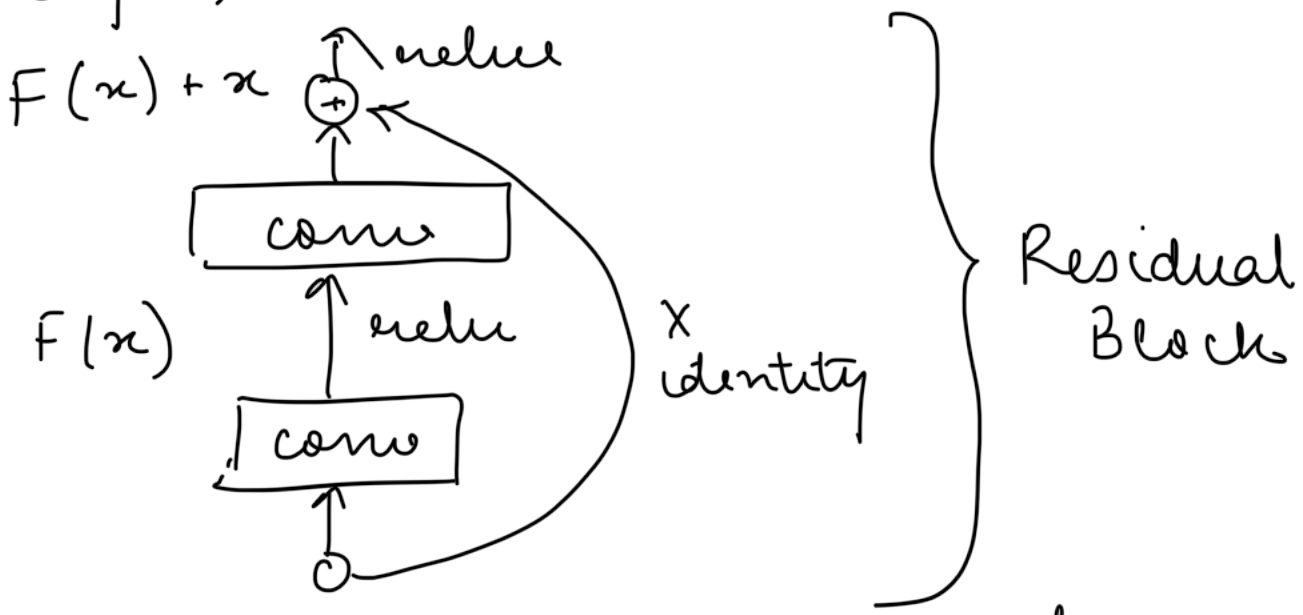
L6 GoogleNet & ResNet

- GoogleNet
  - → Details → deeper networks with computational efficiency.
    1. 22 layers.
    2. efficient 'Inception' module.
    3. No FC layers.
    4. only 5 M params. (12x less than AlexNet)
    5. error 6.7%.
  - → Inception → design a good local network topology (network within a network) & then stack these modules on top of each other.



  - → apply filter operations (1) on the inputs from prev. layers. Finally, concatenate all filter outputs together depth-wise.
  - → problem → computation complexity. There will be total 854 multiplication operations

- ResNet - very deep networks using (152 layers) residual connections.



- the deeper model should be able to perform at least as well as the shallower model. A possible solution is by constructing the copied learning layers from the shallower model & setting additional layers to identity mapping.