C4

WEEK-2

- classic networks
  - → LeNet-5
  - → AlexNet
  - → VGG

ResNet (152 layer deep neural network)
Inception

- **LeNet-5** (60 k parameters)
  - → handwritten digit recognition
  - → greyscale images

non-linearity after pooling (σ)

$32 \times 32 \times 1$ (input image) $\xrightarrow[f=5,\ s=1,\ 6\ filters]{}$ $28 \times 28 \times 6$ $\xrightarrow[f=2,\ s=2]{avg\ pool}$ $14 \times 14 \times 6$ $\xrightarrow[f=5,\ s=1,\ 16\ filter]{}$ $10 \times 10 \times 16$ $\xrightarrow[f=2,\ s=2]{avg\ pool}$ $5 \times 5 \times 16$ (400) → 120 → 84 → $\hat{y}$

... modern (softmax)

conv - pool - conv - pool - fc - fc - output

$n_H, n_W \downarrow$; $n_C \uparrow$

- **AlexNet**

$227 \times 227 \times 3$ (input image) $\xrightarrow[f=11\times 11,\ s=4,\ no.\ of\ filters = 96]{}$ $55 \times 55 \times 96$ $\xrightarrow[3\times 3,\ s=2]{max\ pool}$ $27 \times 27 \times 96$

$27 \times 27 \times 96$ $\xrightarrow[5\times 5,\ same,\ no.\ of\ filters = 256]{}$ $27 \times 27 \times 256$ $\xrightarrow[3\times 3,\ s=2]{max\ pool}$ $13 \times 13 \times 256$ $\xrightarrow[3\times 3,\ same,\ 384\ filter]{}$ $13 \times 13 \times 384$ $\xrightarrow{3 \times 3}$ $13 \times 13 \times 384$ $\xrightarrow{3 \times 3}$ $13 \times 13 \times 256$ $\xrightarrow[3\times 3,\ s=2]{max\ pool}$ $6 \times 6 \times 256 = 9216$ $\xrightarrow{FC}$ 4096 $\xrightarrow{FC}$ 4096 $\xrightarrow{softmax}$ 1000

- similar to LeNet but much bigger (60M parameters)
- ReLU activation
- Multiple GPUs
- Local Response Normalization (LRN)

- **VGG-16** (VGG-19 also used)

16 weighted layers.

conv = 3×3 filter, $s = 1$, same
Max pool = 2×2, $s = 2$

$224 \times 224 \times 3$ $\xrightarrow[\times 2]{[CONV\ 64]}$ $224 \times 224 \times 64$ $\xrightarrow{pool}$ $112 \times 112 \times 64$ $\xrightarrow[\times 2]{[CONV\ 128]}$ $112 \times 112 \times 128$ $\xrightarrow{pool}$ $56 \times 56 \times 128$ $\xrightarrow[\times 3]{[CONV\ 256]}$ $56 \times 56 \times 256$

56 × 56 × 256 $\xrightarrow{\text{POOL}}$ 28 × 28 × 256 $\xrightarrow[\times 3]{[\text{CONV } 512]}$ 28 × 28 × 512

Pool ↓

POOL ↓ 14 × 14 × 512 ← [CONV 512] × 3 ← 14 × 14 × 512

7 × 7 × 512

↓

FC 4096 → FC 4096 → softmax
1000

n_H, n_W ↓ & n_C ↑

Parameters ≈ 138 M.

• ResNets

* Skip connections = take activation from one layer & feed it to another layer deep in the neural network

* Residual block : $a^{[l]}$ → [o o o o] $a^{[l+1]}$ → [o o o o] $a^{[l+2]}$ →

"shortcut" / "skip connection"

$a^{[l]}$ → Linear → ReLU $a^{[l+1]}$ → Linear → ReLU → $a^{[l+2]}$
(a) (b) (c) (d)

"main path"

(a) ⇒ $z^{[l+1]} = W^{[l+1]} a^{[l]} + b^{[l+1]}$

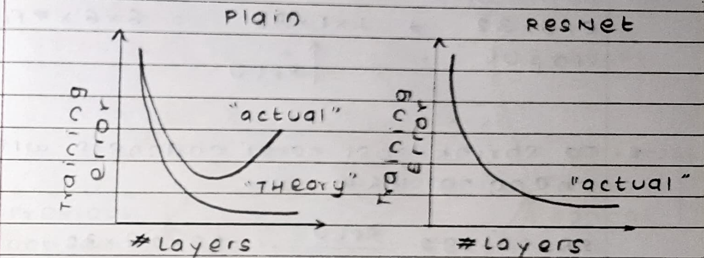
(b) ⇒ $a^{[l+1]} = g(z^{[l+1]})$

(c) ⇒ $z^{[l+2]} = W^{[l+2]} a^{[l+1]} + b^{[l+2]}$

(d) ⇒ $a^{[l+2]} = g(z^{[l+2]})$

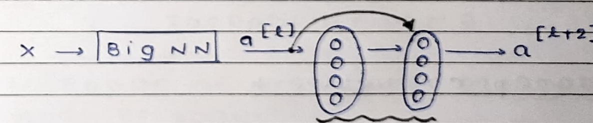
→ skip all layers.

For skip connection ⇒ $a^{[l+2]} = g(z^{[l+2]} + a^{[l]})$
(Reduce Training error)



• Make use of same convolutions & preserve dimensions

• Identity fx^n is easy for residual block to learn



Activation ⇒ ReLU → $a \geq 0$

$a^{[l+2]} = g(z^{[l+2]} + a^{[l]})$

$= g(\underset{\uparrow 0}{W^{[l+2]}} a^{[l+1]} + \underset{\uparrow 0}{b^{[l+2]}} + a^{[l]})$: if

$= g(a^{[l]})$

$= a^{[l]}$

Page No.
Date

- 1x1 convolution (1x1 filter)

  ∴ network in network

* particularly useful for images & filters with more than 1 channels.

6x6x32 * 1x1x32 = 6x6x # filters
input

↑ + ReLU

* To shrink just no. of channels without changing H & W

$28\times28\times192 \xrightarrow[\text{conv } 1\times1]{\text{ReLU}} 28\times28\times32$
filters: 32

* If we want to keep no of channels same, we can just add non-linearity (ReLU) using 1x1 convolution

- Inception Network

* stack / concatenate output of all,

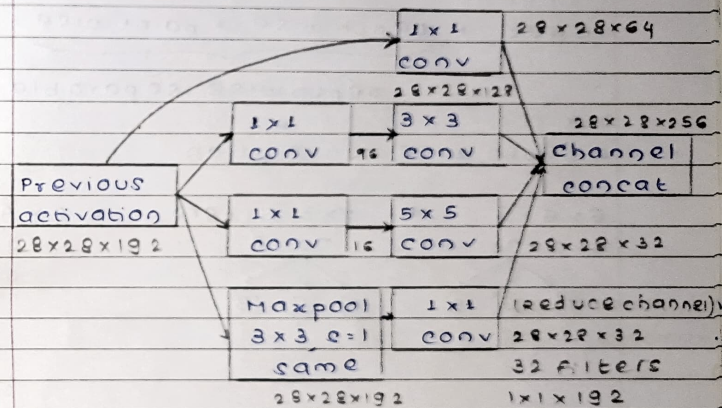
28x28x64 1x1 convolution
28x28x128 3x3 convolution with same padding
28x28x32 5x5 with same padding
28x28x32 max pooling (same padding, s=1)

28x28x192 ⟶ 28x28x256
i/p image

- SOFTMAX LAYERS ⇒ make predictions

Page No.
Date

* problem of computational cost can be reduced significantly by performing 1x1 convolution before 3x3 or 5x5. This creates a bottleneck layer.



Inception module

* Inception network puts inception modules together

- MobileNet

Build & deploy neural nets that work in low compute environment like phones. (less powerful GPU / CPU)

→ Normal convolution

6x6x3 * 3x3x3 = 4x4
n x n x n_c f x f x n_c (n-f+1) x (n-f+1)
(1 filter)

* computational cost = #filter params × #filter positions × # of filters

eg: $\underline{2160} = (3 \times 3 \times 3) \times (4 \times 4) \times (5)$

→ Depthwise separable convolution

$\underbrace{\text{input} * \text{Depth wise} * \text{pointwise}}_{\text{Depthwise separable}} = \text{o/p.}$

1) Depth wise convolution

$6 \times 6 \times 3$ * 3×3 filters $= 4 \times 4 \times 3$

$(n \times n \times n_c)$ $n_c = 3$ 1 each (RGB)

computational cost = #filter params × #filter positions × # of filters

$432 = (3 \times 3) \quad (4 \times 4) \quad (3)$

2) Pointwise convolution

$4 \times 4 \times 3$ * $1 \times 1 \times 3$ $= 4 \times 4 \times 5$

$(n_o \times n_o \times n_c)$ $(1 \times 1 \times n_c)$ n_c'

1 filter 5 $(1 \times 1 \times 3)$ filters

computational cost = #filter params × #filter positions × # of filters

$240 = (1 \times 1 \times 3) \times (4 \times 4) \times 5$

NOTE: Depthwise separable convolution cost over normal convolution cost ratio

$= \frac{1}{n_c'} + \frac{1}{f^2} \; \ldots \left(\frac{1}{5} + \frac{1}{3^2} = 0.31 = \frac{432+240}{2160} \right)$

[~10 times cheaper]

* MobileNet v1

Input (RGB) → Depthwise separable convolution (13 times) → Pool/FC/softmax

* MobileNet v2

RESIDUAL CONNECTION

Input (RGB) → expansion → Depthwise → Projection → Pool/FC softmax

17 times

Bottleneck block

* Bottleneck block

• Need:

i) By using expansion it increases size of representation within the block allowing the NN to learn richer function

ii) Projection operation used for down scaling, reducing memory requirement

RESIDUAL CONNECTION

$n \times n \times 3 \rightarrow n \times n \times 18 \rightarrow n \times n \times 18 \rightarrow n \times n \times 3$

$1 \times 1 \times 3$ expansion, 18 filters

Depth-wise

$1 \times 1 \times 18$ pointwise, 3 filters ... projection

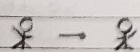
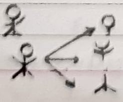
- Efficient Net

Scale up or down the NN based on resources:
i) change r → resolution of image
ii) change d → depth of network
iii) change w → width of layers

- Transfer learning (train your convnets from pretrained convnets of others)
- Train our own softmax layer by freezing parameters of all layers b/w input & softmax
⇒ trainableParameters = 0
⇒ freeze = 1
FOR SMALL TRAINING SET ↑

- For larger training set, freeze fewer layers train more layers

- Data Augmentation

More data input
Data Augmentation improves performa-nce of CV systems.
DISTORTIONS
1. Mirroring 
2. Random cropping 
3. Rotation
4. Shearing
5. Local wrapping
6. Colour shifting
Increase / decrease values of R G & B

Hard-disk ⟶ CPU thread (introduce distortions) ⟶ CPU / GPU Training

- Data v/s hand-engineering

Sources of knowledge:
1) Labeled data
2) Hand engineering features/network architecture/ other components.

Tips:
1. Ensembling → Train several networks independently & average their outputs

2. Multi-crop at test time → Run classifier on multiple versions of test images & average results