Week-4

- ILSVRC → ImageNet Large Scale Visual Recognition Challenge
  - 1000 categories ⇒ task of classifica^n among these categories
  - Metric ⇒ top-5 error rate %

1. AlexNet (2012) ⇒ 28% → 16% ⇒ CNNs were introduced
   - huge jump

(227 × 227 × 3) → RGB
Grayscale (1)

↓ kernel, stride, padding

conv 1 (55 × 55 × 96)

↓ maxpool 2×2

conv 2 (27 × 27 × 96)

↓

conv 3 (27 × 27 × 256)

↓

conv (4) (13 × 13 × 256)

⋮

conv (8)

↓

FC layers (3/4)

↓

Classification layers.

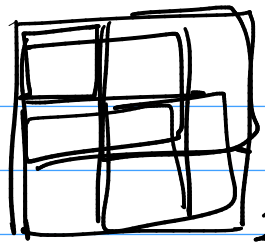
max pool layers btw conv layers

There were total 60 Million parameters

→ used 2 GPUs

→ use of Relu Activa^n & dropout were introduced for the 1st time → max (0, a)

→ only during the train time to make sure there is no bias.



kernel = 2×2
stride = 1
padding = 0

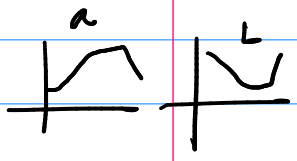
3×3 → 3×3

3×3 → 2×2

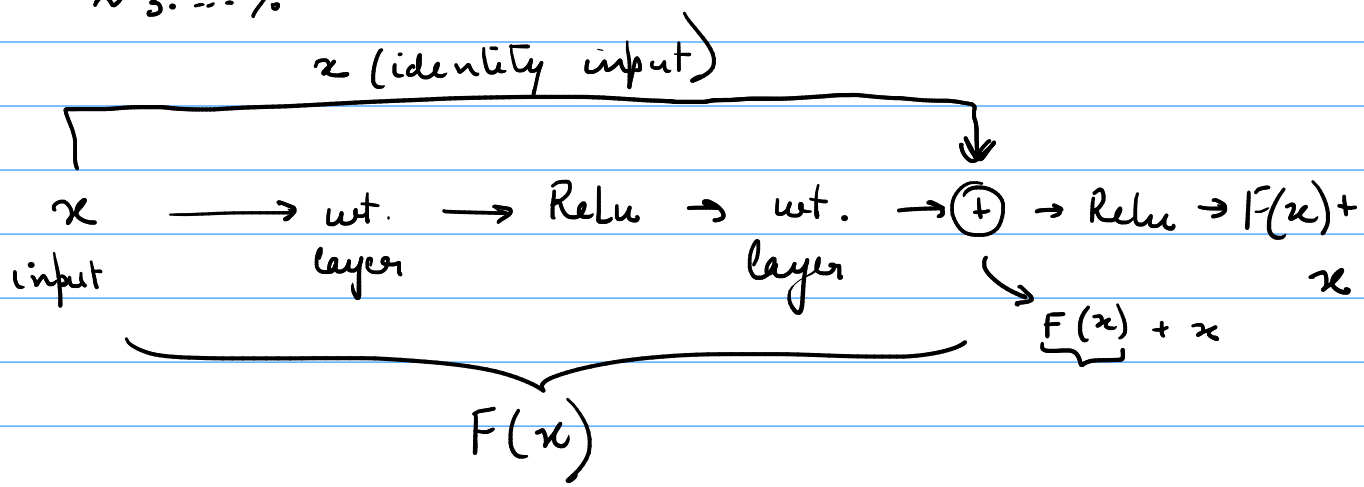
→ ~ 8%

2. VGG Net (2014)
   - → 16 - 19 layers → extremely dense network
   - → performance better
   - → in the network only 3×3 kernel layers were used.
   - → they stacked up these 3×3 layers
     - ↳ larger effective receptive field.

⇒ Challenges → when the network depth (no. of layers) starts increasing beyond a certain pt., then accuracies tend to saturate & then decay exponentially.



3. Res Net ⇒ Residual block ⇒ introduces a skip connection.
   ↳ ~ 3. ... %

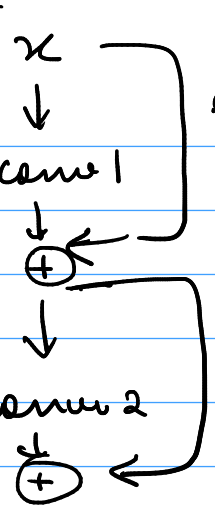


→ It adds block X input directly to the output F(x) ⇒ F(x) ≠ H(x) + x

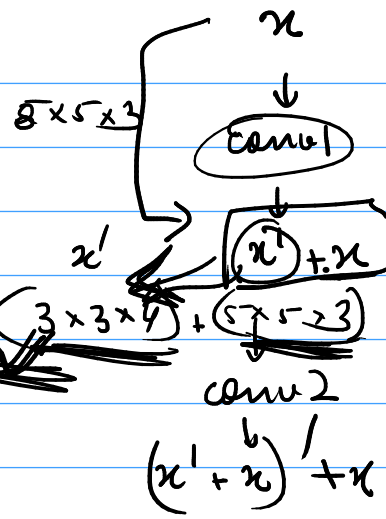
⇒ Residual fun

A + B =

Solved the problem of creating extremely deep networks (152 layers) → w/o sacrifising the performance

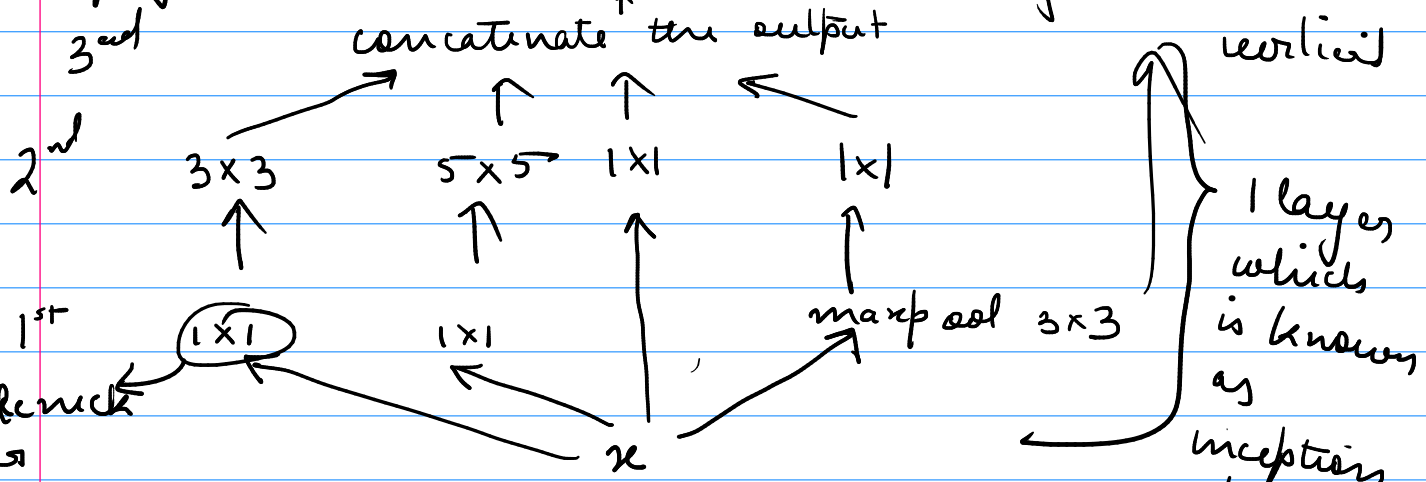


residual block } which brought the skip connection



↗ 6.~ %

4. GoogleNet ⇒ Inception module



bottleneck layer ↓ which performs the dimensionality reduc^n before the 3×3/5×5 expensive computa^ns

each of these vertical gave input to different attributes ⇒ which captures different portions of the same image

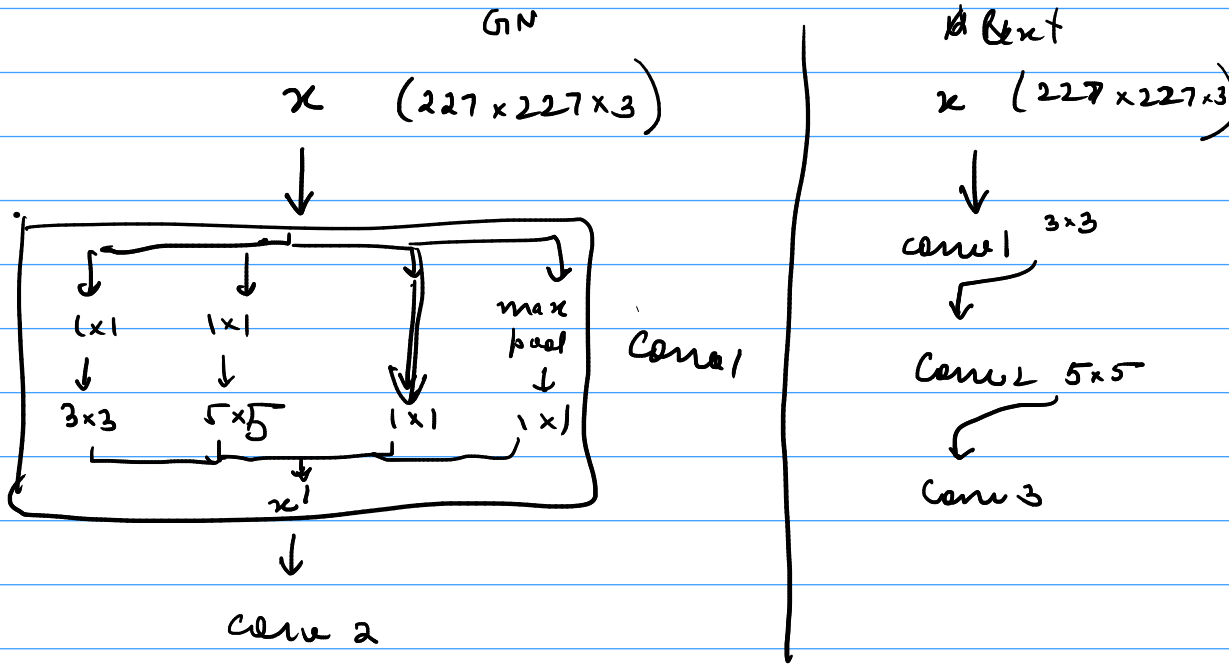
→ they converted the entire vertical network into a cluster of both vertical & horizontal network ⇒ parallel conv.

→ parallel convolution was performed by →
- → kernels → 1×1, 3×3, 5×5
- → max pool

→ even though, the no. of params ↓, it is computationally efficient

AdaptivePool ⟶ PyTorch
⟵

→ FC layers was replaced GAP (global avg. pooling) info ing the parameters



- Transfer Learning
  ↳ how we utilize a pre-existing architecture to prepare for a specific downstream tasks.

General process —
1. use a pre-trained conv. base (wts & parameters)
2. we will remove one single classifier head (1000) layer.
3. add our own custom classifier head.
4. just train it.

- VGGNet → Freeze all its layers & replace the classifier head.
  ↳ only if your dataset is small

- AlexNet → unfreeze all the wts. & use the new data to retrain the model either partially or fully. ⇒ Fine-tuning