Chapter 05 - Transfer Learning

5.1 Introduction to Transfer Learning

Competitio - wining models are difficult to train from scratch :-

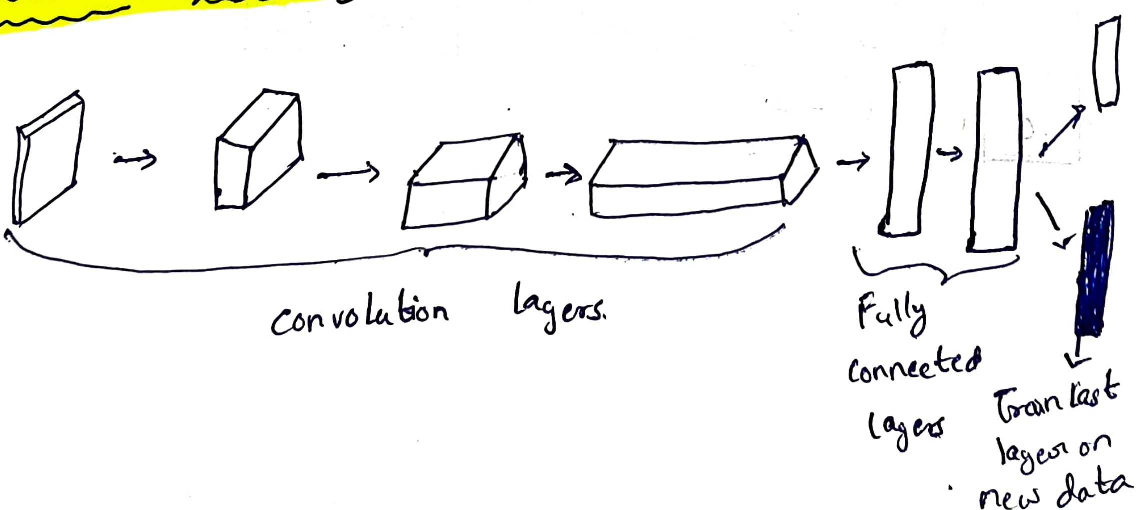
- Huge dataset
- Long no. of iterations - very heavy computation machinery
- Time experimenting hyperparameter

However the basic features (edges & shapes) learned in the early layers of network should generalise

Results of the training are just weights that are easy to store rather than images

Idea to keep the early (images) layers of a pre-trained network, & re-train the later layers for a specific application. This is called Transfer Learning.

Transfer Learning:



5.2 Transfer Learning & Fine Tuning

Transfer Learning Options:

The additional training of a pre-trained model / network on a specific new dataset is reffered to as "Fine Tuning"

→ There are diff options on "how much" & "how far back" to fine tune?

- should I train just the very last layer?
- Go back a few layer?
- Or re-train the entire network?

Guiding principles for Fine Tuning:

* The more similar your data, & problem, are to the source data of pre-trained network, the less finetuning is necessary

  Eg:- using a network trained on 'Image Net' to distinguish 'dogs' from 'cats'. Need little fine-tuning

* The more data you have about your specific problem, the more the network will benefit from longer & deeper fine-tuning.

  Eg:- If you have only 100 dogs & cats, you want to do very little fine-tuning

- If you have 1000 dogs & cats, you may get more value from longer & deeper fine tuning

* Also if your data is substantially different in nature than the data the source model was trained on, Transfer learning may be of little value

Eg:- A network that was trained on recognizing typed Latin alphabet would not be useful in dog & cat distinguishing problem

5.3 Convolutional Neural Network Architectures:-

5.3.1 LeNet:-

- It is firstly developed for black & white images.
- Created by Yann LeCun
- used on MNIST data set
- use convolution to efficiently learn feature on data

LeNet - Structure Diagram:-

Image (32x32) ⟶ CNN (5x5)
(B&W)

stride = 1
output = (28x28)
depth = 6 = kernels
output = 6x28x28

bias
each kernel = 5x5 = 25 +1
= 26
26x6 = 156 weights
To be learned

→ Pooling → CNN (5x5)
stride = 1
padding = None
depth = 16 = kernels.
output = 16 x 10 x 10

weights = 6x5x5+1 = 151
total weights = (151) x 16 = 2416
→ Pooling (2x2)
Output = 16x5x5

→ Flatten into 400 vector → Next layers are fully connected layers. so we can go from 400 → 120 → 84 → 10 → (softmax) predict the class

How many total weights in network?

Conv1 : 1 x 6 x 5 x 5 + 6 = 156

Conv2 : 6 x 15 x 5 x 5 + 16 = 2416

FC1 : 400 x 120 + 120 = 48120

FC2 : 120 x 84 + 84 = 10164

FC3 : 84 x 10 + 10 = 850

Total : 61706

Less than a single FC layers with [1200 x 1200] weights.

LeNet-5 Summarized:-

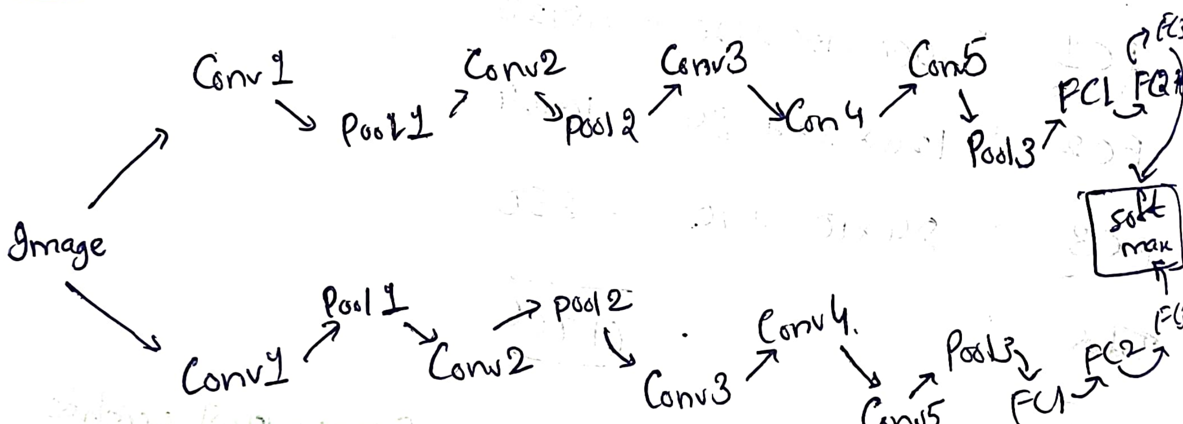
Input → Conv1 → Pooling → Conv2 → Pooling → Flatten → FC1 → FC2 → FC3 → softmax → $\boxed{\text{predict}}$

---- A 10 step process

5.3.2 AlexNet

- Created in 2012 for ImageNet Large scale Visual recognition Challenge (ILSVRC)
- Task: predict image among 1000 classes
- Data: 1.2 million images
- It is considered for 'flash point' in DL
- AlexNet performed "Data Augmentation" before training - cropping, flipping, etc.
- Basic templates → convolutions with ReLU's
- some times adds maxpool after Conv
- Fully connected layer at end before softmax classifier

AlexNet - Structure Diagram



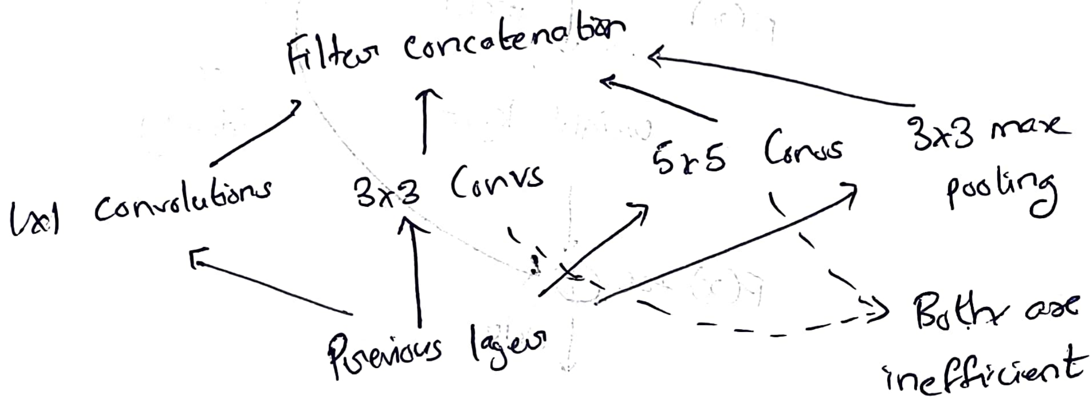
5.3.3 Inception:-

- Idea: network would want to use diff fields
- want computational efficiency
- also want to have sparse activations of group of neurons

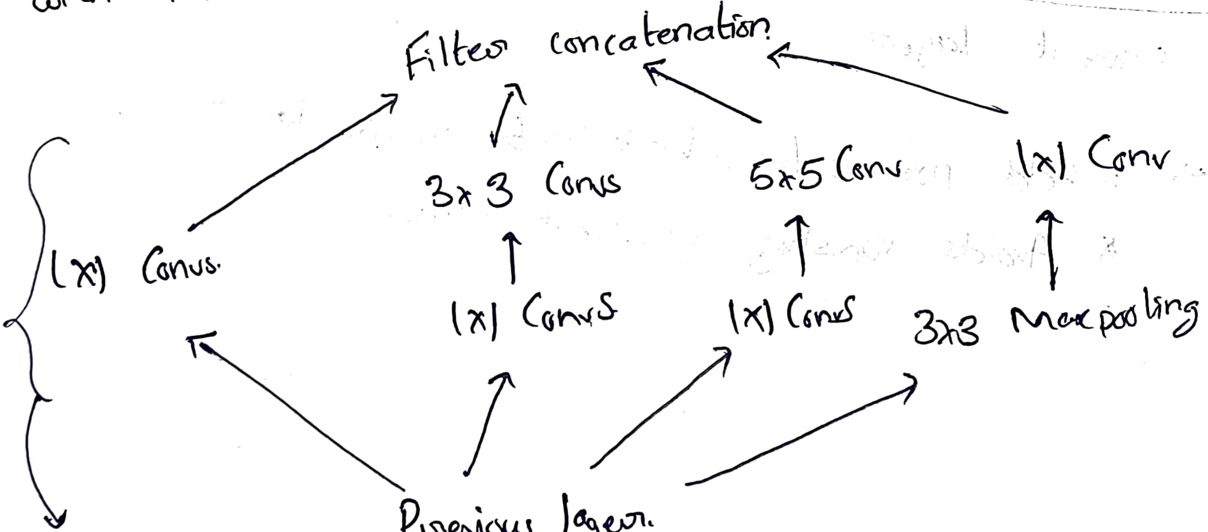
- Solution:-

* Turn each layer into branches of convolutions
* Each branch handles smaller portion of workload
* Concatenate different branches at end

problem 1:- reducing filters from previous layer via 3x3 & 5x5 convolution is inefficient



Insted reduce parameters by reducing filter depth with 1x1 convolutions

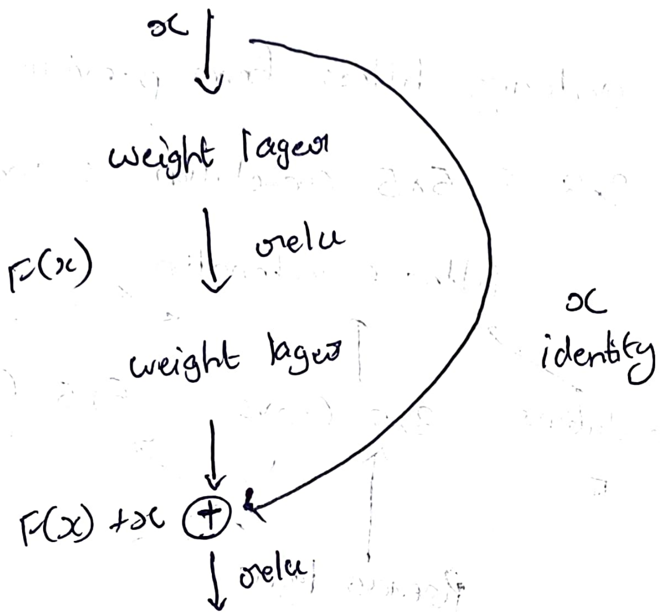


This whole block serves a function of a previous convolution layer.

5.3.4 Res Net

x : input to series of layers

$F(x)$: function represented by several layers (eg convs)



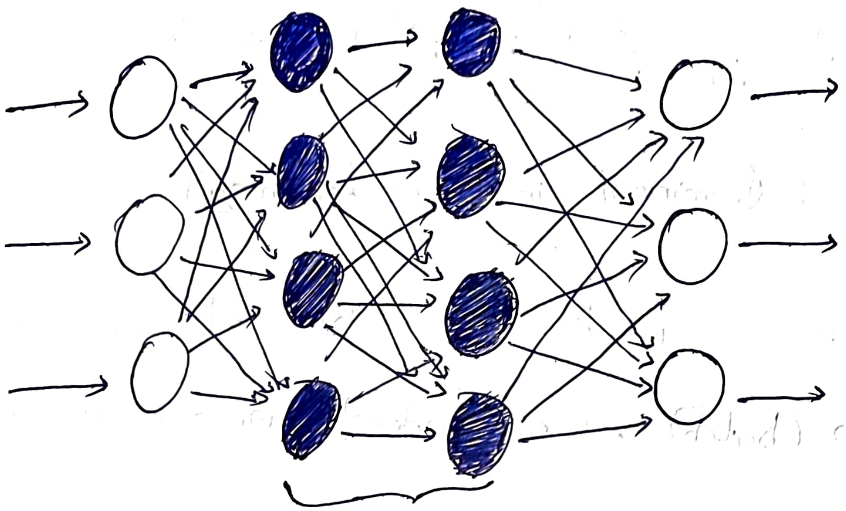
- Enforce "best transformations" by adding shortcut connections
- Add the inputs from an earlier layer to output of current layer

→ * Add previous layer back in to current layer
  * Avoids vanishing gradient issues.

5.4 Regularization Techniques for Deep Learning

A deep NN relates to NN with 2 or more hidden layers in between inputs & outputs

More hidden layers ⟶ More complexity ⟶ Mostly overfitting



hidden layers.

Regularization:- Regularization is any modification we make to a learning algorithm that is intended ~~indeed~~ to reduce it's generalization error but not it's training error

To prevent overfitting, we have several means to regularize neural networks:-

- Regularization penalty, in cost function (like Lasso)
- Dropout (Loosing some neurons)
- Early stopping
- Stochastic - Mini Batch Gradient descent

Penalized Cost Function:

One option is explicitly add a penalty to loss fn for having high weights

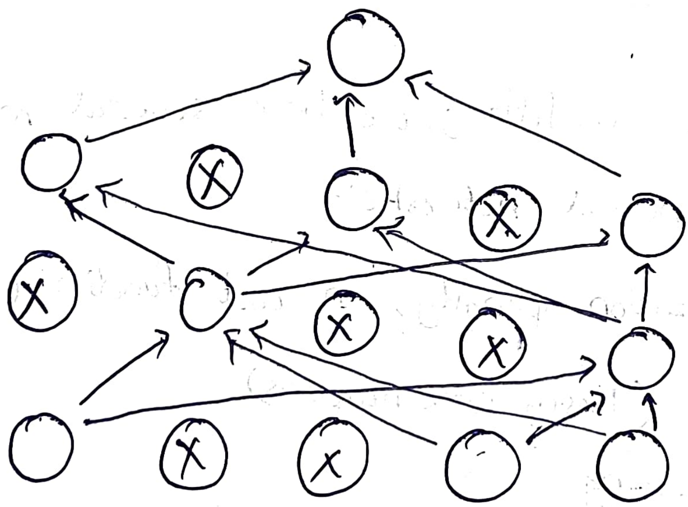
This is similar approach to Ridge Regression:

$$J = \frac{1}{2n} \sum_{i=1}^{n} (\hat{y}_i - y_i)^2 + \lambda \sum_{j=1}^{m} w_j^2$$

Dropout :- (Prominent in Deep Learning)

Dropout is a mechanism where at each training iteration (batch), we randomly remove a subset of neurons:

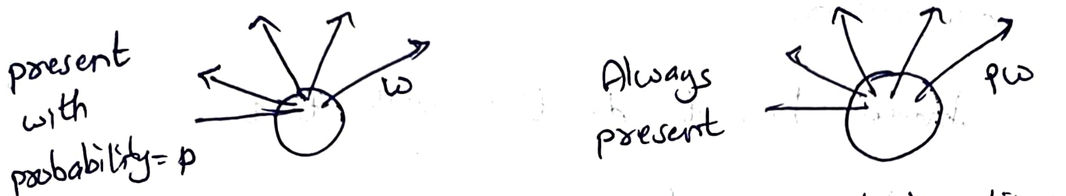
- This prevents the NN from relying too much on individual pathways, making it more 'robust'.
- At test time we 'rescale' the weight of the neuron to reflect the percentage of time it was active



Applying dropout to a Feed Forward Neural Net.

If the neuron was present with probality 'p'.

at test time we scale the outbound weight by a factor of p.



(a) At training time (b) At testing time

Early stopping:

Another heuristic approach to regularization is "early stopping"

This refers to choosing some rules after which to stop training

Example: - Check the validation log-loss every 10sec

- If it is higher than it was last time, stop & use the previous model (i.e from 10 epochs previous)

Important: - Read the Regularization notes after this.

File_name = Regularization Notes.pdf.