Convolutional Neural network

- CNN ax widely used in image recognit -ion image classification, objects detection, recognition face etc.

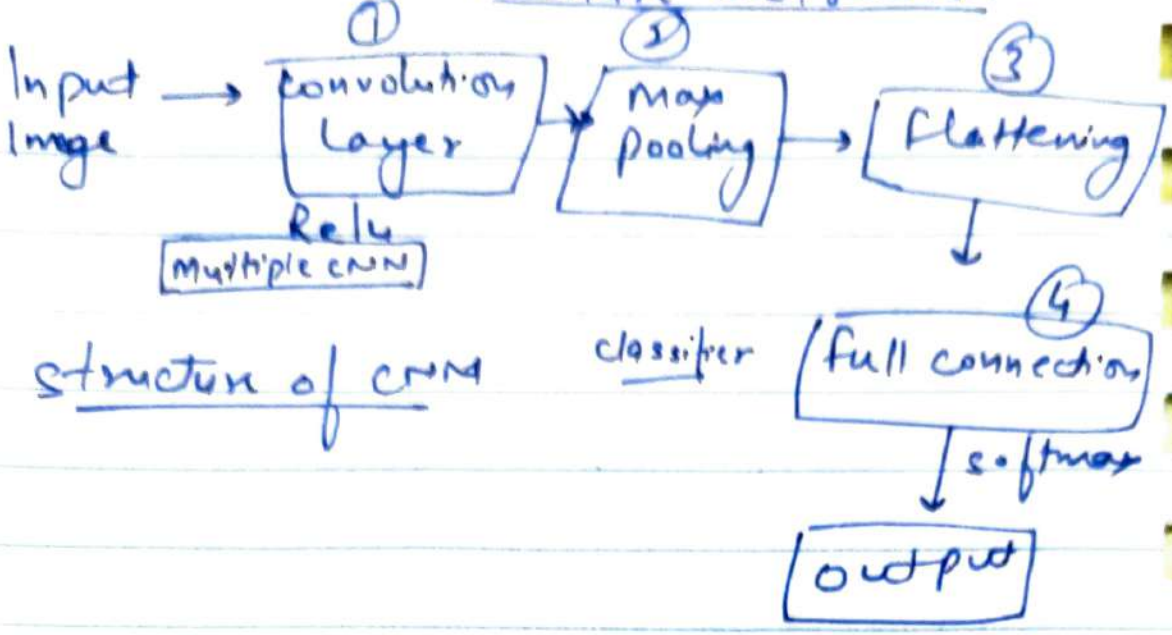
- CNN image classification takes an input image, process it and classify it under certain categories

- CNN is another type of neural network that can be used to enable machine to visulize things and perform tasks such as image classification, image recognition, object detection etc

- Image classification is the task of taking an input image and outputting a class (a cat, dog etc) or a probab -ility of classes that best describe the image

CNN is a specialized type of neural network model designed for working with image data

Feature extraction



structure of CNN

Input Image → CNN → Output label (Image class)

480 × 480 × 3

(Array of pixel values)

RGB value

* Array of number

Output number that describe the probability of the image being a certain class

In a 3D image every pixel will have 3 values separate channel for red green & blue

Each 0 - 255 value describe the pixel intensity at that point

Input image → convolution layer → positive pooling layer → fully connected layer → softmax → output

We want the computer to do is to differentiate between all the images its given:-

- for that ~~output~~ computer perform image classification by looking for low level features such as edge curves and then build up to more abstract concept through a series of convolutio -nal layers

- In CNN the input image pass throug -h a series of convolutinal layers, pooling (downsampling) layers and full connected layers and finally produced the output which can be a single class or a probability of classes that best describes the image

Feed forward neural network can learn a single feature representation of the image but in case of complex image, NN will fail to give better predictions, this is because it can't learn pixel dependencies present in the images

CNN can learn multiple layers of

of feature representations of an image by applying different filter / transformation.

In CNN, number of parameters for the network learn is significantly lower that multilayer network.

Convolution operation: It is a linear application of a smaller filter to a larger input that result in an output feature map.

Steps Convolution layer :- This layer perform an operation called a convolution hence the network is called convolution neural network

- It extract features from the input image Convolution is a linear operation that involves the multiplication of a set of weights with the input

- The technique was designed for 2D input. The multiplication is performed between an array of input data and a 2D array of weight called filter / kernel

- Size of filter is smaller than the input data

- multiplication performed between a filter size patch of the input data and the filter is a dot product, which is then summed and generate a single value

- The filter is applied systematically to each overlapping part or filter sized patch of the input data, left to right top to bottom

- If the filter is designed to detect a specific type of feature in the input (image) then it discover that feature anywhere in the image

- The out put from multiplying the filter with

- If 3 filter have been used then feature map have depth of 3
- The more number of filter the more accurate result

stride Stride is the number of pixels by which we slide our filter matrix

over the input matrix

stride1 :- move the filter one pixel at a time

stride2 :- ——— two ———

Larger stride will produced smaller feature maps

7x7	3x3	5x5
input image	filter/ kernel	output

Input image :- array of pixel values

filter set of weights

feature map :- filter is systematically applied to the input data to create a feature map

- Multiple small feature can be designed to detect multiple feature in the image.

- Value of the filter are weights to be learned during the training of the network so that most useful features are extracted from the input image to classify it

- CNN learn multiple filter in parallel it learn from 32 to 512 filter in

parallel for a given input

- Convolution layer not only applied to input data (raw pixel values) but they can also be applied to the output of other layers

- stacking of convolutional layers allow a hierarchical decomposition of the input

$$n - f + 1$$

weight sharing :- In CNN weights are being used to compute the output

In ANN each neuron present in the hidden layers will have separate weights for itself

Padding :- In convolution operation reduces the size of image (spatial dimensions decrease faster

- Zero padding allows us to control the size of the feature maps
- padding is used to make output size is same as the input size

p - padding amount = no. of rows/columns that we will insert in top, bottom, left and right of the image

$$n + 2p - f + 1$$

Problem with convolution are

1) shrinks output
2) Throwing away a lot of information that are in the edge.

Soln: Pad the input image before convolution by adding some rows and column.

stride s :- It tell us the number of pixels we will jump when we are convolving filter

Input $n \times n$ * filter $f \times f$ p - padding s - stride

feature map $\left[\frac{n+2p-f+1}{s}\right] \times \left[\frac{n+2p-f+1}{s}\right]$

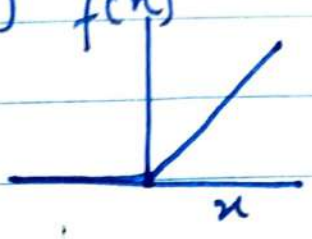
The convolution operation shrinks matrix if $f > 1$

ReLu After every convolution operation, Relu operation has been used.

Rectified Linear unit is a non linear activation function

output = Max(0, Input)

$f(x) = \max(0, x)$



ReLu is an element wise operation that measure if applied perpixel and replace all negative pixel values in the feature map by zero

- Convolution is a linear operation non linearity is introduce by using a non linear activation function like ReLu

1	-2	8
-3	-1	1
1	1	-5

ReLu →

1	0	8
0	0	1
1	1	0

feature-

CNN uses pooling layers to reduce the sized input to speed up computation and to make some of the feature it detect more robust.

steps:- Pooling (spatial pooling) layer
Also called as subsampling or downsampling.

- It is applied after convolution and ReLu operations
- It reduces the dimensionality of each feature map but retains the most important information

- Since the number of hidden layers required to learn the complex relations present in the image would be large

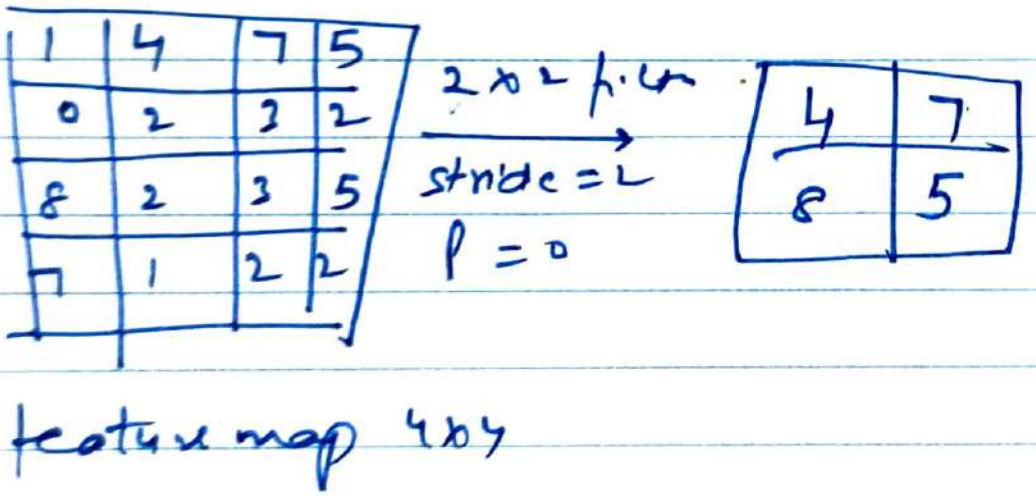
- we apply pooling operation to reduce the input feature representation there by reducing the computation power required for the network
- Spatial pooling can be of different types:-

(1) Max pooling (2) Average pooling (3) Sum pooling

① Max Pooling / subsampling

Once we obtain the feature map of the input, we will apply a filter of determined shape across the feature map to get the maximum value from that portion of the feature map.

- It is also known as subsampling because from the entire portion of the feature map covered by filter/kernel we are sampling one single maximum value



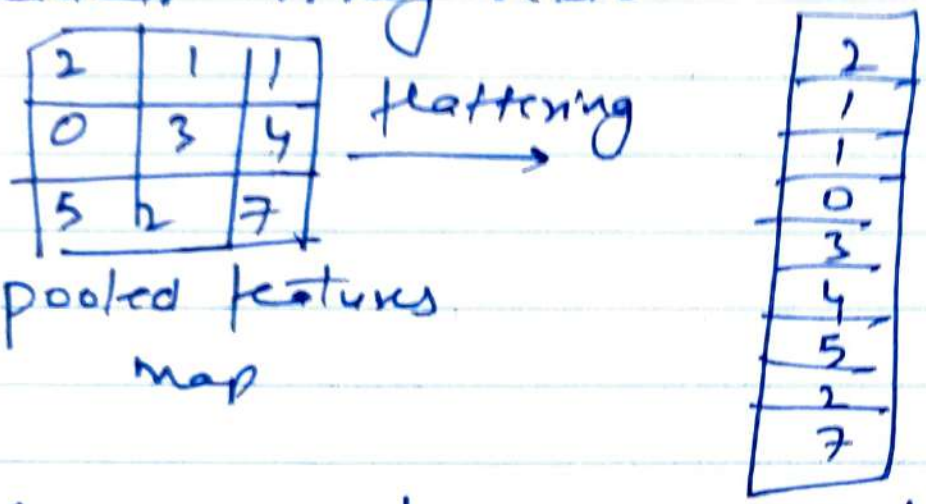
feature map 4×4

② Average pooling

③ Sum pooling compute the sum of all elements

Step3 Flattening

After a series of convolution and pooling operation on the feature representation of the image. We will flatten the output of the final pooling layer into a 2D single long continuous linear array vector



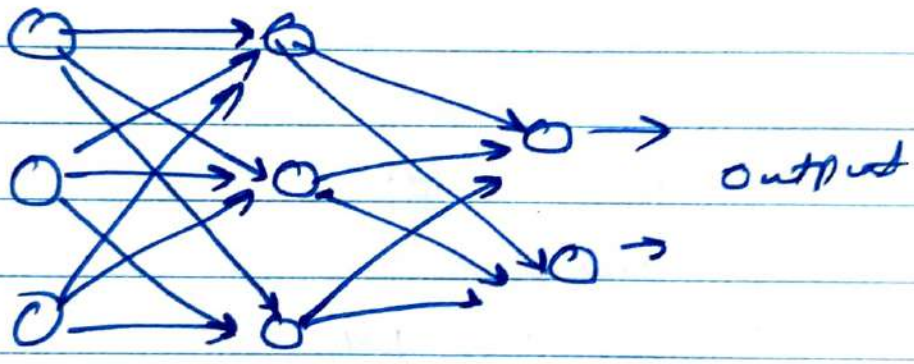
pooled features map

The process of converting all the resultant 2D array into a vector called flattening.

CNN. | FC layer is to use high level of information merge produced by convolution and pooling layer for classifying the input image into various classes based on the training dataset

fully connected means every neuron in the previous layer is connected

to every neuron in the next layer.
* Sum of output probabilities from the fc layer is one.
Fully connected uses a softmax activation function in the output layers
The softmax function takes a vector of arbitary real valued scores and transform it to a vector



Step 4: Fully connected layer (dense layer)
flatten output is fed as input to the fully connected ANN

- ANN have varying number of hidden layers to learn the non linear complexities present with the feature representation

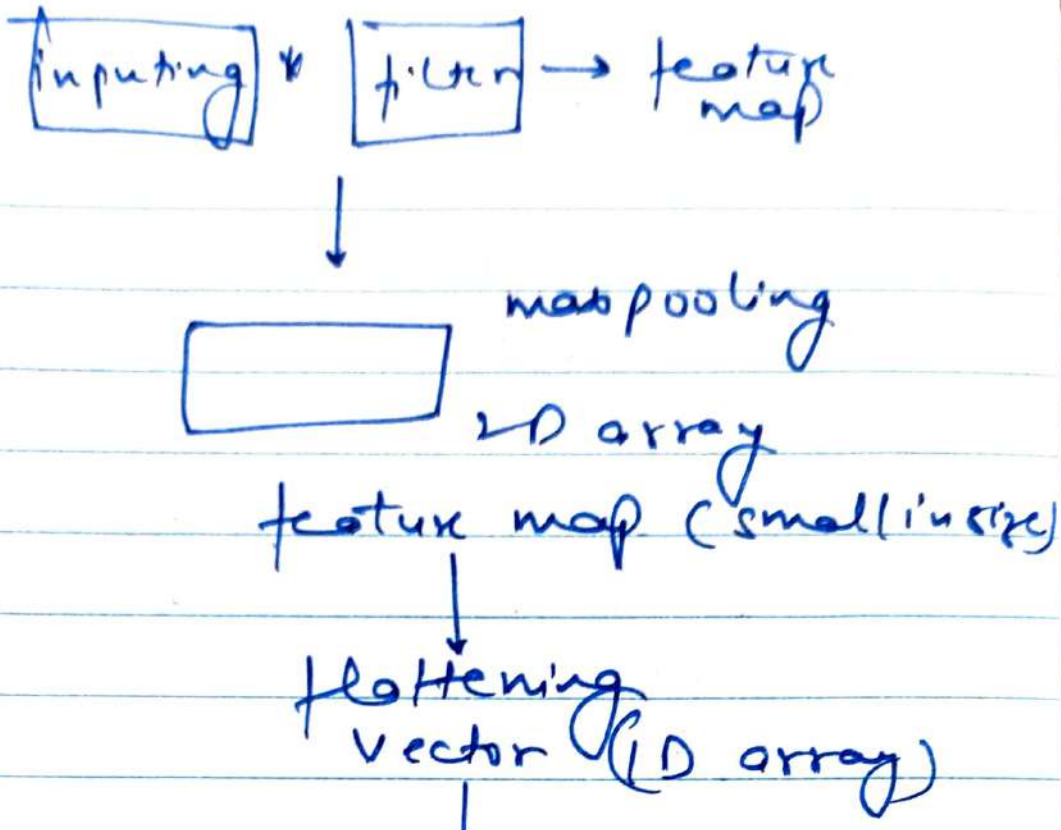
* Sum of output probabilities from FC Layer is one

Fully connected uses a softmax activation function in the output layer
-r
the softmax function takes a vector of arbitrary ~~transition~~. real valued scores and transform it to a vector of values b/w zeros and one that sums to one

Working of CNN

Input image
↓
convolution (multiply filters)
+ Relu
↓ feature map
↓
max. pooling (reduction size of feature map)
↓
flattening
↓
Dense layer / fully connected layer
↓
output vector

* One filter creates one corresponding feature map.



Training of CNN

(1) Initialize all filter weight with random values

(2) Forward propagation (Input → convolution → Relu → pooling → fc → o/p probability for each class)

(3) Calculate the total error (SSE)

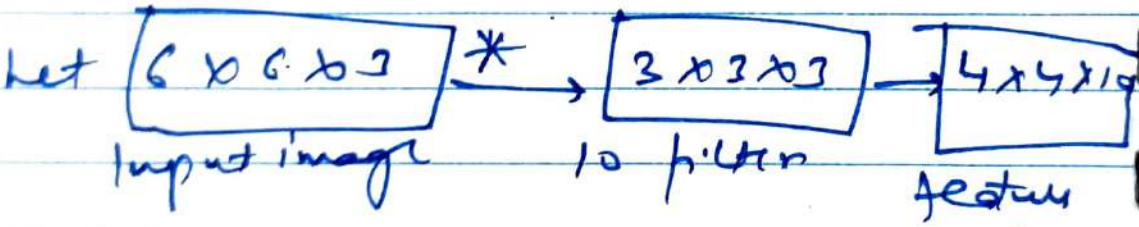
$$\frac{1}{2}\sum(\text{target}_p - \text{output}_p)^2$$

4. Backpropagation (using gradient descent) update weights and parameters Repeat till minimum loss / error. Repeat for all input images (training set)

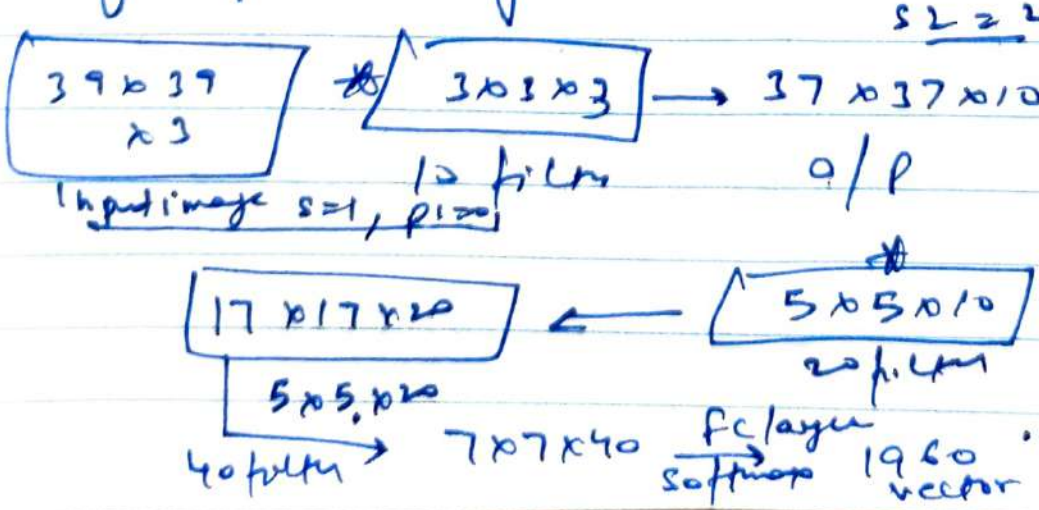
Convolution over 3D images

We will convolve an image of height, width, number of channel with a filter of a height, width, same number of channels.

no. of channels in image = no. of channels in filter



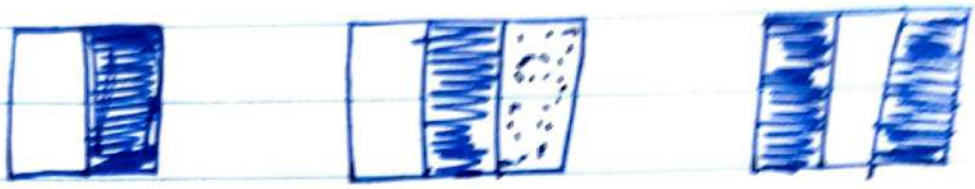
we use multiple ~~features~~ filters to detect multiple features or edges of input image



Vertical and Horizontal edge detection

Vertical Edge detection

1	0	-1
1	0	-1
1	0	-1



Horizontal Edge detection

1	1	1
0	0	0
-1	-1	-1

1X1 Convolutions (Network in Network)

Problem of CNN — 3x3 filter will be applied in 3x3x1 block

3x3x3 blocks to create a single value.

6x6x3 → 3x3x3 → 4x4x# filter

- If the size of filter is relatively large like 5x5 or 7x7 pixels, it can result considerably more parameters (weights) which require large computation to

perform the convolutional operations.

- Convolutional and pooling Layer reduces the height and width of feature map leaving the depth number of channels or number of filters unchanged (number of feature maps)

- So deep CNNs requires a layer that can downs sample or reduce the depth or number of feature maps.

3x3 filter will be applied on 3x3x64 blocks to create a single value to make up the single output feature map.

When the depth of network ↑es, the depth of input or number of filter used in convolutional layers often ↑es, resulting in an ↑se in number of resulting feature maps

- A Large ~~map~~ number of feature maps in a CNN can cause a problem as a convolutional operation must be performed down through the depth of the input

Solution Use a 1×1 filter to downsamp. or number of feature maps

1×1 convolution can be used to reduce only the depth as it slices through the input volume with just 1×1 unit

- It save computation task
- It adds non linearity
- It allows to learn more complex function.

- It generates an element wise product the depth, slices through the entire volume.

- Useful inner larger filter sizes and used such as 5×5, 7×7

- It is used for dimensionality reduction

- It is so useful in many CNN models

channel ↓ — ↓ channel

6×6×32 * 1×1×32 → 6×6×5

Input #filter 5 output

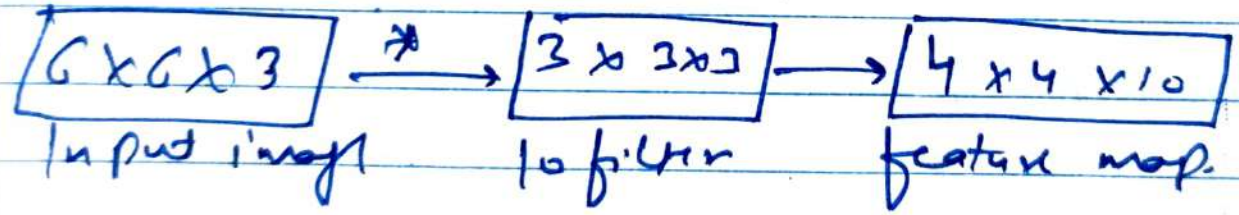
- It has been used in a lot of modern CNN implementations like ResNet & inception models

- It is useful when we want to shrink the number of channel called feature transformation. This save a lot of computations

Convolution over 3D images

We will convolve an image of height, width, number of channels with a filter of a height width, same number of channels

no. of channels in image = no. of channels in filter

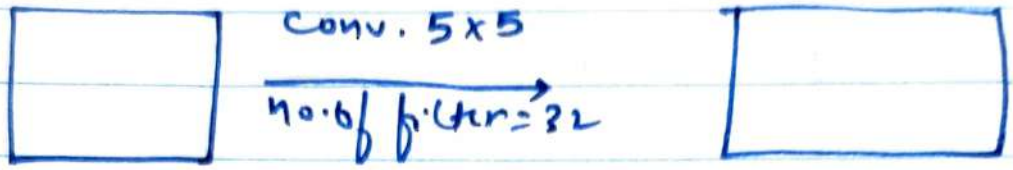


We use multiple features to detect multiple features or edges of input images

Inception network During design of a CNN you have to decide all the layers yourself

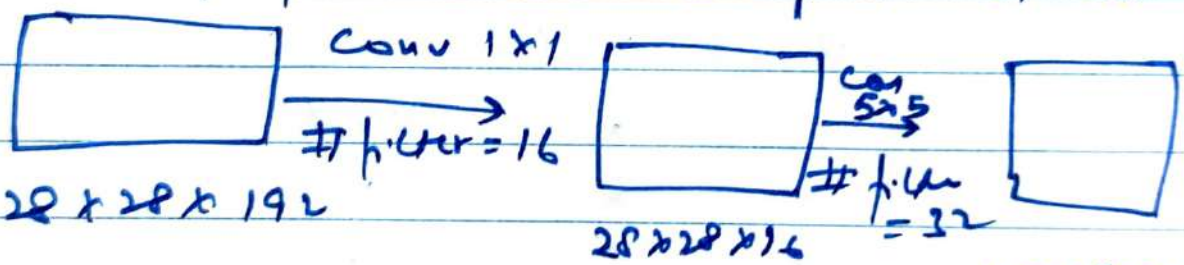
Choices — 3×3
5×5
max pooling layer

5×5 convolutions



computation cost $28 \times 28 \times 5 \times 5 \times 192 \times 32$
$= 120$ M (high computation cost)

Using a 1×1 convolution with 5×5 convolution to reduce computation cost



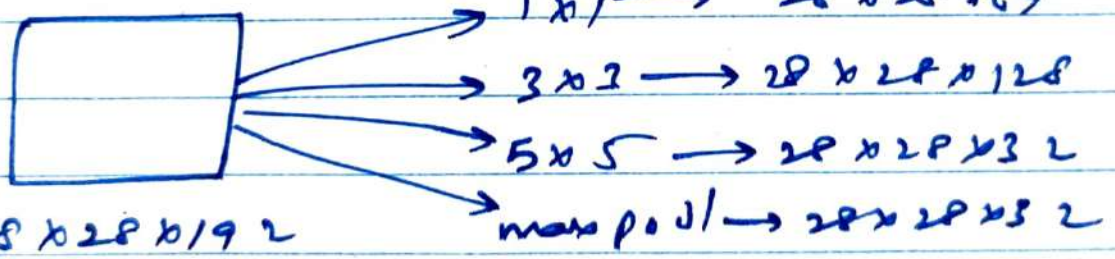
(dimension reduction)

- computation cost $(28 \times 28 \times 16) \times (1 \times 1 \times 192)$
$+ (28 \times 28 \times 32) \times (5 \times 5 \times 16)$
$= 12.4$ M (approx)
- Hence 12.4 M is immensely smaller than 120 M ~~If~~ It means 1×1 convolution can help to reduce model size which is then also help to reduce computation

Inception network, instead of choosing a 1x1 or 3x3 or 5x5 filter or a pooling layer we apply all together

- This makes the network architecture more complicated but remarkably improve performance as well
- The inception network or inception layer says - instead of choosing the desired filter size in a conv. layer or whether we want a convolutional layer or a pooling layer lets apply all of them

Inception module, naive version.



$= 64 + 128 + 32 + 32 = 256$

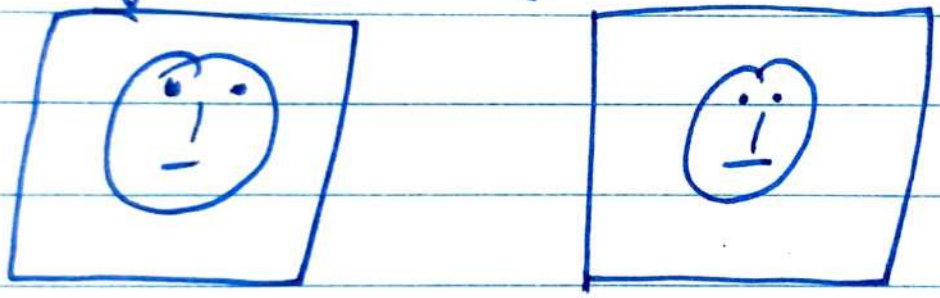
28x28x192 → 28x28x256

Input output

High computation cost

Here we used all the conv & pool we might want and will let the neural network learn and decide which it want to use most

Inception layer allows the internal layers to pick & choose which filter size will be relevant to learn the required information so even if the size of the face is in the image is different, the layer works according to recognize the face

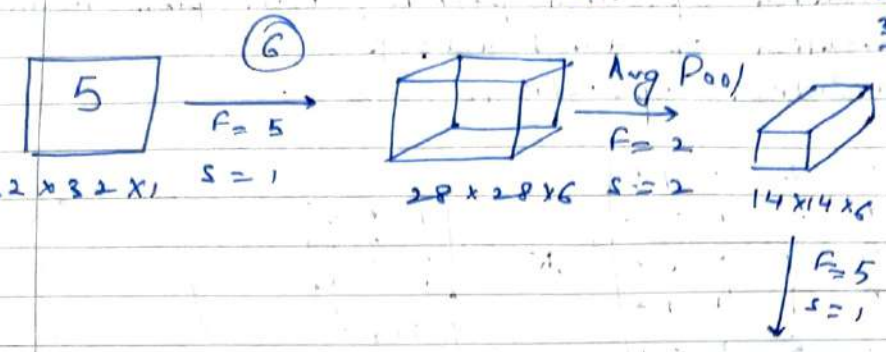


Spares connections converting CONVs to Deep Neural
Parameters sharing. networks : Non linear computation

DATE

LeNet-5 architecture:

LeCun et al., 1998., Gradient-based learning Applied to Document Recognition



$\frac{n-f+2p}{s} + 1$

$\frac{32-5}{1} + 1 = 28$

$\frac{28-2}{2} + 1$

$\frac{26}{2} + 1$

$= 14$

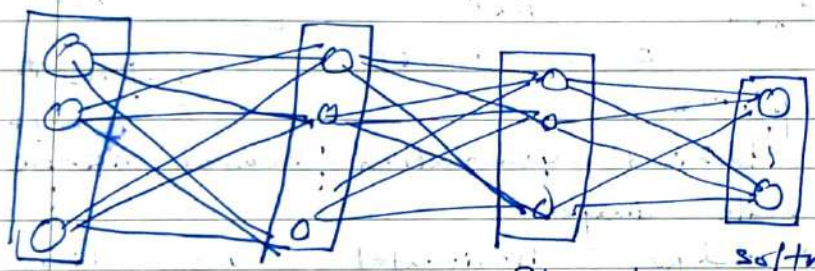
$14 - 5 + 1$

$9 + 1$

$= 10$

$\frac{10-2}{2} + 1$

$\frac{8}{2} + 1 = 5$



(400 × 120) (120 × 84) (84 × 10)

~60k

tanh / Sigmoid

ReLU ✗

$224 \times 224 \times 3 \xrightarrow{f=11,\ s=4,\ p=2} 55 \times 55 \times 96$

$55 \times 55 \times 96 \xrightarrow{\text{max pool } f=3,\ s=2} 27 \times 27 \times 96$

$27 \times 27 \times 96 \xrightarrow{f=5} 27 \times 27 \times 256$

$27 \times 27 \times 256 \xrightarrow{\text{max pool}} 13 \times 13 \times 256$

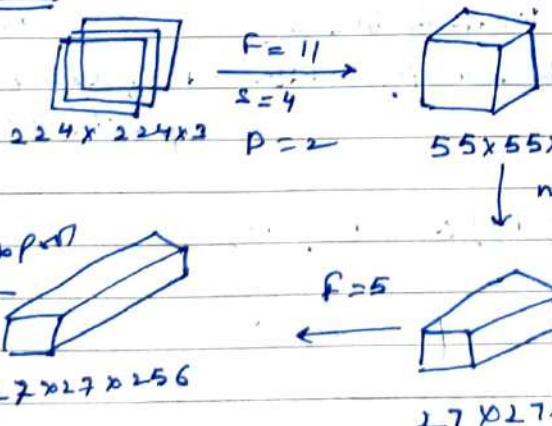
$\frac{224 - 11}{4} + 1$

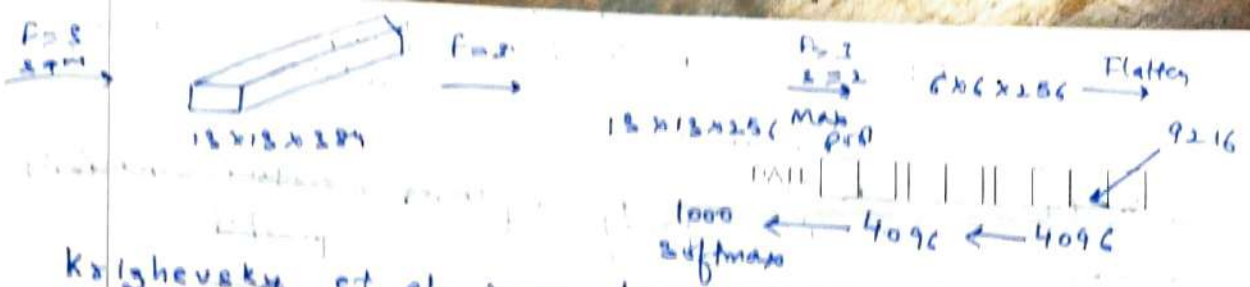
$= \frac{213}{4} + 1$

$\frac{217}{4} + 1$

$54 + 1 = 55$

Alex Net



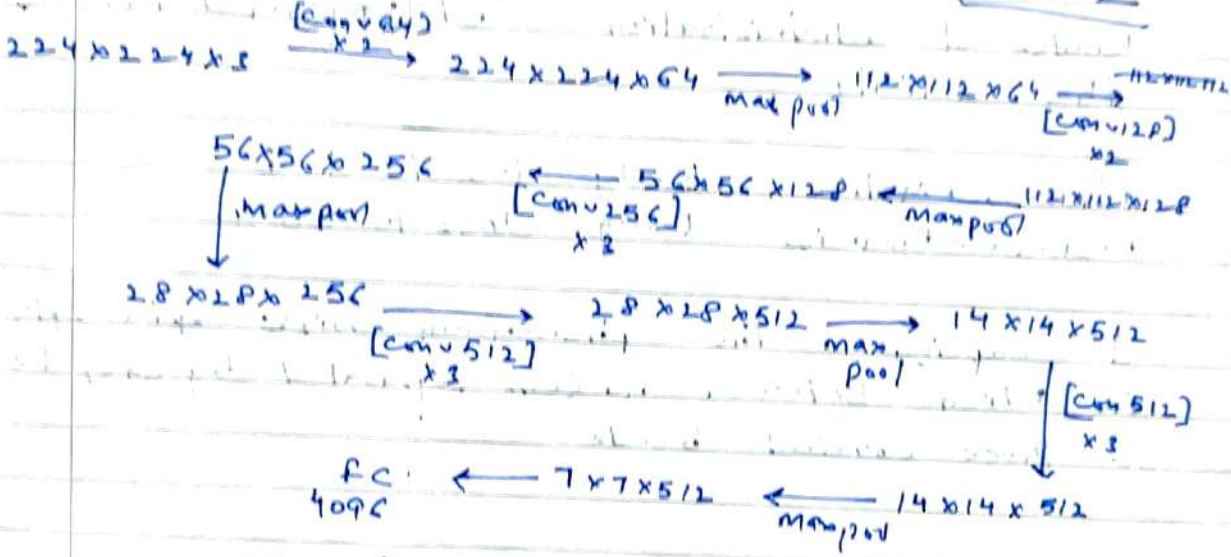


Krizhevsky et al. 2012, Imagenet classification with Deep convolution neural network

ReLU used

VGG-16: Simonyan & Zisserman 2015, Very Deep Convolution Networks for Large-scale image Recognition

Conv = 3×3 filter, s=1, Padding same, Max pool: F: 2×2, s=2



16 Represent no. of learning parameter
~138 Million parameters

VGG-19

ResNet Architecture Kaiming He, Xiangyu Zhang Ren, & Jian Sun "Deep Residual Learning for image Recognition" CVPR 2016

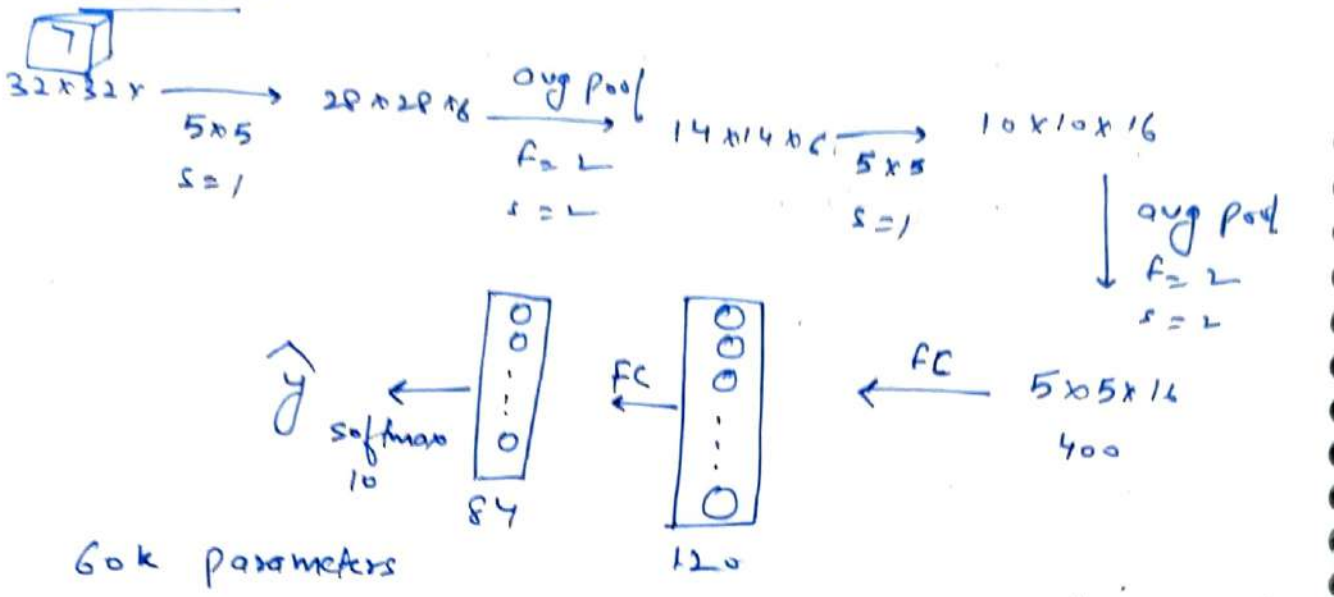
Dotted use the dimension

skip connection

152 Layers, Less overfit

Residual block

→ LeNet 5



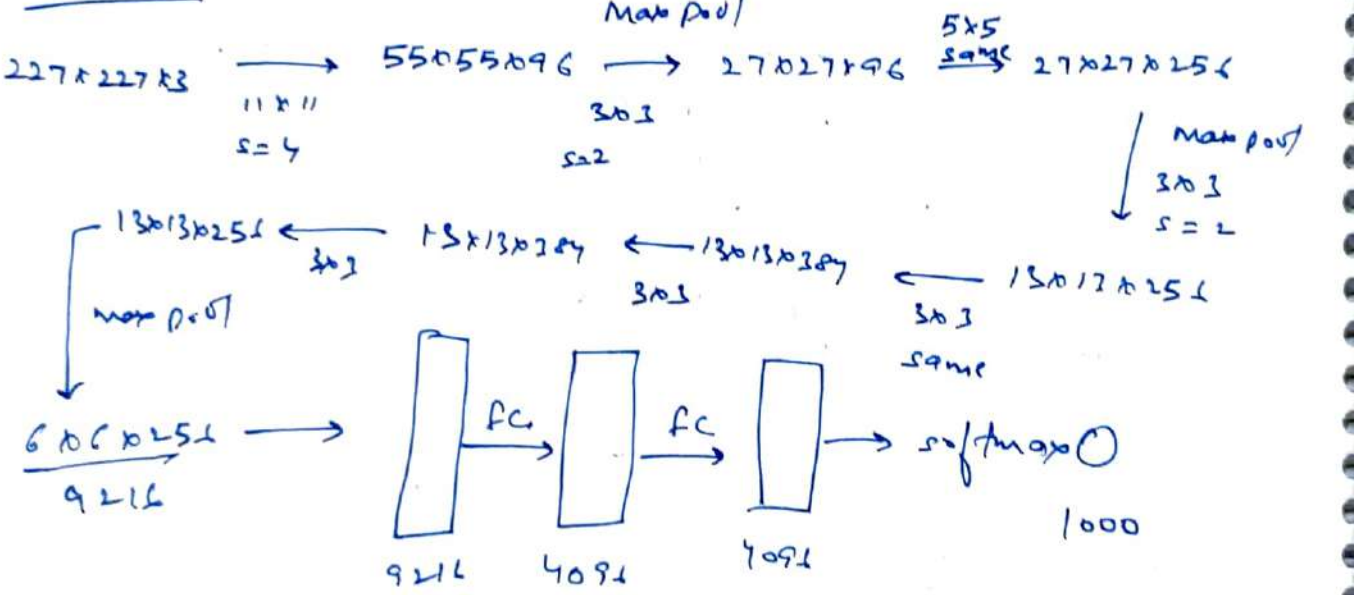
60k parameters

$n_H, n_W \downarrow$, $n_c \uparrow$

conv pool conv pool FC FC output

Advance comments :- . Sigmoid / tanh .

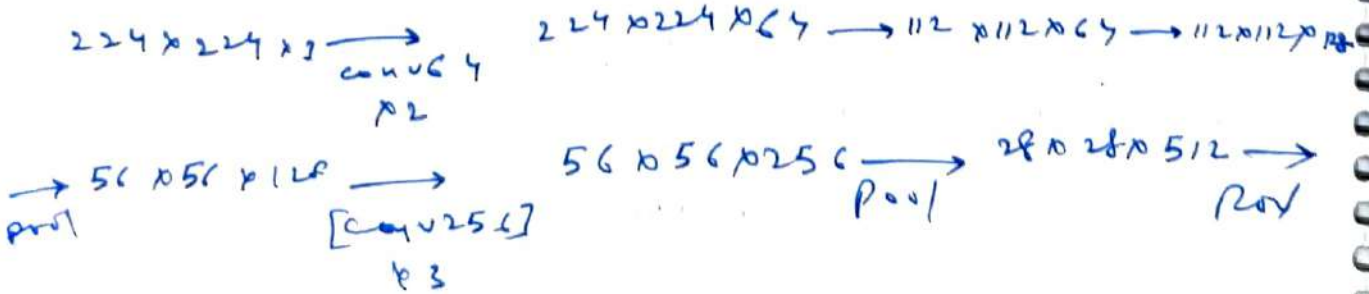
Alex Net



VGG-16

Conv = 3×3 filter, s=1, same

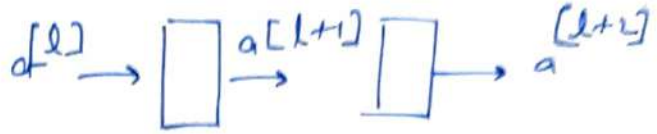
Max pool = 2×2, s=2



→ 14×14×512 → 14×14×512 → 7×7×512 → fc → fc → softmax
Pool [Conv 512] ×3 Pool 4096 4096 1000

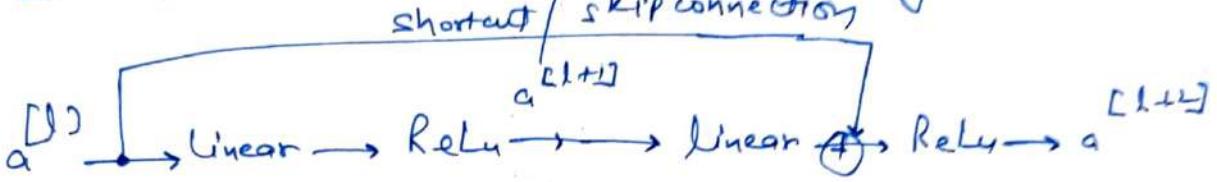
ResNet Residual network

Residual block



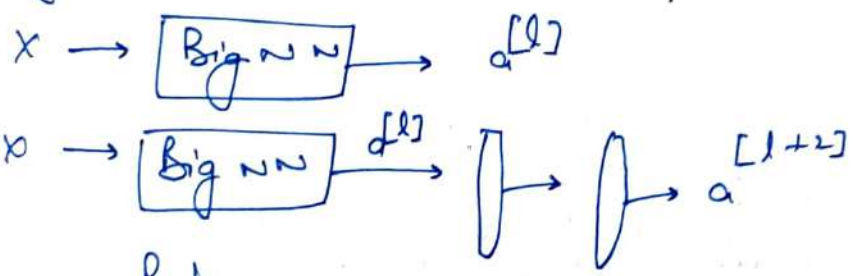
$$z^{[l+1]} = w^{[l+1]} a^{[l]} + b^{[l+1]} \qquad a^{[l+1]} = g(z^{[l+1]})$$

$$z^{[l+2]} = w^{[l+2]} a^{[l+1]} + b^{[l+2]} \qquad a^{[l+2]} = g(z^{[l+2]})$$



Why ResNets work

Why do residual networks work?



ReLu $a \geq 0$

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

$$= g(w^{[l+2]} a^{[l+1]} + b^{[l+2]} + a^{[l]})$$

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

— Network in network (1×1 convolution do)

— Inception network

- Using open source implementation

→ Data Augmentation.
- Mirroring
- Random cropping
- Rotation
- shearing
- Local warping ....

color shifting

Advanced:
PCA
PCA color Augmentation.

Implementing distortions during training

→ state of computer vision
Data Vs hand engineering

→ Object Localization

what are localization and detection?

① Image classification, classification with localization, Detection

b_x, b_y, b_h, b_w

- Defining the target label y

eg 2

$$\begin{bmatrix} 1 \\ b_x \\ b_y \\ b_h \\ b_w \\ 0 \\ 1 \\ 0 \end{bmatrix} \quad \begin{bmatrix} 0 \\ ? \\ \vdots \\ ? \\ 1 \\ \vdots \\ ? \end{bmatrix}$$

1 if object is in image

don't object

$$y = \begin{bmatrix} P_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \end{bmatrix}$$

is there any object

$$L(\hat{y}, y) = (\hat{y}_1 - y_1)^2 + (\hat{y}_2 - y_2)^2 \cdots (\hat{y}_f - y_f)^2$$

→ Landmark detection

→ sliding windows detection

→ Convolutional Implementation sliding windows
Sermanet et al, 2013 Overfeat: Integration, localization, and detection using convolution networks]

→ Bounding Box. predictions
— Yolo algorithm.
- labels for training
- for each grid cell
label for training for each gridcell:
- assign mid point of gridcell
- relative to grid cell
- Intersection over union
Intersection over union
= $\frac{\text{size of intersection}}{\text{size of union}}$
correct IOU if IOU ≥ 0.5
- IOU measure of the overlap b/w two bounding box
- non max ~~supression~~ supression
- Discard all boxes with $P_c \leq 0.6$
- Anchor boxes
for multiple objects
- predefined two shapes. Anchor boxes (1, 2)
- Yolo algorithm
- Region proposal
R CNN
Propose Regions. classify proposed regions one at a time. Output label + bounding box
Fast R-CNN
faster R-CNN