PLEASE ONLY USE GLASS BOARD MARKERS

PLEASE DO NOT WRITE ON THE WALLS

CNN— CONVOLUTIONAL NEURAL NETWORKS

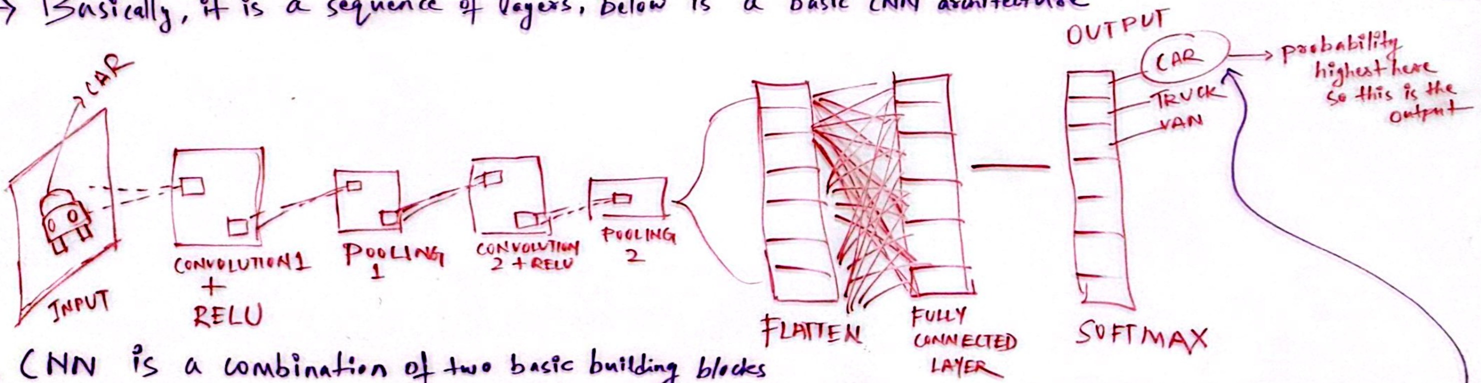
→ SPECIAL TYPE of Neural network suited for analysing visual data (images, videos)

→ It works by extracting relevant features via convolution and pooling layers and classify or detect [illegible]

→ Examples of data processed by CNN include time-series data (1D grid of samples at regular time intervals) and image data (2D grid of pixels)

→ Very similar to ordinary neural networks — made of neurons with learnable weights and biases, that are updated every iteration and lo gets minimised.

→ Basically, it is a sequence of layers, below is a basic CNN architecture



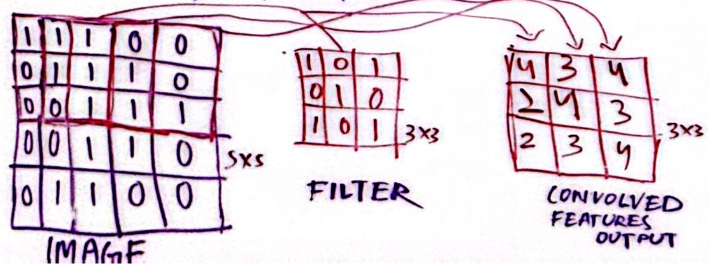
→ CNN is a combination of two basic building blocks

1. Convolution block — consists of the convolution and pooling layer. Forms the essential component of feature extraction.
2. Fully connected block — consists of a fully connected simple neural network architecture. Forms essential component for classification

1. CONVOLUTION BLOCK

→ Convolution — Special operation applied on the input (image) matrix using another matrix (filter) to extract a particular feature. The operation involves multiplying the values of a cell corresponding to a particular row and column of the image matrix, with the value of the corresponding cell in the filter matrix. This is repeated for the values of all cells within the span of the filter matrix then added together to form the output.

Eg:—

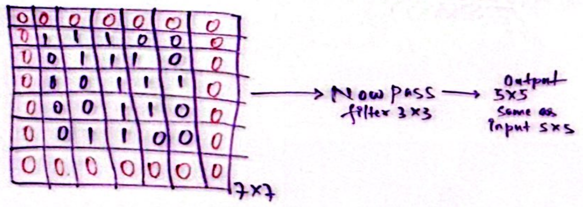


$$\underbrace{\begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 \end{bmatrix}}_{\text{IMAGE}}{}_{5\times5} \quad \underbrace{\begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix}}_{\text{FILTER}}{}_{3\times3} \quad \underbrace{\begin{bmatrix} 4 & 3 & 4 \\ 2 & 4 & 3 \\ 2 & 3 & 4 \end{bmatrix}}_{\text{CONVOLVED FEATURES OUTPUT}}{}_{3\times3}$$

→ So, for this process, we have as our input say a colored image i.e 2D RGB matrix. We also have no. of filter matrix
we take each filter matrix, on .... the corresponding part of the image-matrix of all the red green blue channel matrices and and add the values from each channel together to form the value of cell of the output matrix

→ Padding :- When convolution is performed, information of the input image at the borders and corners is lost. To overcome this extra pixels (typically 0s) are added around the edges of the input image. This is known as padding. It helps maintain the same dimensions for the output as the input. Basically prevents information loss.

Eg:-



→ Stride - The number by which the filter is shifted. If this is increased the no. of computation decreases, which also reduces the size of the output.

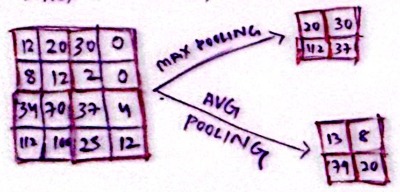
→ RELU Activation — RELU or rectified linear unit is applied in all the cells of all the output matrix

$$\text{RELU} \longrightarrow f(x) = \begin{cases} 0 & \forall\, x < 0 \\ x & \forall\, x \geq 0 \end{cases}$$

Basic intuition behind this after convolution, if a particular convolved feature output results in 0 or -ve value, it implies the feature is not present and is denoted as 0

→ POOLING - It is the next procedure of extracting a particular value from the convolution output, usually the max value or the average value. This further reduces the size of the output matrix

Ex— MAX POOLING, AVERAGE POOLING



CONVOLUTION LAYER + POOLING LAYER forms the convolutional block of the CNN architecture.

Generally, the CNN architecture consists of a minimum of three of these convolutional block that performs the feature extraction at various levels.

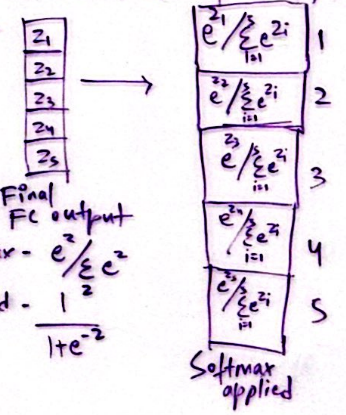
2. FULLY CONNECTED BLOCK

Output of the final pooling layer is flattened (1D vector) which forms the origin of the fully connected layers.

→ Fully connected layer - This layer forms the last block of the CNN architecture whose objective is to classify the output. This consists of two or three hidden layers and the output layer that uses softmax function for classification among a large no. of categorie
Basically the flattened vector is fed into one or more fully connected layer (FC). Each neuron in an FC layer takes the input vector, applies weights, adds a bias and passes it through an activation function i.e ReLU. This is done to introduce non-linearity that helps the model capture complex patterns of the datasets. These FC layers learn high-level patterns and combinations of the features.

→ Output layer - The last FC layer that has a no. of neurons equal to the no. of classes for classification.
For multiclass Classification, softmax activation is applied. This converts the raw output scores into probabilities

Ex -


z_1
z_2
z_3
z_4
z_5
Final FC output
$e^{z_1}/\sum_{i=1}^{5} e^{z_i}$
$e^{z_2}/\sum_{i=1}^{5} e^{z_i}$
$e^{z_3}/\sum_{i=1}^{5} e^{z_i}$
$e^{z_4}/\sum_{i=1}^{5} e^{z_i}$
$e^{z_5}/\sum_{i=1}^{5} e^{z_i}$
1
2
3
4
5
Softmax applied

Class of the input - $\arg\max_i \; e^{z_i} / \sum_{j=1}^{n} e^{z_j}$

Eg:- Input - car image
output probability of car class is 0.8
which is good enough

Softmax - $e^{z} / \sum e^{z}$

Sigmoid - $\frac{1}{1+e^{-z}}$

→ For binary classification, a sigmoid activation activation f^n for probability is used that outputs it between 0 and 1. If $\frac{1}{1+e^{-z}} > 0.5$ class 1
else $\frac{1}{1+e^{-z}} < 0.5$ class 0

→ Loss function - measures how well the model predicted comparing them with the ground truth (labels).
For multi-class, cross-entropy loss is used. $L = -\sum_{i=1}^{C} y_i \log \hat{y}_i$

This loss is for one example, usually we take the average of this loss considering all examples.

y_i - true label for class i (1 if its the correct class, 0 otherwise)
$\hat{y}_i$ - Predicted probability for class i (softmax output)
C - No. of classes

Objective - minimize this loss using backpropagation and steepest gradient descent method to get the updated weights and bias for each layer.

Popular CNN architectures
1. LeNet
2. AlexNet
3. VGG-16
4. ResNet