→ The Autoencoder can effectively summarize data points when all relevant info is present in the individual data points themselves.

Chapter 1

1) Vanilla Deep Neural networks

Deep learning aims to automate feature selection, crucial for computer vision tasks.

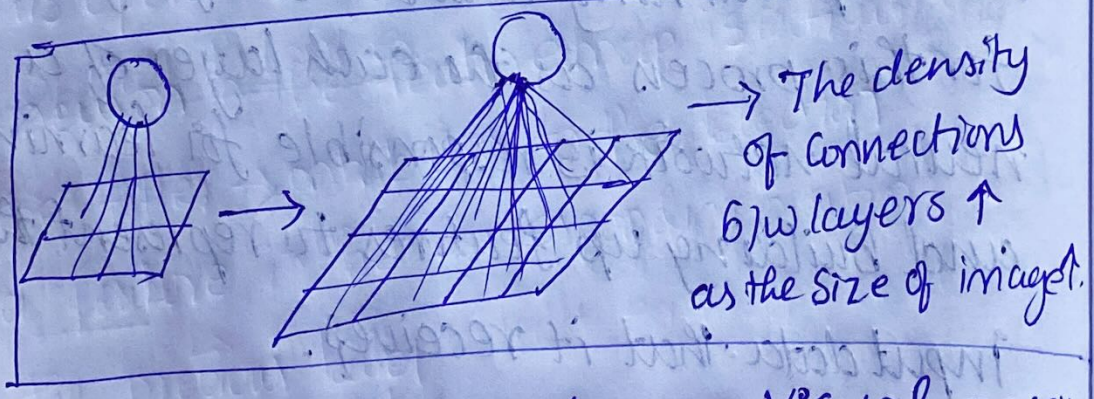
The Deep learning network are prefects for this process as the each layer of a neural network is responsible for learning and building up features to represent the input data that it receives.

The Vanilla Deep neural network are not scalable for image classification as image size increases due to an exponential increase in number of weights.

In the MNIST dataset, the problem appears manageble with 28x28 Pixel images, but Scaling up to larger images rapidly increases the complixity.

for large image, such as 200x200 full-color images, The no of weights required for just the input layer becomes high, leading to high risk of overfitting and ineffiriency.

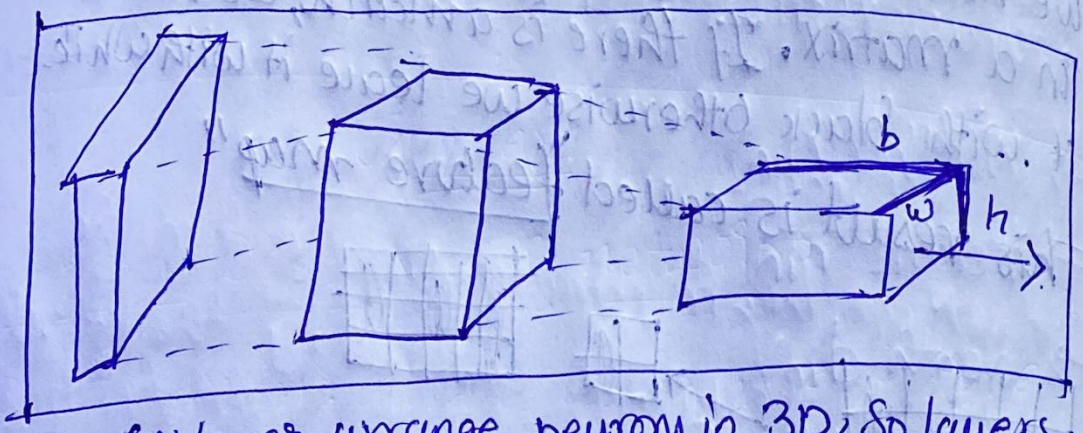
The CNN addres this Scalability issue by arranging neurons in 3D and connecting each neuron to only a Small, localized region of the previous layer.



This architecture mimics human visual processing and reduce the no of parameters significantly.

CNN manage the complexity by processing info in a 3D; So layers have width, length & depth; So new volumes of Processed info layer by layer.

This design of CN layer avoids the problem of full connectivity and reduce the potential for overfitting while handling larger images more efficiently.



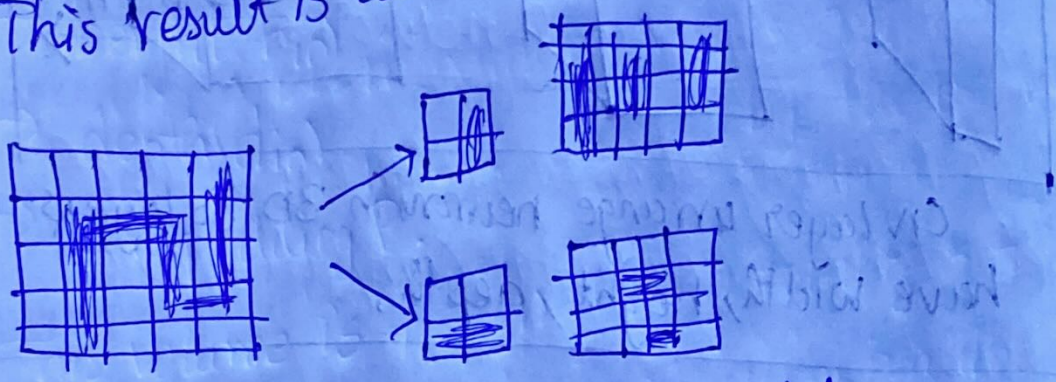
CN layer arrange neurons in 3D, so layers have width, height, depth.

Filters and Feature Map:

The visual cortex layers structures, where each layer builds upon the features detected by the previous one, inspired the design of CN layers in neural networks.

The filters is essentially a feature dector that slide across the image to identify features like edges, lines & angles.

To detect Vertical lines, we would use the feature dectors on the top, Slide it across the image and at Each and Every step check if we have a match. we keep track of answer in a matrix. If there is a match, we shade it with black otherwise we leave it with white. This result is called "feature map"



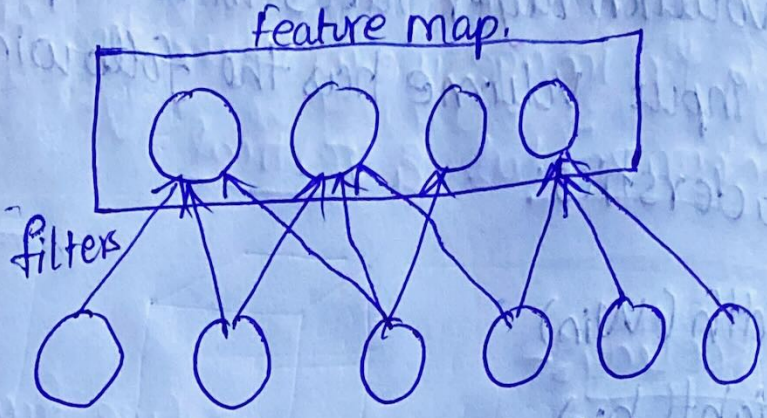
This operation is called Convolution.
A neuron in the feature map is activated if the filter contributes to the detection of a feature in the corresponding position in the previous layer

---

Expressing the Convolution Operation as neurons in a network

Layers of neurons in a feed forward neural net represents either the orginal

image or a feature map that get replicated across the Entirety of the input.



Let's denote the k^{th} feature map in layer m as m^k. $W \rightarrow$ corresponding filters weight. $b^k \rightarrow$ biaf. we mathimatically Expres feature map as:

$$m_{ij}^k = f\left((W * x)_{ij} + b^k\right)$$

The fundamental operation of Slides filters across the input data to produce features, maps, capturing more Complex features. They capture Patterns like edges or textures.

filters in CNN have depth, allowing them to Process multi-channel input, such as Colour Images.

The CNN Converts one Volume of values into another Volumes of values.

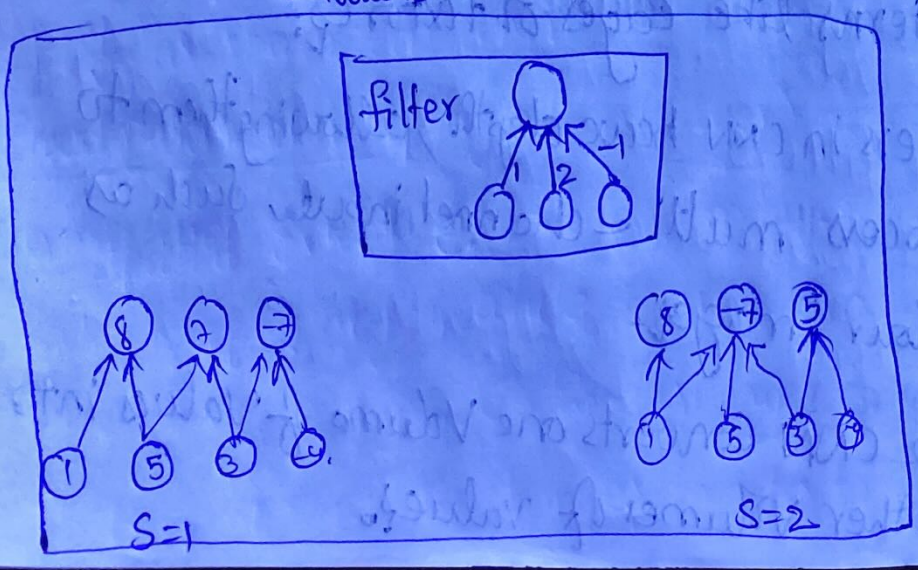
Full Description of Convolution layer:

a Convolution layer takes in a input Volume, This input volume has the following Characterstics:

→ Width (Win)
→ height (hin)
→ depth (din)
→ Zero padding P.

This volume is processed by a total of K filters, This filters has many no of hyper parameters:

→ Spatial Extent e → which is equal to the filters height & width.
→ Stride S → distance b/w consective applications of the filters on the input volume.
→ bias b → which is added to each component of the Convolution.



The Result in the output volume has :

→ function f,

→ Width $W_{out} = \left[\frac{W_{in} - e + 2p}{S}\right] + 1$

→ height $h_{out} = \left[\frac{h_{in} - e + 2p}{S}\right] + 1$

→ depth $d_{out} = k$

w=5
h=5
d=3
Zero padd=1

w=3
h=3
d=2.

Padding :-

If the input is n x n and filter size is f x f the output size will be (n-f+1) x (n-f+1)

There are two disadvantages:

→ Every time we apply a convolution Operation, the Image size shrinks

→ Pixels present in the corners of the images are used only a few number of times during convolutional as compared to center pixels.

To overcome the issue we can pad the image with additional borders. We add one pixel around the edges. This means input size will be 9 x 9 instead of 7x7.

input : n x n
Padding : p
filter size : f x f

output : (n+2p-f+1) x (n+2p-f+1)

Striding

No of steps taken for convolution.

Suppose we choose a stride of 2. So while convolving through the image, we will take two steps: both in horizontal & vertical

---

Max Pooling:-

It returns the maximum value from the portion of the image covered by the kernel.

~~We can describe~~

The pooling layer is responsible for Reducing the spatial size of the convolved feature.

This is to decrease the computational power required to process the data through dimensionality reduction and hence speed up the computation.

↓

two parameters:

1) Spatial Extent "e".

2) Stride "s".

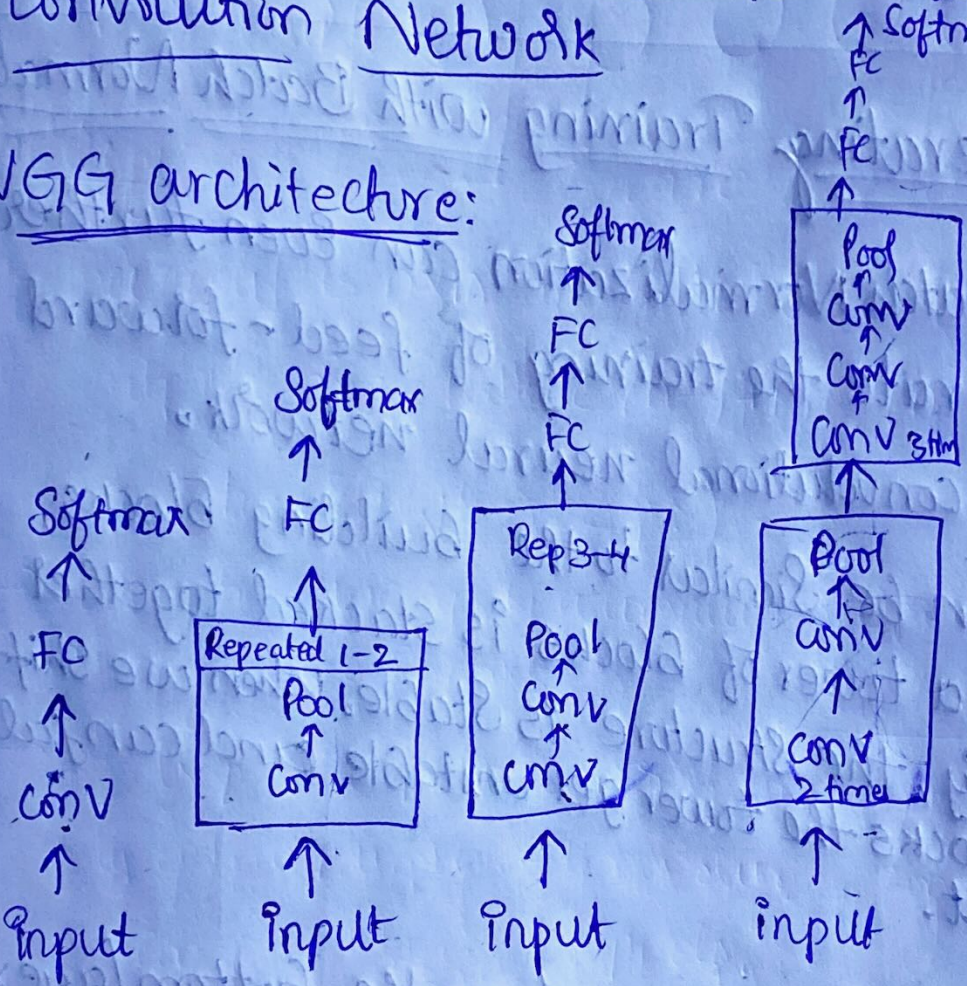
Width $W_{out} = \left[\frac{W_{in} - e}{s}\right] + 1$.

Height $H_{out} = \left[\frac{h_{in} - e}{s}\right] + 1$.

(FYI:)

Full Architectural Description of Convolution Network

VGG architecture:



FC → Flatten

The VGG architecture is a deep convolution neural network known for its simplicity and its deep stack of convolution layers with

Small filters.

Input layer: The input layer would be size 28x28x3, assuming the images are RGB.

Link

Accelerating Training with Batch Normaliz

The batch Normalization can even further accelerate the training of feed-forward and convolutional neural network.

It can be similar to the building blocks, when a tower of blocks is stacked together neatly, the structure is stable. When we shift the blocks the tower get unstable and can fall a part.

A similar phenomeon where the bottom layer (first layer) and top layer (last layer) happens, when we give data to the bottom layer the data keep changing while passing from bottom layer to upper layers. So the last layer has to keep adapting to the changes in order to give

~~UNIT-III~~

1) Collect output. So the more the layers we have the more problem we get.

Normalization of image inputs helps out the training process by making it more robust to variations. Batch Normalization takes this a step further by normalizing inputs to every layer in our neural network.

We modify our network to include the below operations for batch Normalization'.

1) Grab the vectors of bias which is incoming to a layer before they pass through the nonlinearity.

2) Normalize each component of the vector of bias across all Examples of the minibatch ~~by subtracting the~~ m.

3) Given Normalized input $\hat{x}$, use an affine transform to restore representation power with 2 Vectors of Parameter: $\gamma\hat{x} + \beta$.

Batch Normalization also allows us to Significantly ↑ the learning rate.