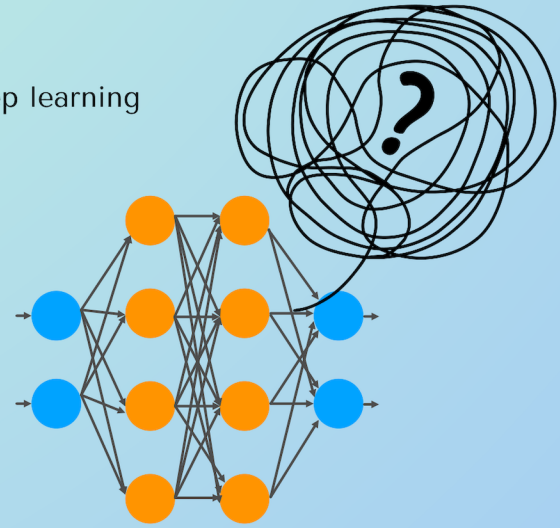
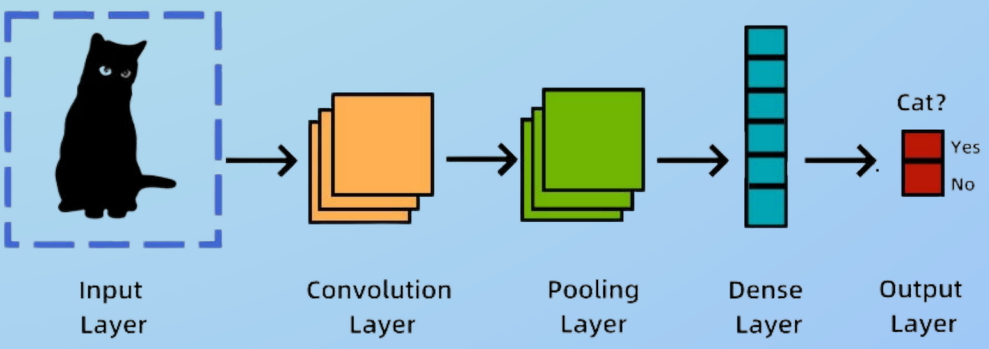
Fundamentals of Deep Learning

The need for Convolutional Neural Networks

When simple machine learning and deep learning models are not enough.





Shivang Kainthola

The I.I.D assumption of data

Neural networks and machine learning models work under the assumption that the data is independent and identically distributed.

⟶ In independent and identically distributed data (I.I.D data), the outcome of one data point does not influence any other data point.

⟶ All data points come from the same probability distribution . This means they are governed by the same statistical properties, such as the same mean and variance.

This assumption is considered for the data, it is ignorant of any domain knowledge, background, context or patterns that may occur within or in the source of data.

The IID assumption is an ideal scenario, and real-world data might not perfectly adhere to it.

For example,

if you roll a fair six-sided die multiple times, each roll has the same probability distribution of outcomes, just like drawing a card from a well-shuffled deck (each card has an equal chance of being picked.

But when processing things like :

1) ***Temperature***, every subsequent reading is influenced by the previous readings, as well as yearly patterns, seasons, climate etc.

2) ***Customer reviews*** :

Customer for a product might be influenced by recent events or promotions. For example, a sudden influx of negative reviews could be due to a recent product defect, violating the independence assumption.

3) ***Stock prices*** : Stock prices are not independent of each other.

Today's closing price can influence tomorrow's opening price, creating a dependence. Additionally, there might be seasonal trends or external events impacting the overall market, further violating the identical distribution assumption.

4) ***Images or Videos*** :

Images and videos have inherent spatial relationships between pixels (e.g., edges forming objects), which can be missed by treating pixels in isolation as I.I.D data.

Real-world images and videos can have variations in lighting, camera angles, or scene composition. These variations can disrupt the model's understanding if it expects a perfectly uniform distribution.

In this manner, when a machine learning model applies logistic regression to predict the presence of heart disease, it relies on the I.I.D assumption for the dataset :

a) One patient's medical history and disease prediction outcome **is not dependent on the next or previous** patient, or any other patient for that matter.

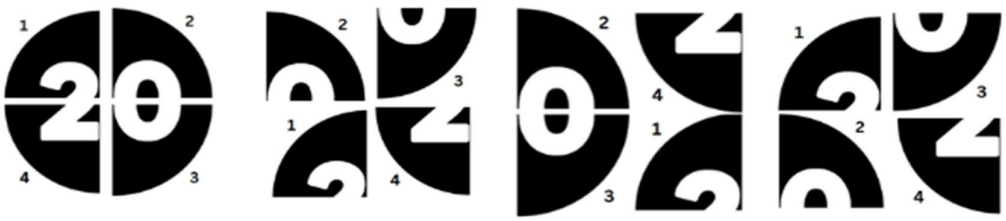
b) The data should represent the general population you want to predict for. It shouldn't be skewed towards a specific age group, location, or socioeconomic background.

Now, if you feed an image to a machine learning/ deep learning model that **works on the I.I.D assumption**, we face some problems :

The image processed as pixels. - [P1,P2,P3,P4]

Since it assumes the data i.e. the pixels, is independent and identically distributed, it may not retain the *original order and position of the pixels*.

1)***Without the spatial relationship of the pixels*** intact, the pixels arranged or processed in random orders will be difficult to work with, because *the image is only formed if the pixels are arranged in a specific order.*



2) The sheer number of parameters

One color image (RGB) of 200x200 size will have (200x200x3) = 120000 features

Further, in a fully connected neural network,

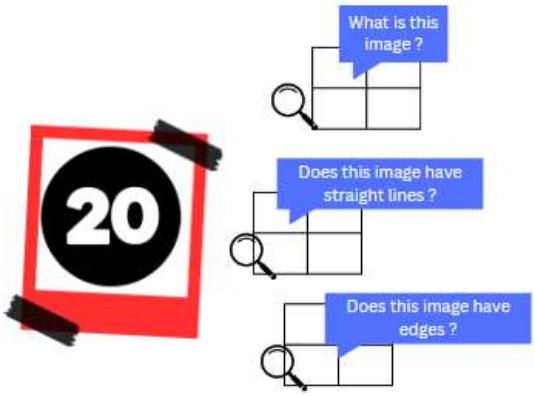
A single layer would need to have (200x200x3)^2 = 14400000000 weights !

⟶ With images of larger sizes, and working with collections of images, the processing will only become more complex.

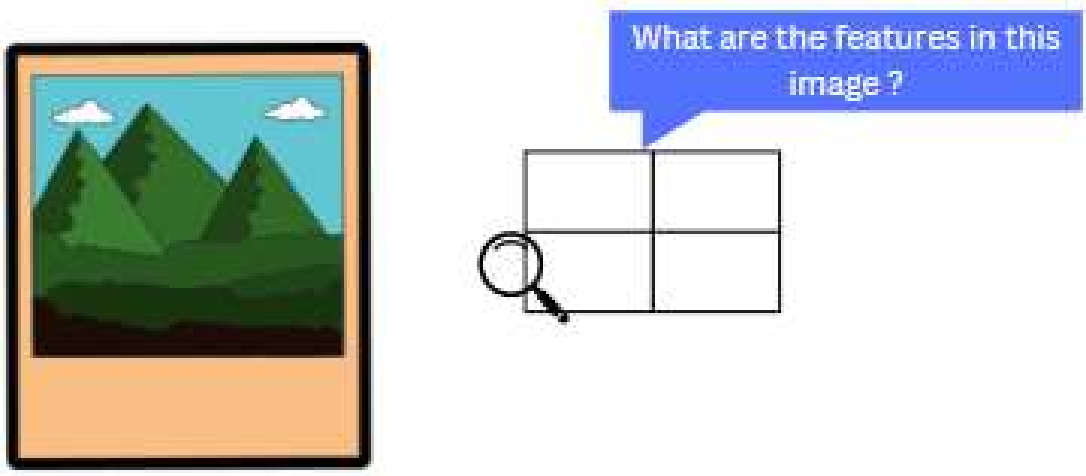
Therefore, using machine learning models that work on the I.I.D assumption to work on image or video data may not be ideal.

Then what do Convolutional Neural Networks do ?

The key element in the working of Convolutional Neural Networks is **the Kernel**.



A kernel is a grid of weights that can detect features in an image, from its pixel values.



These features can be basic shapes like edges (horizontal, vertical, diagonal), corners, blobs, outlines or textures (smooth, grainy, etc.).

From identifying simple features, the model can detect complex elements like objects, people etc.

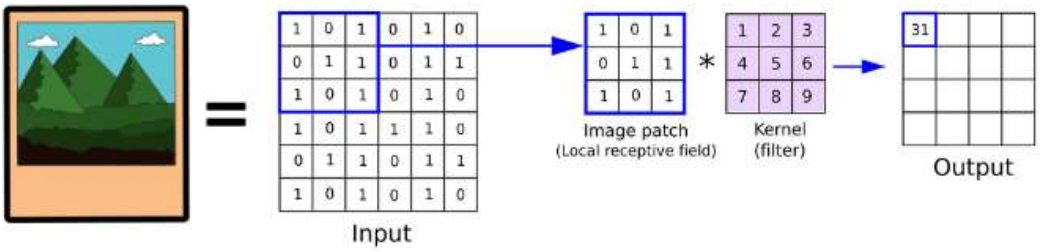
And if the network can identify objects, structures, faces etc. , it can be used for classification or other related tasks.

How do the Kernels work ?

Obtaining the pixel representation of the image, the kernel is centered on one pixel at a time, each weight of the kernel is multiplied by the pixel value.

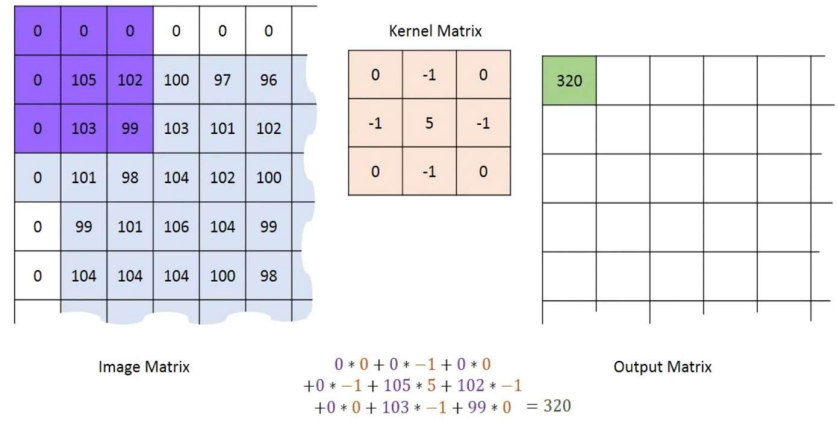
This process is called convolution.

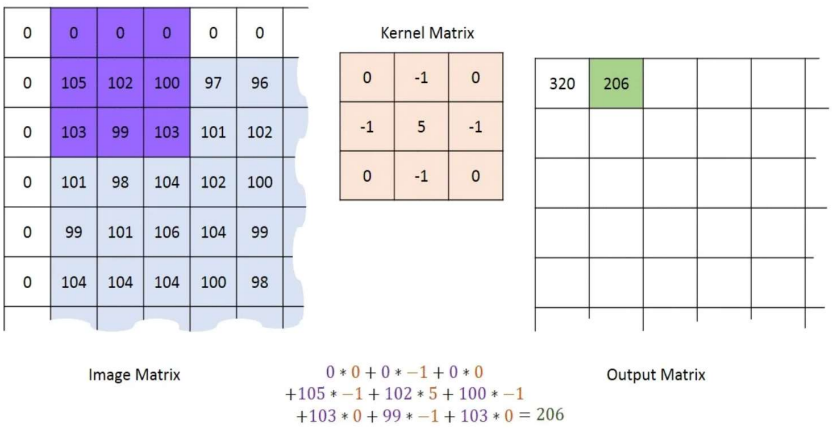
(Image Source : Convolutional Neural Network (Siddharth Sankhe; edits added)

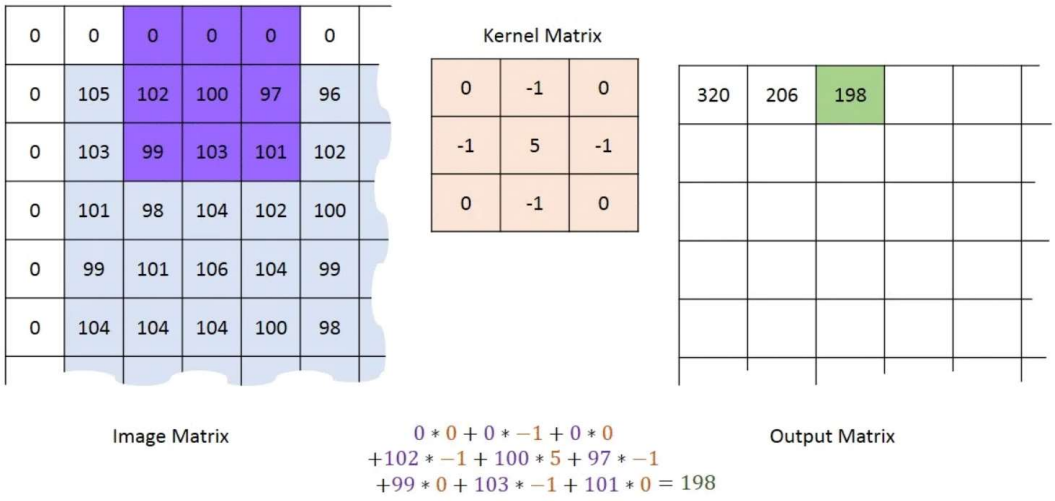


By sliding across the image, and processing each and every pixel, the kernel processes the entire image and identifies features :

https://qph.cf2.quoracdn.net/main-qimg-9c308b74774a2b7dacf0fa21a2c3f1c1







Different kernels can be used for detecting different types of features like :

$$\begin{Bmatrix} -1 & 1 & -1 \\ -1 & 1 & -1 \\ -1 & 1 & -1 \end{Bmatrix} \qquad \begin{Bmatrix} -1 & -1 & -1 \\ 1 & 1 & 1 \\ -1 & -1 & -1 \end{Bmatrix} \qquad \begin{Bmatrix} -1 & -1 & -1 \\ 1 & 1 & 1 \\ -1 & 1 & 1 \end{Bmatrix}$$

Vertical line detectors | Horizontal line detectors | Detecting corners

⟶ By using a variety of kernels, the CNN can progressively build up a complex understanding of the image by identifying these basic building blocks.

The Convolutional Layer

⟶ The application of kernels is done in a layer called Convolutional layer , and the output of a convolution operation is a feature map.

A feature map is a two-dimensional array or grid of numbers resulting from applying a convolutional filter (also called a kernel) to an input image or a previous layer's feature map.

Unlike fully connected neural networks, a neuron in the Convolutional layer is concerned only with the application of a kernel to one specific region of the image.

Primary ideas behind Convolutional Neural Networks :

1) **Capture and represent different features detected by the filters** (kernels) at various levels of abstraction.

Allow the network to learn hierarchical features, from simple edges in early layers to complex patterns in deeper layers.

Early Layers : Detect simple features like edges, corners and textures.

Middle Layers : Detect complex features like shapes and patterns.

Deep Layers : Detect high-level features like objects, or parts of objects.

2) Use the same set of kernels across the entire image

If a feature (like an edge or a specific object) is detected by a kernel at a particular location in the image, then shifting the object in the image will cause the same feature to be detected at a corresponding shifted location in the output.

This property is known as **translation equivariance**.

3) Let the neural network learn and identify which kernels and their outputs are more useful.

The CNN will iteratively refine the weights of all kernels based on how they contribute to the overall task. This learning process allows the network to **extract increasingly meaningful features from images**.

This approach offers a much better alternative to handling images or videos and introduces the concept of convolutional neural networks.

Thank You !

If you found this article helpful, please do leave a like and comment any feedback you feel I could use.

While I’m working on the next project, I’ll be over at :

www.linkedin.com/in/shivang-kainthola-2835151b9/

shivang.kainthola64@gmail.com

https://shivangkainthola28.medium.com/

https://github.com/HeadHunter28