**Introduction to CNN**

Yann LeCun, director of Facebook’s AI Research Group, pioneered convolutional neural networks. In 1988, he built the first one, LeNet, which was used for character recognition tasks like reading zip codes and digits.

Have you ever wondered how facial recognition works on social media, or how object detection helps in building self-driving cars, or how disease detection is done using visual imagery in healthcare? It’s all possible thanks to convolutional neural networks (CNN). Here’s an example of convolutional neural networks that illustrates how they work:

Imagine there’s an image of a bird, and you want to identify whether it’s really a bird or some other object. The first thing you do is feed the pixels of the image in the form of arrays to the input layer of the [neural network](https://www.simplilearn.com/tutorials/deep-learning-tutorial/what-is-neural-network) (multi-layer networks used to classify things). The hidden layers carry out feature extraction by performing different calculations and manipulations. There are multiple hidden layers like the convolution layer, the ReLU layer, and pooling layer, that perform feature extraction from the image. Finally, there’s a fully connected layer that identifies the object in the image.

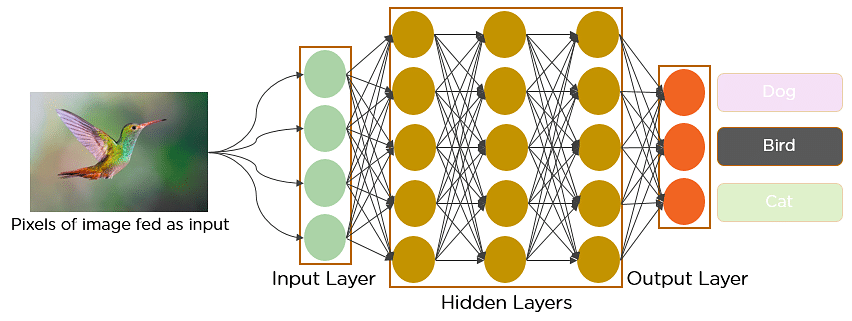
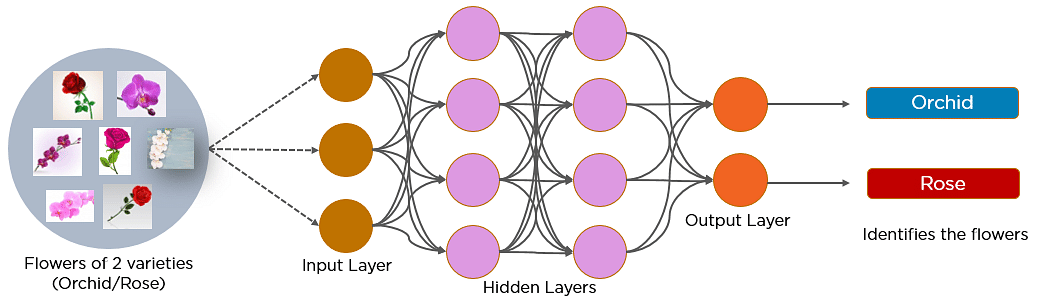


Fig: Convolutional Neural Network to identify the image of a bird

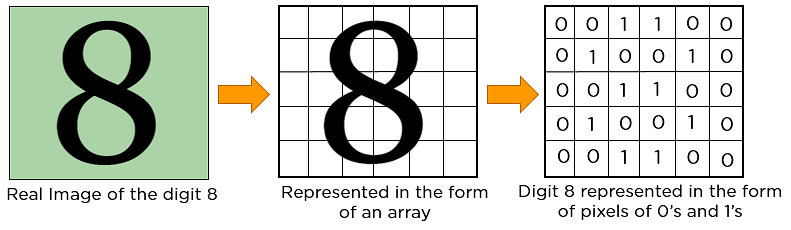
What is Convolutional Neural Network?

A convolutional neural network is a feed-forward neural network that is generally used to analyze visual images by processing data with grid-like topology. It’s also known as a ConvNet. A convolutional neural network is used to detect and classify objects in an image.

Below is a neural network that identifies two types of flowers: Orchid and Rose.



In CNN, every image is represented in the form of an array of pixel values.



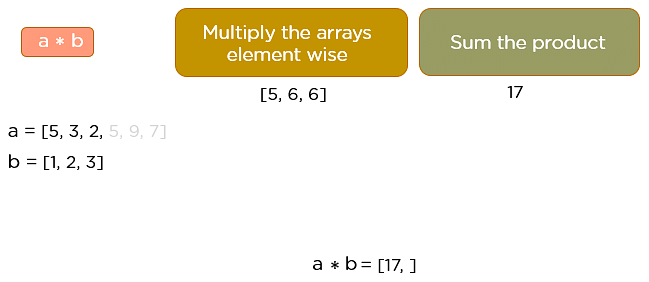
The convolution operation forms the basis of any convolutional neural network. Let’s understand the convolution operation using two matrices, a and b, of 1 dimension.

a = [5,3,7,5,9,7]

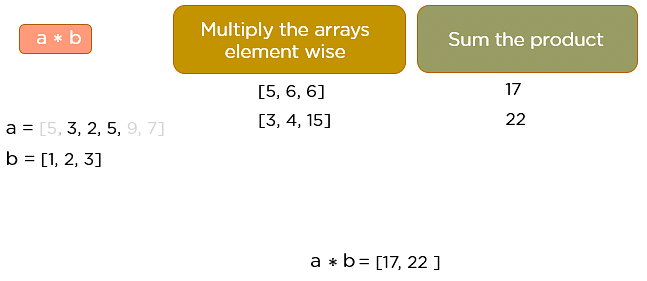
b = [1,2,3]

In convolution operation, the arrays are multiplied element-wise, and the product is summed to create a new array, which represents a\*b.

The first three elements of the matrix a are multiplied with the elements of matrix b. The product is summed to get the result.



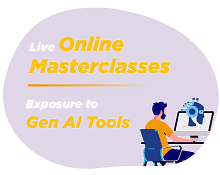
The next three elements from the matrix a are multiplied by the elements in matrix b, and the product is summed up.



This process continues until the convolution operation is complete.

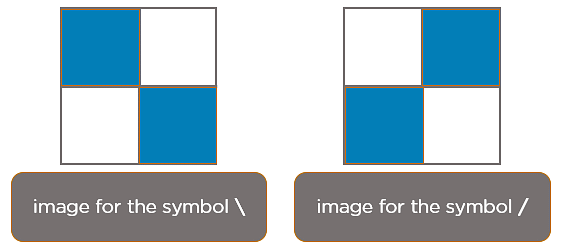
Master Gen AI Strategies for Businesses with

Generative AI for Business Transformation Program[Explore Program](https://www.simplilearn.com/generative-ai-for-business-transformation-course?source=GhPreviewCTABanner)



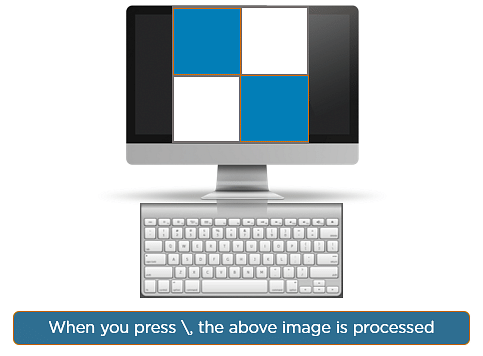
How Does CNN Recognize Images?

Consider the following images:

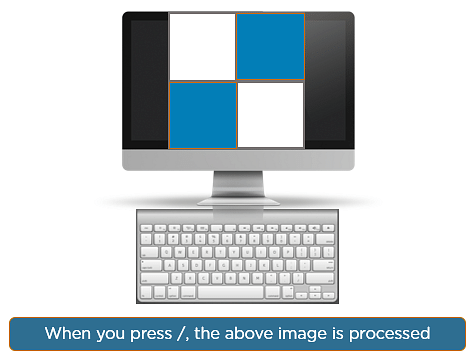


The boxes that are colored represent a pixel value of 1, and 0 if not colored.

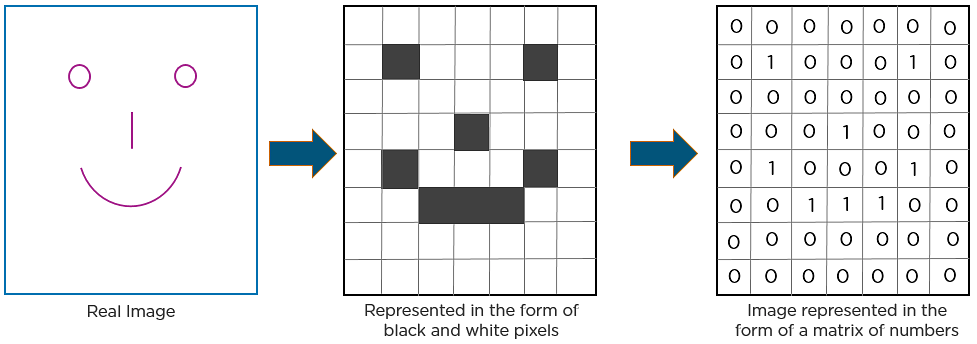
When you press backslash (\), the below image gets processed.



When you press forward-slash (/), the below image is processed:



Here is another example to depict how CNN recognizes an image:



As you can see from the above diagram, only those values are lit that have a value of 1.

Layers in a Convolutional Neural Network

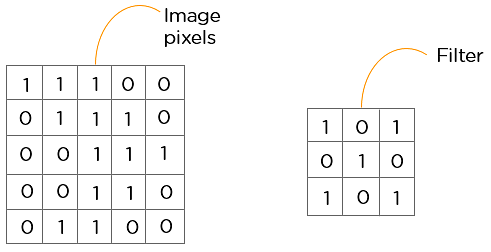
A convolution neural network has multiple hidden layers that help in extracting information from an image. The four important layers in CNN are:

1. Convolution layer
2. ReLU layer
3. Pooling layer
4. Fully connected layer
5. ReLU layer/ Activation Layer
6. Flattening
7. Output Layer

Convolution Layer

This is the first step in the process of extracting valuable features from an image. A convolution layer has several filters that perform the convolution operation. Every image is considered as a matrix of pixel values.

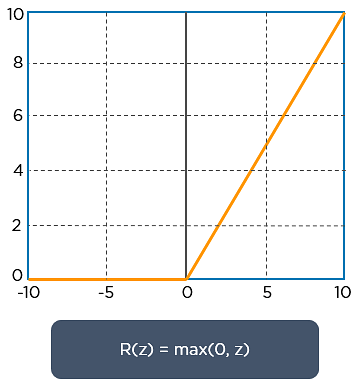
Consider the following 5x5 image whose pixel values are either 0 or 1. There’s also a filter matrix with a dimension of 3x3. Slide the filter matrix over the image and compute the dot product to get the convolved feature matrix.



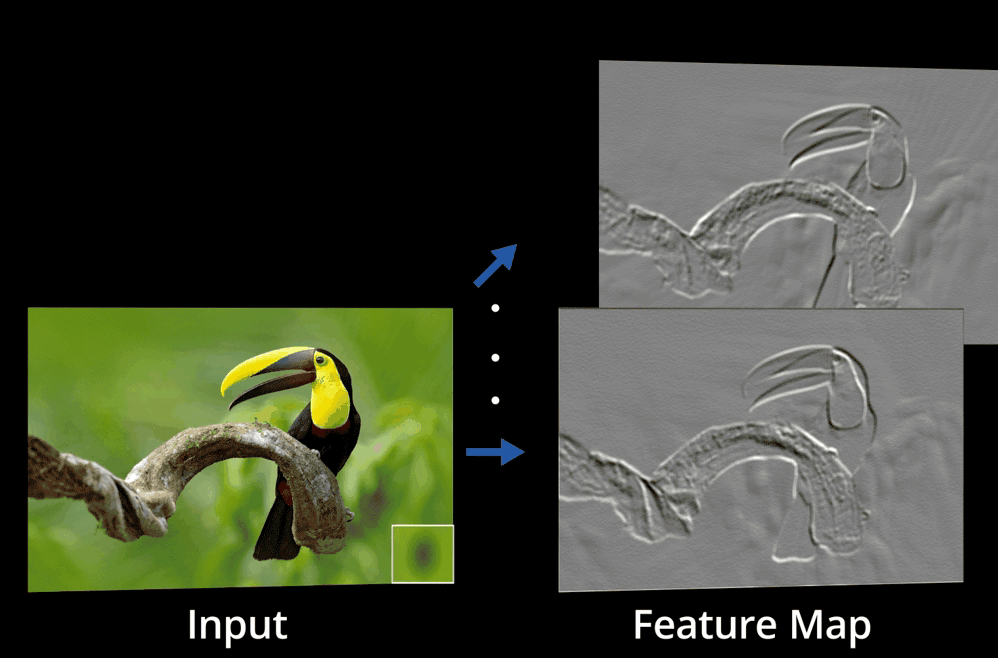
ReLU layer

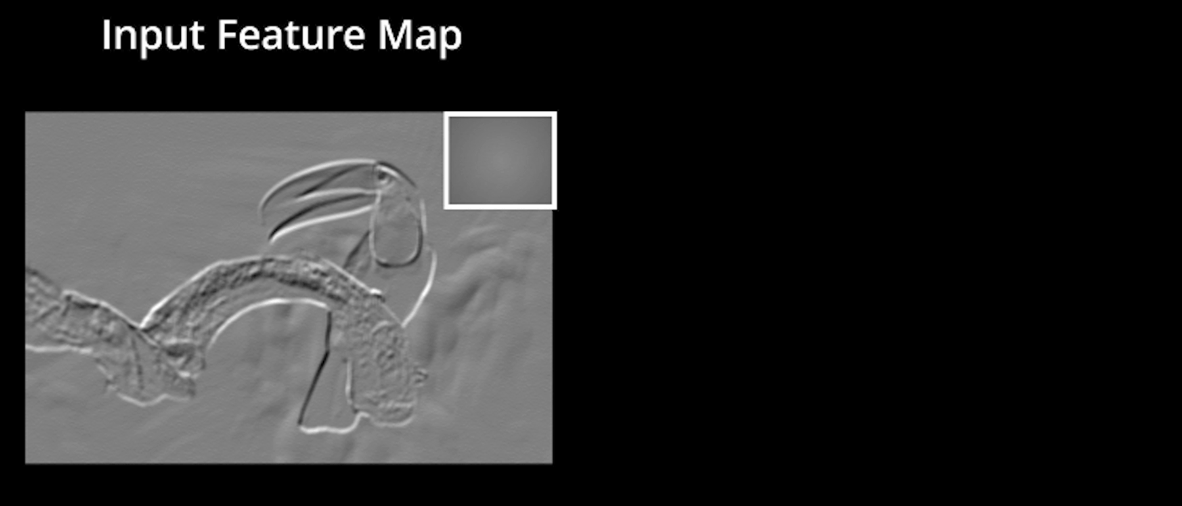
ReLU stands for the rectified linear unit. Once the feature maps are extracted, the next step is to move them to a ReLU layer.

ReLU performs an element-wise operation and sets all the negative pixels to 0. It introduces non-linearity to the network, and the generated output is a rectified feature map. Below is the graph of a ReLU function:



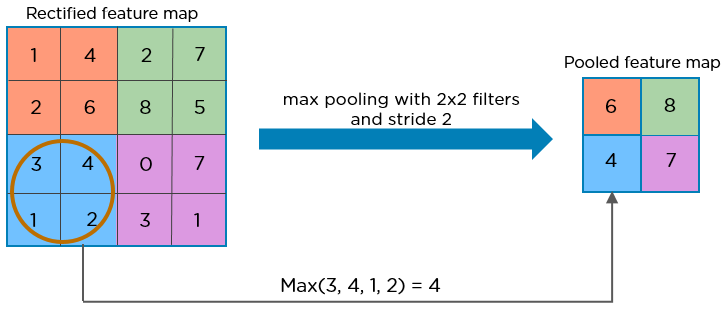
The original image is scanned with multiple convolutions and ReLU layers for locating the features.



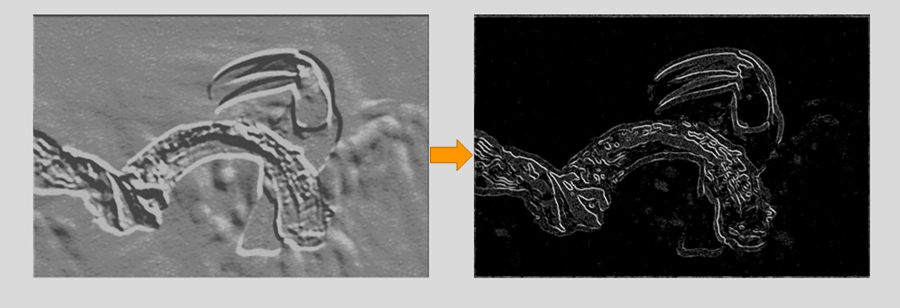


Pooling Layer

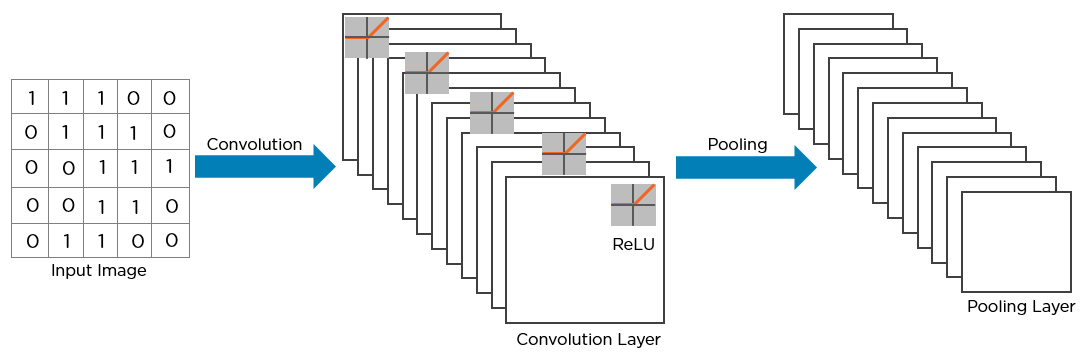
Pooling is a down-sampling operation that reduces the dimensionality of the feature map. The rectified feature map now goes through a pooling layer to generate a pooled feature map.



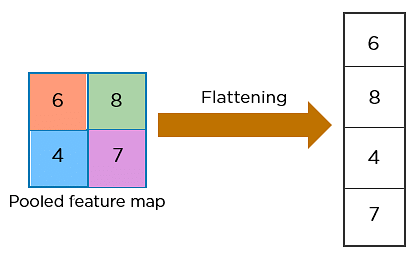
The pooling layer uses various filters to identify different parts of the image like edges, corners, body, feathers, eyes, and beak.



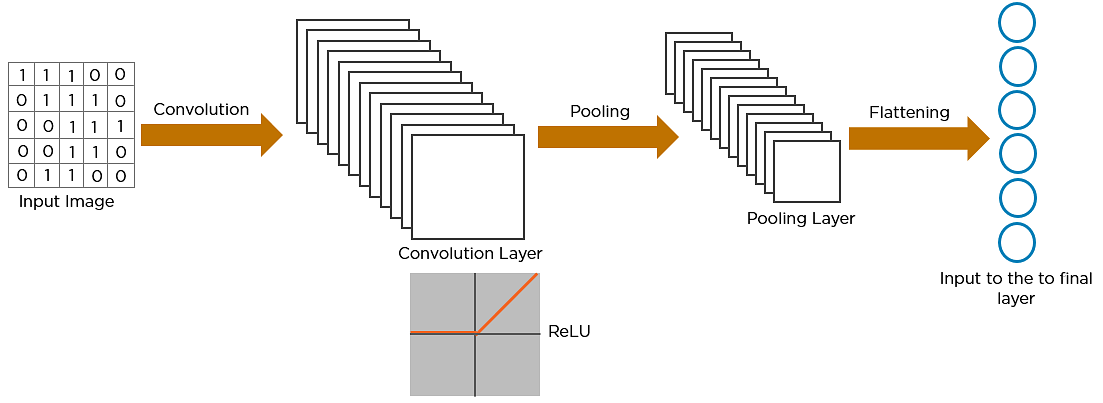
Here’s how the structure of the convolution neural network looks so far:

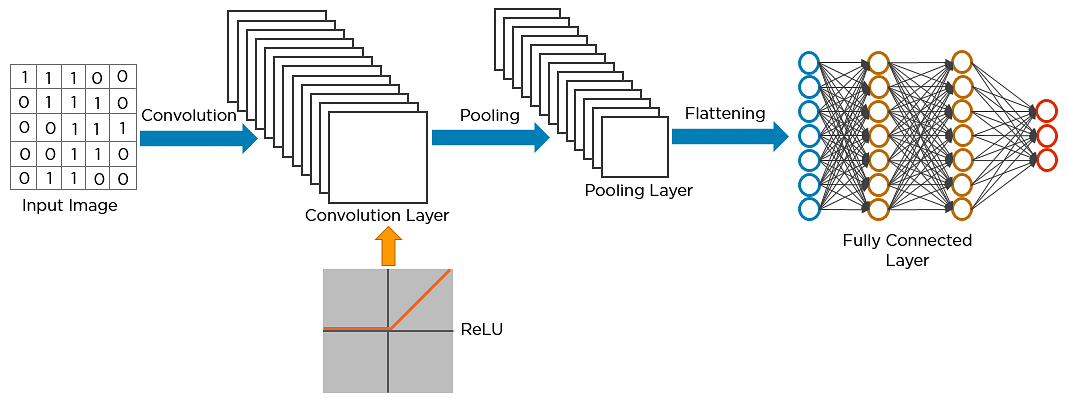


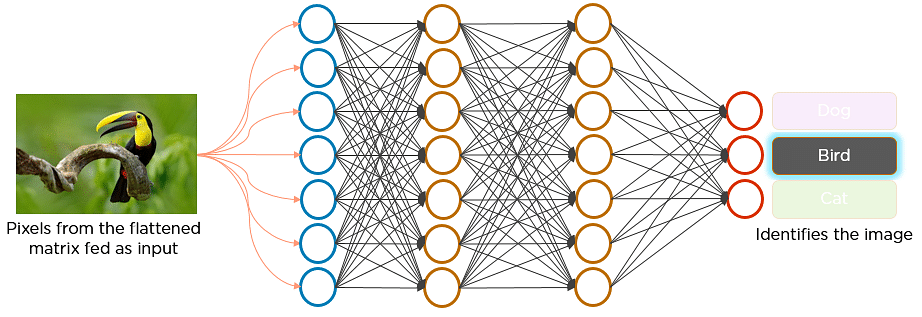
The next step in the process is called flattening. Flattening is used to convert all the resultant 2-Dimensional arrays from pooled feature maps into a single long continuous linear vector.



The flattened matrix is fed as input to the fully connected layer to classify the image.

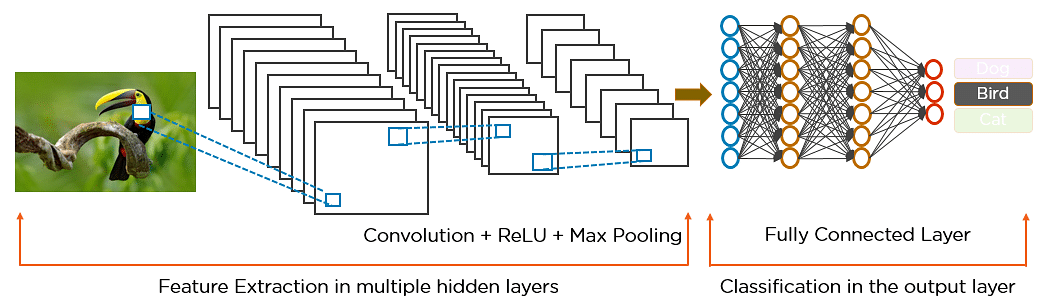






Here’s how exactly CNN recognizes a bird:

* The pixels from the image are fed to the convolutional layer that performs the convolution operation
* It results in a convolved map
* The convolved map is applied to a ReLU function to generate a rectified feature map
* The image is processed with multiple convolutions and ReLU layers for locating the features
* Different pooling layers with various filters are used to identify specific parts of the image
* The pooled feature map is flattened and fed to a fully connected layer to get the final output



* Activation Layer

The activation layer introduces nonlinearity into the network by applying an activation function to the output of the previous layer. This is crucial for the network to learn complex patterns. Common activation functions, such as ReLU, Tanh, and Leaky ReLU, transform the input while keeping the output size unchanged.

* Flattening

After the convolution and pooling operations, the feature maps still exist in a multi-dimensional format. Flattening converts these feature maps into a one-dimensional vector. This process is essential because it prepares the [data](https://www.simplilearn.com/what-is-data-article) to be passed into fully connected layers for classification or regression tasks.

* Output Layer

In the output layer, the final result from the fully connected layers is processed through a logistic function, such as sigmoid or softmax. These functions convert the raw scores into probability distributions, enabling the model to predict the most likely class label.

Convolutional Neural Network Training

Training a Convolutional Neural Network (CNN) involves guiding the model to recognize patterns in data through a step-by-step learning process. This is typically done using supervised learning, where the CNN is fed a bunch of images with their correct labels, and it gradually learns how to associate images with the right labels. Here’s how the process works:

* Data Preparation

First things first, the images need to be prepared before training can start. This means making sure all the images are uniform in terms of format and size. By preprocessing the data in this way, you ensure that the CNN gets consistent input, which is crucial for its learning process.

* Loss Function

Once the images are ready, the next step is to figure out how well the CNN is doing. This is where the loss function comes into play. Think of it as a scorecard that measures the difference between what the model predicted and the actual label of the image. The smaller the difference, the better the model is performing, so the goal is to reduce this gap as much as possible.

* Optimizer

Now that we know how well (or poorly) the CNN is performing, it’s time to improve it. The optimizer is like a coach that adjusts the network’s weights to help it do better. It tweaks the model's parameters to minimize the loss function, ultimately leading to more accurate predictions over time.

* Backpropagation

Backpropagation is the magic behind the scenes that makes everything work. It’s the process of figuring out how much each weight in the network contributed to the errors and then adjusting those weights accordingly. The optimizer uses this information to make smarter updates, helping the model get better with each round of training.

CNN Evaluation

Several key metrics are used to evaluate your Convolutional Neural Network after its training process is complete:

* Accuracy

Accuracy tells you the overall percentage of test images that the CNN correctly classifies. It’s a straightforward measure of how often the model gets the right label.

* Precision

Precision focuses on how precise the CNN is when it predicts a particular class. It measures the percentage of test images that were predicted as a specific class and actually belong to that class. High precision means that when the CNN predicts a class, it’s likely correct.

* Recall

Recall looks at how well the CNN identifies all instances of a particular class. It measures the percentage of test images that are of a certain class and were correctly identified as that class by the CNN. High recall indicates that the CNN is good at finding all relevant examples of a class.

* F1 Score

The F1 Score combines precision and recall into a single metric by calculating their harmonic mean. This is particularly useful for evaluating the CNN’s performance on classes where there’s an imbalance, meaning some classes are much more common than others. The F1 Score provides a balanced measure that considers both false positives and false negatives, offering a more comprehensive view of the CNN’s performance.

Types of Convolutional Neural Networks

Here are the key types of Convolutional Neural Networks that have significantly impacted the field of image recognition:

* LeNet

LeNet, developed by Yann LeCun and his team in the late 1990s, is one of the earliest CNN architectures designed for handwritten digit recognition. It features a straightforward design with two convolutional and pooling layers followed by subsampling, and three fully connected layers. Despite its simplicity by today’s standards, LeNet achieved high accuracy on the MNIST dataset and laid the groundwork for modern CNNs.

* AlexNet

AlexNet, created by Alex Krizhevsky and colleagues in 2012, revolutionized image recognition by winning the ImageNet Large Scale Visual Recognition Challenge (ILSVRC). Its architecture includes five convolutional layers and three fully connected layers, with innovations like ReLU activation and dropout. AlexNet demonstrated the power of [deep learning](https://www.simplilearn.com/tutorials/deep-learning-tutorial), leading to the development of even deeper networks.

* ResNet

ResNet, or Residual Networks, introduced the concept of residual connections, allowing the training of very deep networks without overfitting. Its architecture uses skip connections to help gradients flow through the network effectively, making it well-suited for complex tasks like keypoint detection. ResNet has set new benchmarks in various image recognition tasks and continues to be influential.

* GoogleNet

GoogleNet, also known as InceptionNet, is known for its efficiency and high performance in image classification. It introduces the Inception module, which allows the network to process features at multiple scales simultaneously. With global average pooling and factorized convolutions, GoogleNet achieves impressive accuracy while using fewer parameters and computational resources.

* MobileNet

MobileNets are designed for mobile and embedded devices, offering a balance of high accuracy and computational efficiency. By using depth-wise separable convolutions, MobileNets reduce the model size and computational demand while maintaining strong performance in image classification and keypoint detection. Their efficiency makes them ideal for resource-constrained environments.

* VGG

VGG networks are recognised for their simplicity and effectiveness, using a series of convolutional and pooling layers followed by fully connected layers. Their straightforward architecture has made them popular in various image recognition tasks, including object detection in self-driving cars. VGG’s design remains a powerful tool for many applications due to its versatility and ease of use.

Applications of CNN

Now, let's look at the various applications of CNN in machine learning and how they are used across different fields:

* Image Classification

CNN in deep learning excels at image classification, which involves sorting images into predefined categories. They can effectively identify whether an image depicts a cat, dog, car, or flower, making them indispensable for tasks that require sorting and labeling large volumes of visual data.

* Object Detection

CNNs are particularly skilled in object detection, allowing them to identify and pinpoint specific items within an image. Whether it's recognizing people, cars, or buildings, CNNs can locate these objects and highlight their positions, which is crucial for applications needing accurate object placement and identification.

* Image Segmentation

CNNs are highly effective for tasks that involve breaking down an image into distinct parts. Image segmentation allows CNNs to distinguish and label different objects or regions within an image. This capability is essential in fields like medical imaging, where detailed analysis of structures is required, and in robotics, where intricate scenes need to be understood.

* Video Analysis

CNNs are also adept at video analysis, where they can track objects and detect events over time. This makes them valuable for applications like surveillance and traffic monitoring, where continuously analyzing dynamic scenes helps in understanding and managing real-time activities.

Advantages of CNN

Apart from their diverse applications, here are some notable advantages of CNN in deep learning that highlight their effectiveness and versatility:

* High Accuracy

Convolutional Neural Networks are known for their exceptional accuracy in image recognition tasks. They perform impressively in areas like classifying images, detecting objects, and segmenting visuals, setting a high benchmark for performance in these fields.

* Efficiency

These networks are particularly efficient when used with specialized hardware such as GPUs. This efficiency allows CNNs to process large amounts of data quickly, which is crucial for applications that require heavy computational power.

* Robustness

Convolutional Neural Networks handle noisy or inconsistent input data with impressive resilience. Their ability to maintain performance despite data imperfections makes them dependable for real-world applications where conditions can vary.

* Flexibility

Another key advantage of Convolutional Neural Networks is their adaptability. They can be tailored to different tasks simply by altering their architecture. This makes them versatile tools that can be easily repurposed for diverse applications, from medical imaging to autonomous vehicles.

Disadvantages of Convolutional Neural Networks (CNNs)

Although Convolutional Neural Networks (CNNs) are powerful, they come with their own set of challenges:

* Complexity and Training Difficulty

CNNs are intricate, and this complexity can make them difficult to train, especially when working with large datasets. Managing and fine-tuning the layers requires a deep understanding of the architecture, making it challenging even for seasoned professionals.

* High Computational Demands

Another significant disadvantage is the high computational power required to train and deploy CNNs effectively. Advanced hardware, such as GPUs, is often necessary, which increases costs and limits access for those without these resources. This makes it difficult for smaller organizations to utilize CNNs efficiently.

* Large Data Requirements

CNNs need a large amount of labeled data to perform well. Gathering and labeling data is time-consuming and expensive. For more complex applications, such as medical imaging, the precision needed in data labeling further increases the cost and effort involved.

* Lack of Interpretability

One of the most notable challenges with CNNs is their black-box nature. It’s often difficult to understand why a CNN makes a certain prediction, which can be a significant issue in areas where decision-making transparency is important. This lack of interpretability can limit the trust placed in CNN-based systems, especially in critical applications like healthcare.