VIPS Technical Campus
योगः कर्मसु कौशलम्
IN PURSUIT OF PERFECTION

SCHOOL OF ENGINEERING AND TECHNOLOGY

Unit-IV

Deep Learning Essentials
Day 3: Convolutional Neural Networks

Faculty : Dr. Pooja Gupta
Assistant Professor
VIPS-TC

VIPS Technical Campus

योगः कर्मसु कौशलम्

IN PURSUIT OF PERFECTION

Introduction to CNNs and Model Evaluation Metrics

AGENDA

- Introduction to Convolutional Neural Networks
- What is Convolutional Neural Networks ?
- Layers in a Convolutional Neural Network
- Complete working of CNN
- Practical Implementation of CNN
- CNN Evaluation
- Types of CNNs
- Applications of CNNs
- Transfer Learning

Introduction to Convolutional Neural Networks

SCHOOL OF ENGINEERING AND TECHNOLOGY

$$(f * g)(t) = \int_{-\infty}^{\infty} f(\tau)g(t-\tau)d\tau$$

Part 1: Hospital Analogy

Imagine you manage a hospital treating patients with a single disease. You have:

- **A treatment plan:** [3] Every patient gets 3 units of the cure on their first day.
- **A list of patients:** [1 2 3 4 5] Your patient count for the week (1 person Monday, 2 people on Tuesday, etc.).

Question: How much medicine do you use each day? Well, that's just a quick multiplication:

Plan	*	Patients	= Daily Usage
[3]	*	[1 2 3 4 5]	= [3 6 9 12 15]

Multiplying the plan by the patient list gives the usage for the upcoming days: [3 6 9 12 15]. Everyday multiplication (3 x 4) means using the plan with a single day of patients: [3] * [4] = [12].

Introduction to Convolutional Neural Networks Contd…

Intuition For Convolution

Let's say the disease mutates and requires a multi-day treatment. You create a new plan: Plan: [3 2 1]

That means 3 units of the cure on the first day, 2 on the second, and 1 on the third. Ok. Given the same patient schedule [1 2 3 4 5], what's our medicine usage each day?

Uh... shoot. It's not a quick multiplication:

- On Monday, 1 patient comes in. It's her first day, so she gets 3 units.
- On Tuesday, the Monday gal gets 2 units (her second day), but two new patients arrive, who get 3 each (2 * 3 = 6). The total is 2 + (2 * 3) = 8 units.
- On Wednesday, it's trickier: The Monday gal finishes (1 unit, her last day), the Tuesday people get 2 units (2 * 2), and there are 3 new Wednesday people... argh.

The patients are overlapping and it's hard to track. How can we organize this calculation?

Introduction to Convolutional Neural Networks Contd…

An idea: imagine *flipping* the patient list, so the first patient is on the right:

```
      Start of line
5 4 3 2 1
```

Next, imagine we have 3 separate rooms where we apply the proper dose:

```
Rooms    3 2 1
```

On your first day, you walk into the first room and get 3 units of medicine. The next day, you walk into room #2 and get 2 units. On the last day, you walk into room #3 and get 1 unit. There's no rooms afterwards, and your treatment is done.

To calculate the total medicine usage, line up the patients and walk them through the rooms:

```
Monday
----------------------------
Rooms              3 2 1
Patients       5 4 3 2 1
Usage              3
```

Introduction to Convolutional Neural Networks Contd…

On Monday (our first day), we have a single patient in the first room. She gets 3 units, for a total usage of 3. Makes sense, right?
On Tuesday, everyone takes a step forward:

```
Tuesday
----------------------------
Rooms              3 2 1
Patients ->    5 4 3 2 1
Usage             6 2     = 8
```

The first patient is now in the second room, and there's 2 new patients in the first room. We multiply each room's dose by the patient count, then combine.
The total day-by-day usage looks like this (don't forget Sat and Sun, since some patients began on Friday):

```
Plan      *  Patient List    = Total Daily Usage
[3 2 1]   *  [1 2 3 4 5]     = [3 8 14 20 26 14 5]

                M T W T F      M T W  T  F  S  S
```

Introduction to Convolutional Neural Networks Contd…

Convolution reverses one of the lists! Here's why.

Let's call our treatment plan $f(x)$. In our example, we used [3 2 1].

The list of patients (inputs) is $g(x)$. However, we need to *reverse* this list as we slide it, so the earliest patient (Monday) enters the hospital first (first in, first out). This means we need to use $g(-x)$, the horizontal reflection of $g(x)$. [1 2 3 4 5] becomes [5 4 3 2 1].

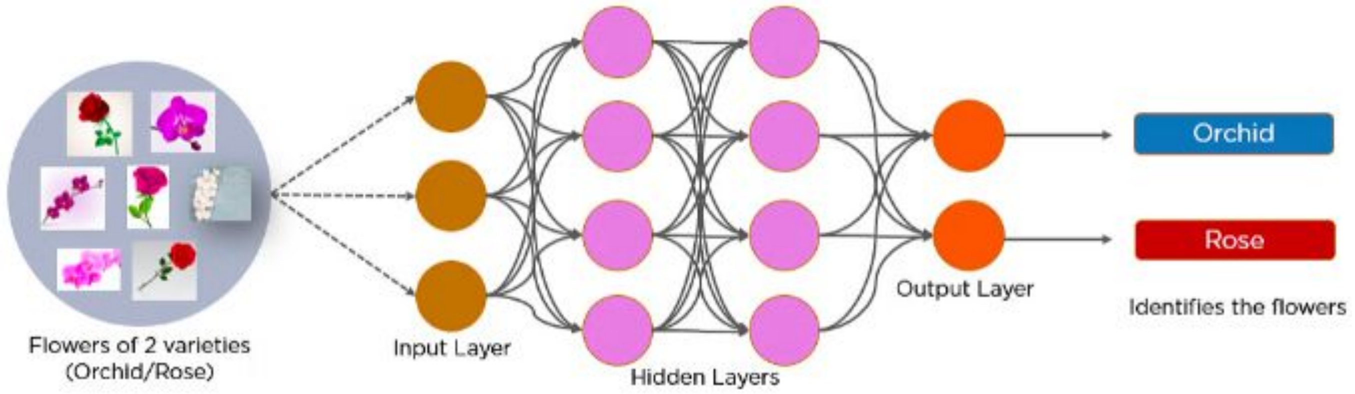
Now that we have the reversed list, pick a day to compute (t=0,1,2...). To slide our patient list by this much, we use: $g(-x+t)$
. That is, we reverse the list ($-x$) and jump to the correct day ($+t$).
We have our scenario:

- $f(x)$
- is the plan to use
- $g(-x+t)$
- is the list of inputs (flipped and slid to the right day).

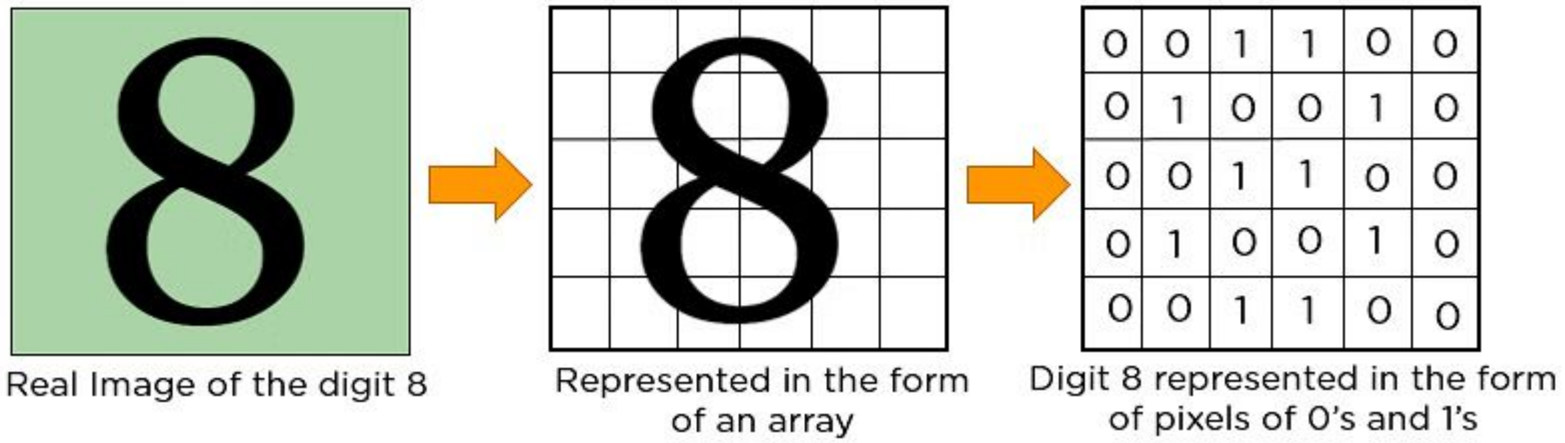
To get the total usage on day t
, we multiply each patient with the plan, and sum the results (an integral). To account for any possible length, we go from -infinity to +infinity.

What is Convolutional Neural Networks ?

- 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
- It is a feed-forward neural network that is generally used to analyze visual images by processing data with grid-like topology.

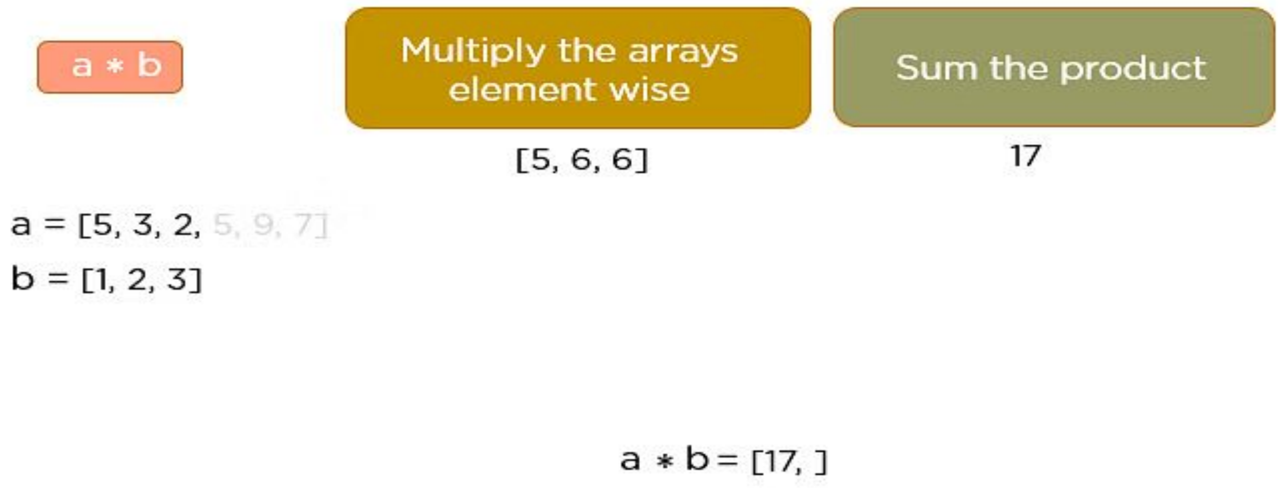


In CNN, every image is represented as an array of images

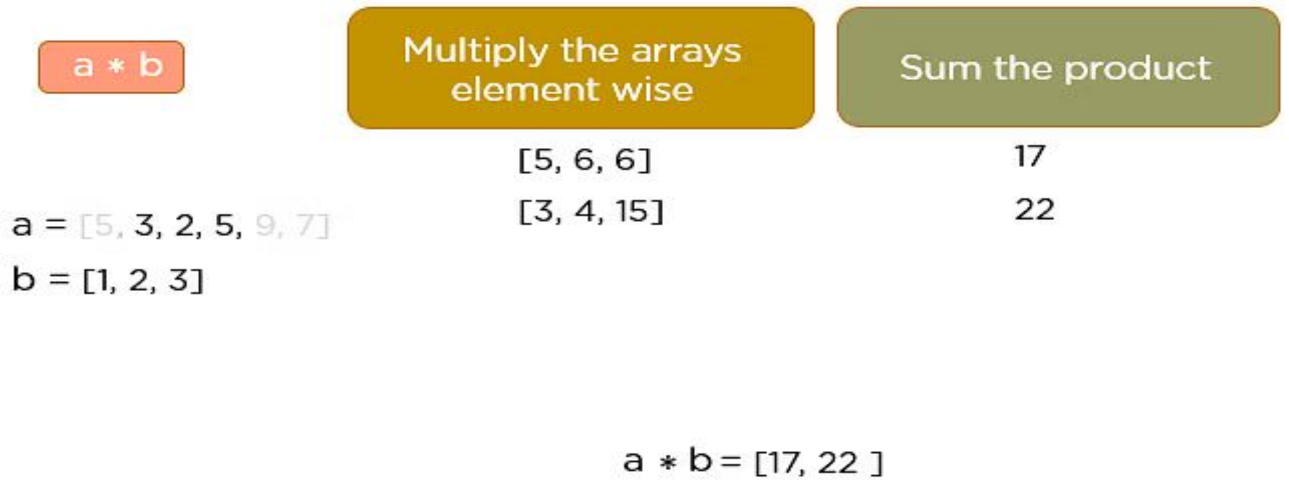

Real Image of the digit 8
Represented in the form of an array
Digit 8 represented in the form of pixels of 0's and 1's

0	0	1	1	0	0
0	1	0	0	1	0
0	0	1	1	0	0
0	1	0	0	1	0
0	0	1	1	0	0

What is Convolutional Neural Networks ? contd…



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



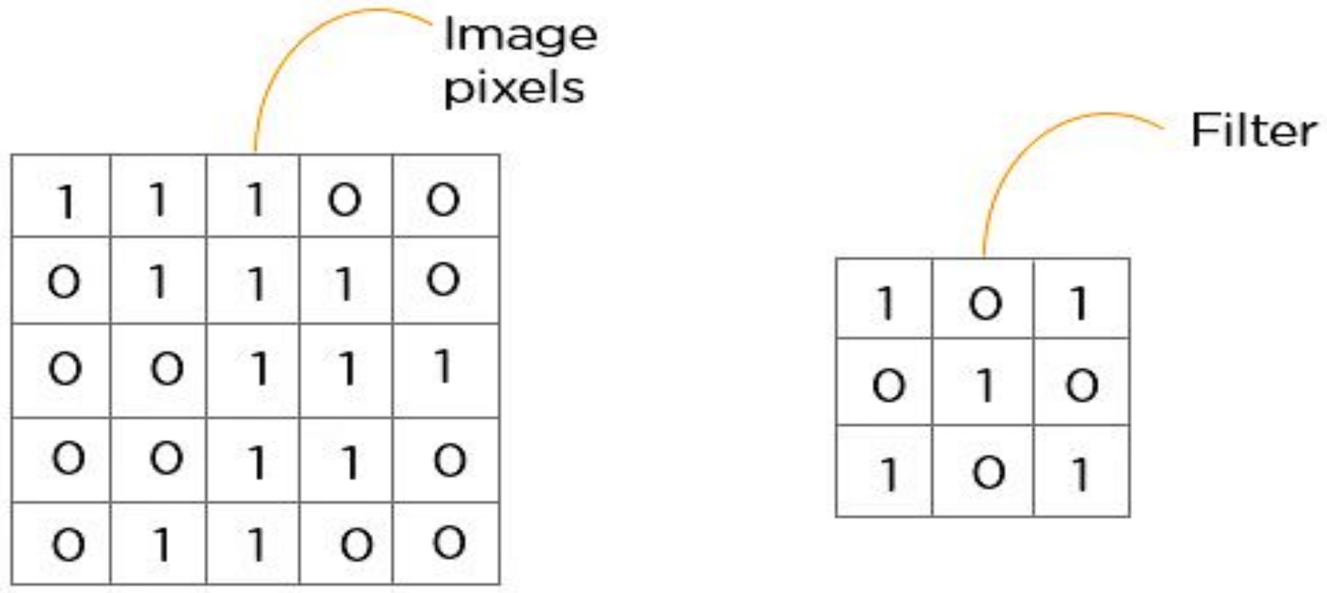
Layers in a Convolutional Neural Network

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

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

1. Convolution Layer

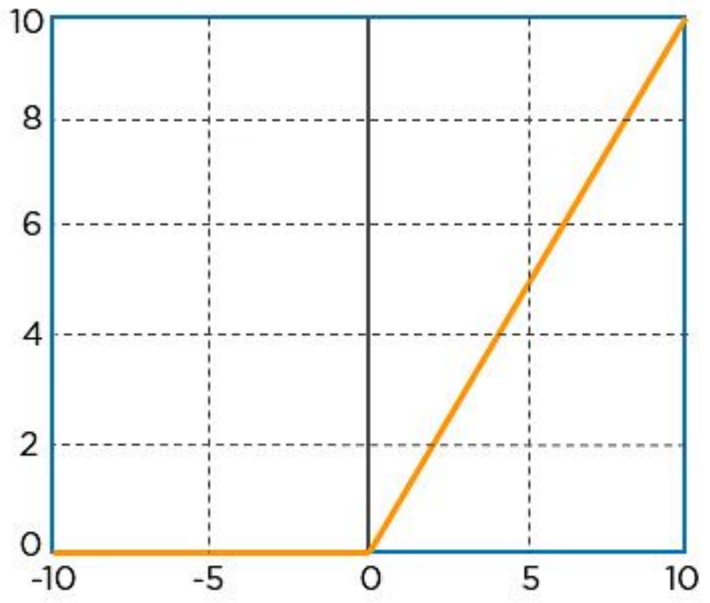
- first step in the process of extracting valuable features from an image.
- A convolution layer has several filters that perform the convolution operation.
- Every image is considered as a matrix of pixel values.



$$\begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 \end{bmatrix} \qquad \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix}$$

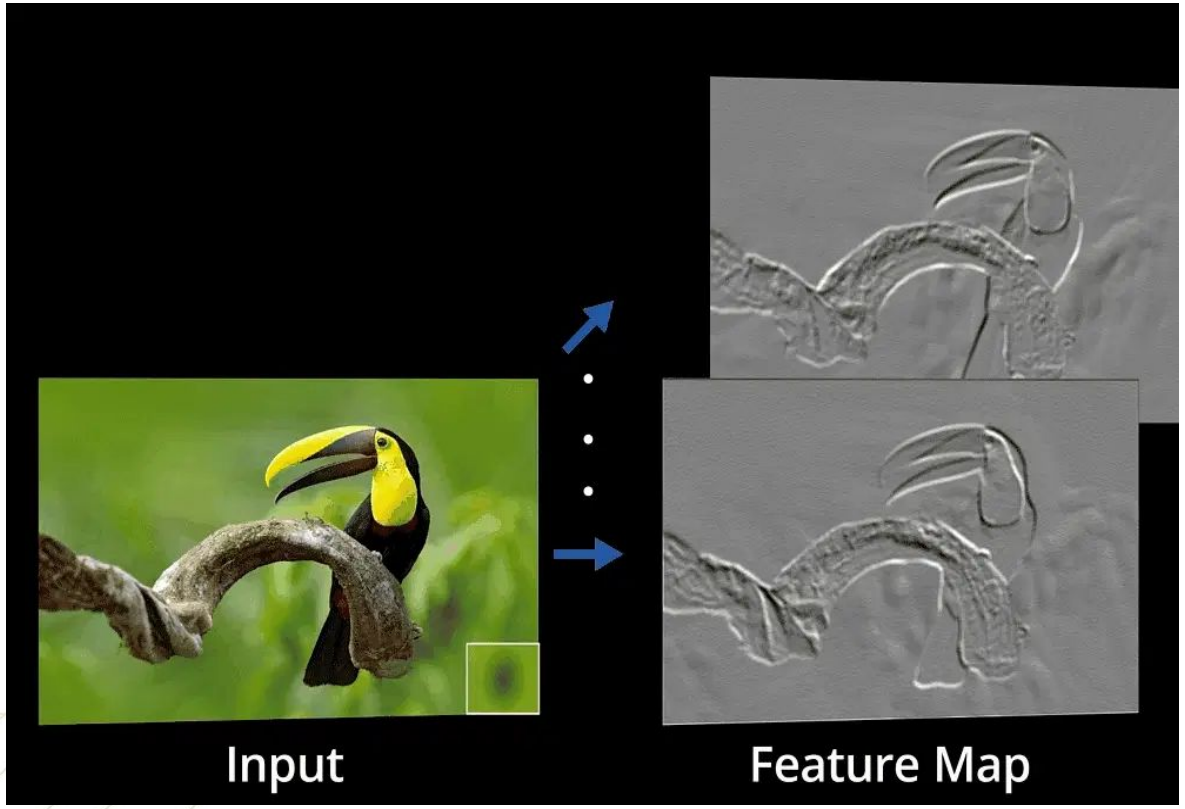
2. ReLU / Activation Layer

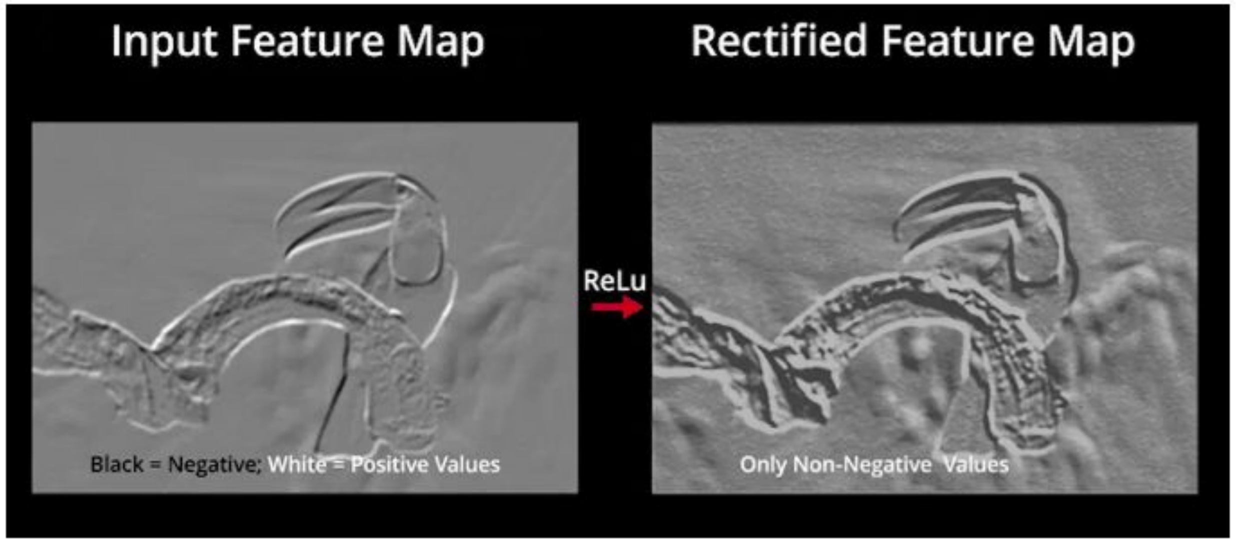
- ReLU stands for the rectified linear unit.
- ReLU performs an element-wise operation and sets all the negative pixels to 0.
- It introduces non-linearity to the network, and the generated output is a rectified feature map.
- Below is the graph of a ReLU function:



$R(z) = \max(0, z)$

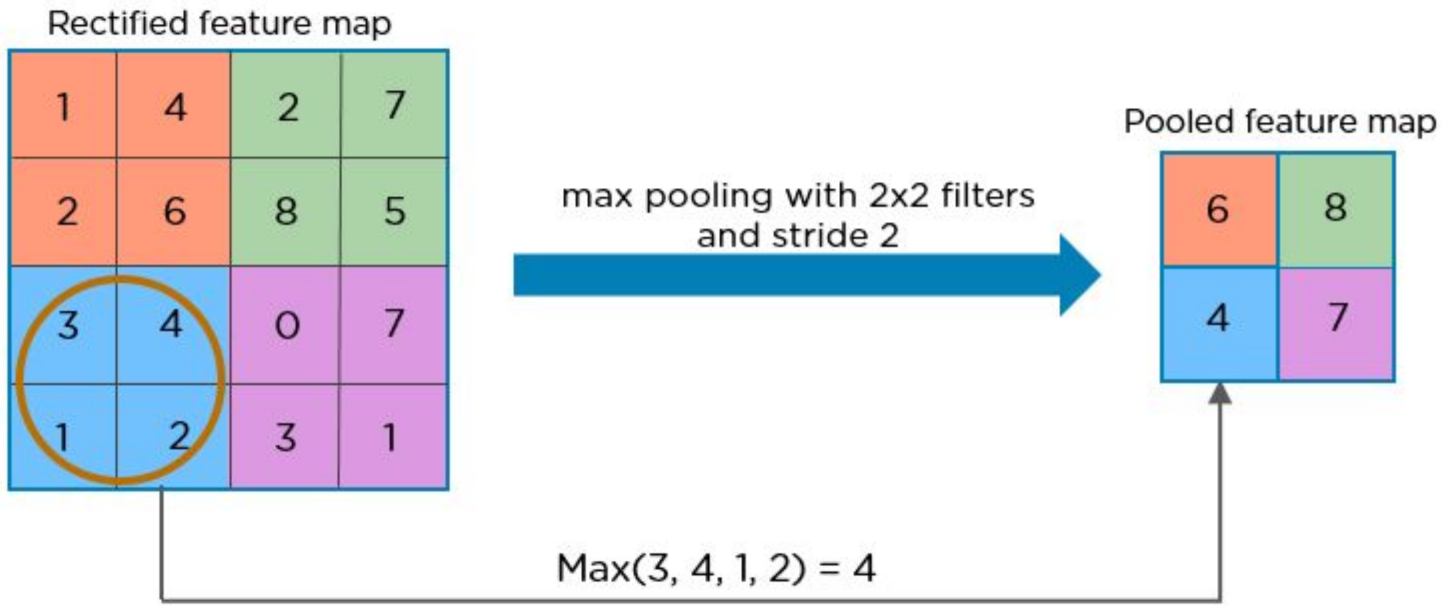
2. ReLU / Activation Layer Contd..





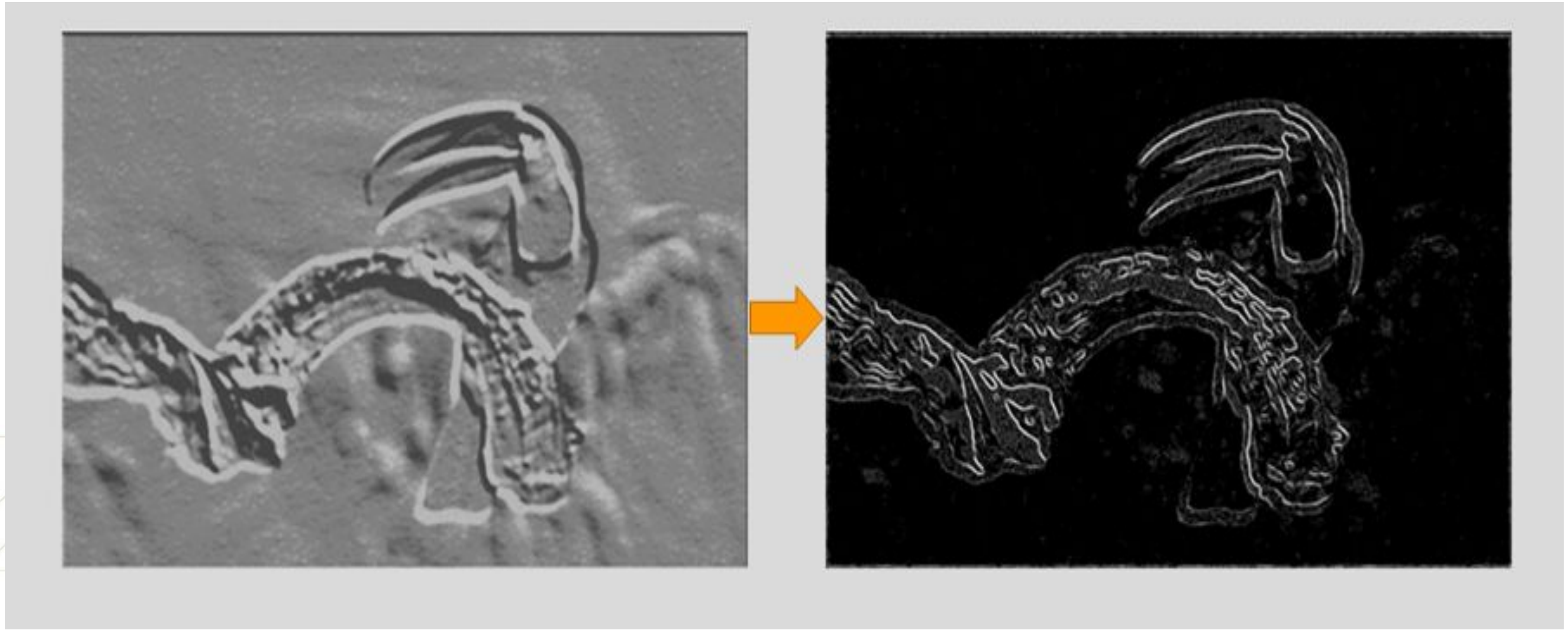
3. Pooling Layer

- 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.

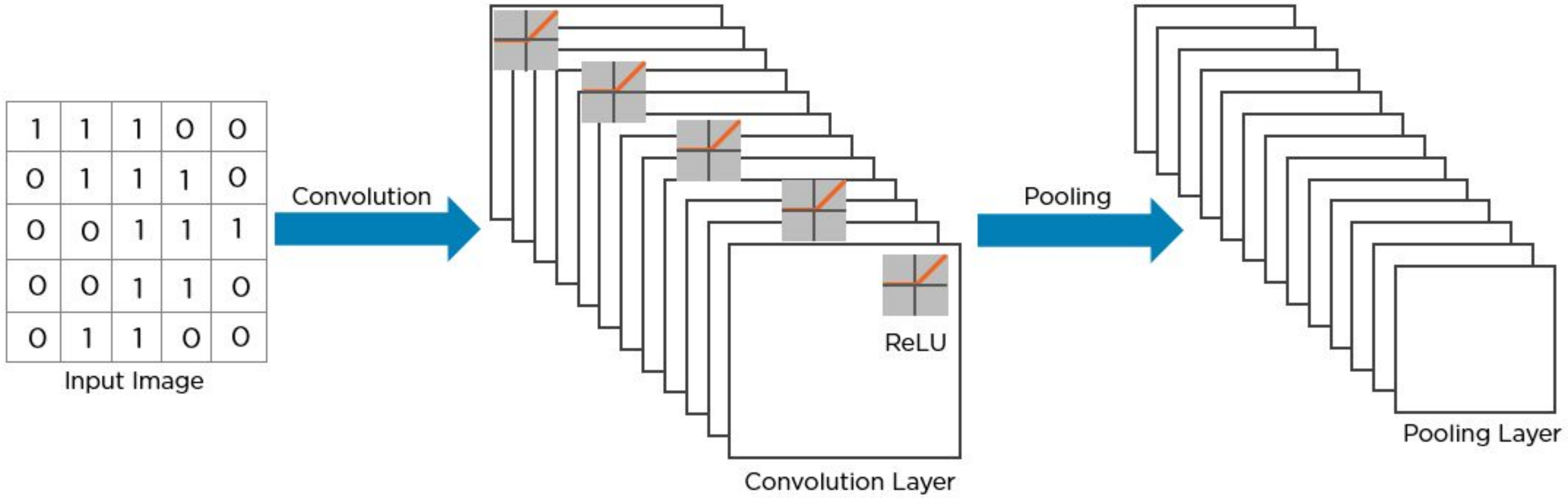


3. Pooling Layer Contd…

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

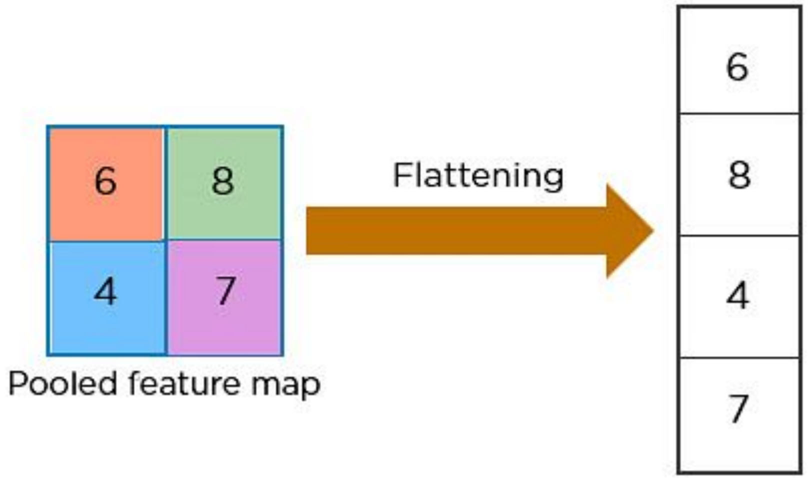


CNN so far…


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
Input Image
Convolution
ReLU
Convolution Layer
Pooling
Pooling Layer

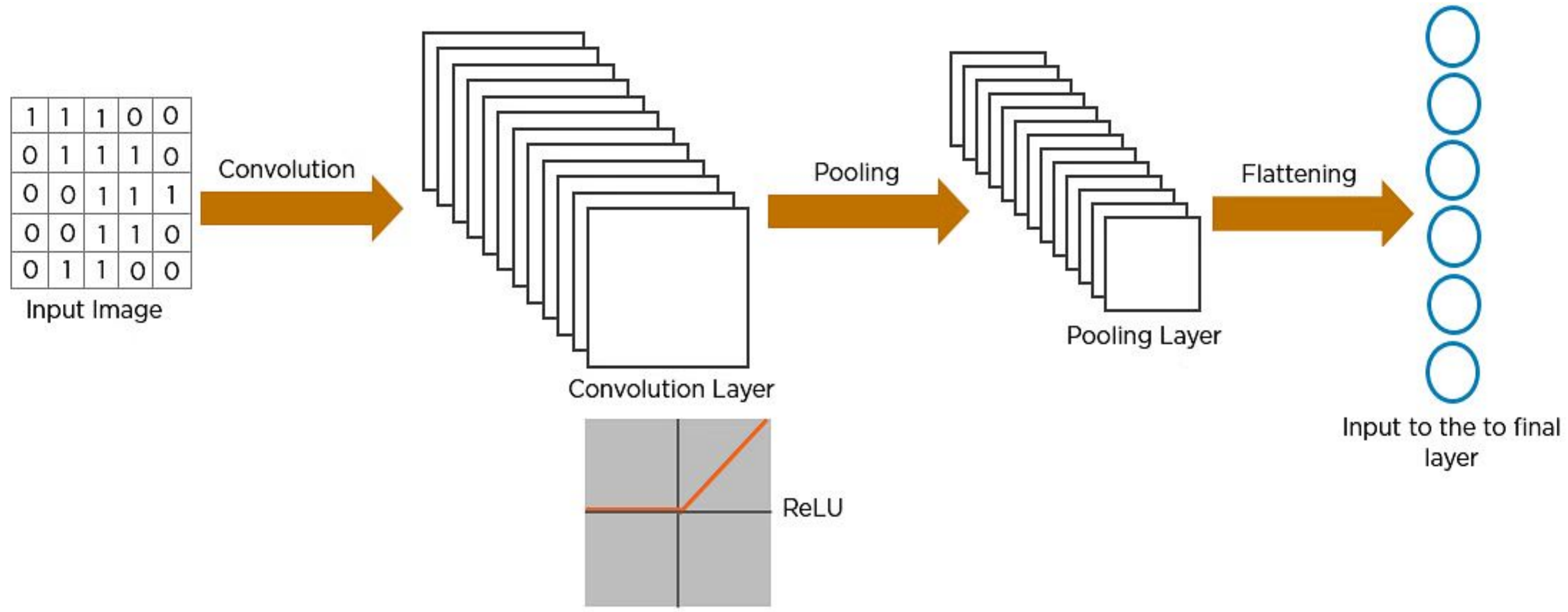
4. Flattening Layer

- Flattening is used to convert all the resultant 2-Dimensional arrays from pooled feature maps into a single long continuous linear vector.

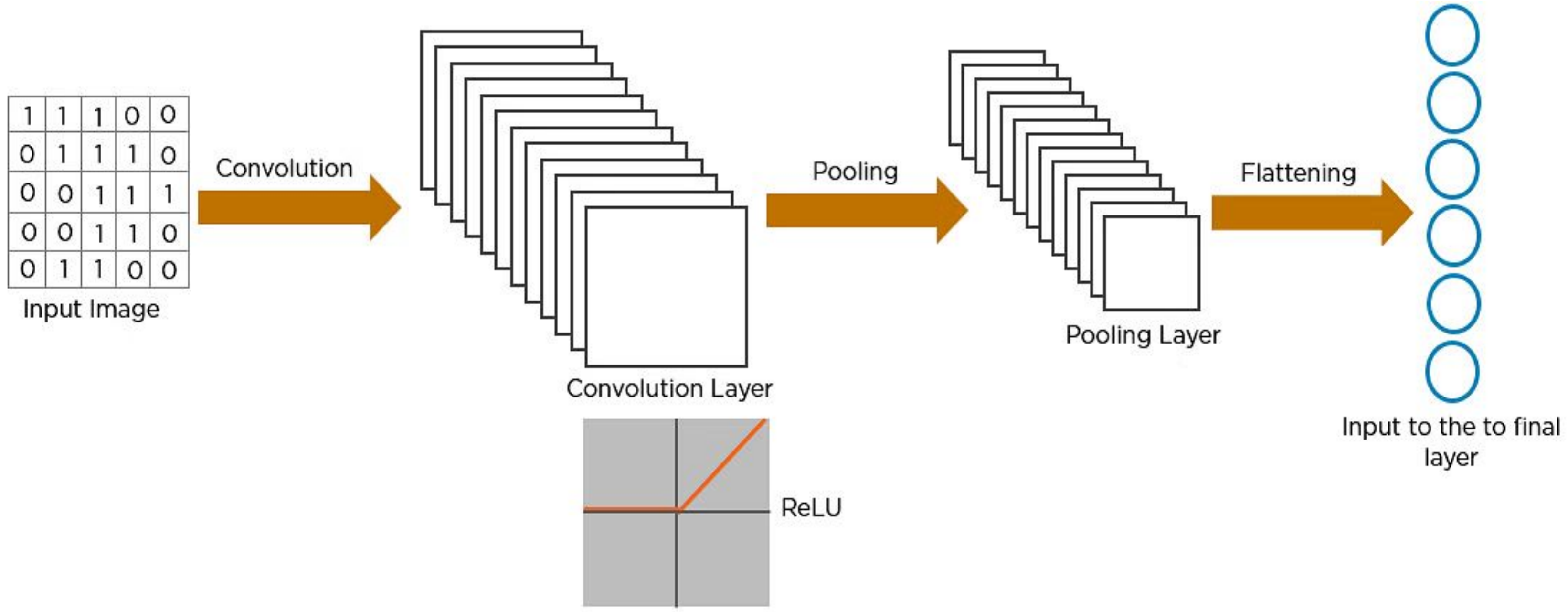


The flattened matrix is fed as input to the fully connected layer to classify the image.

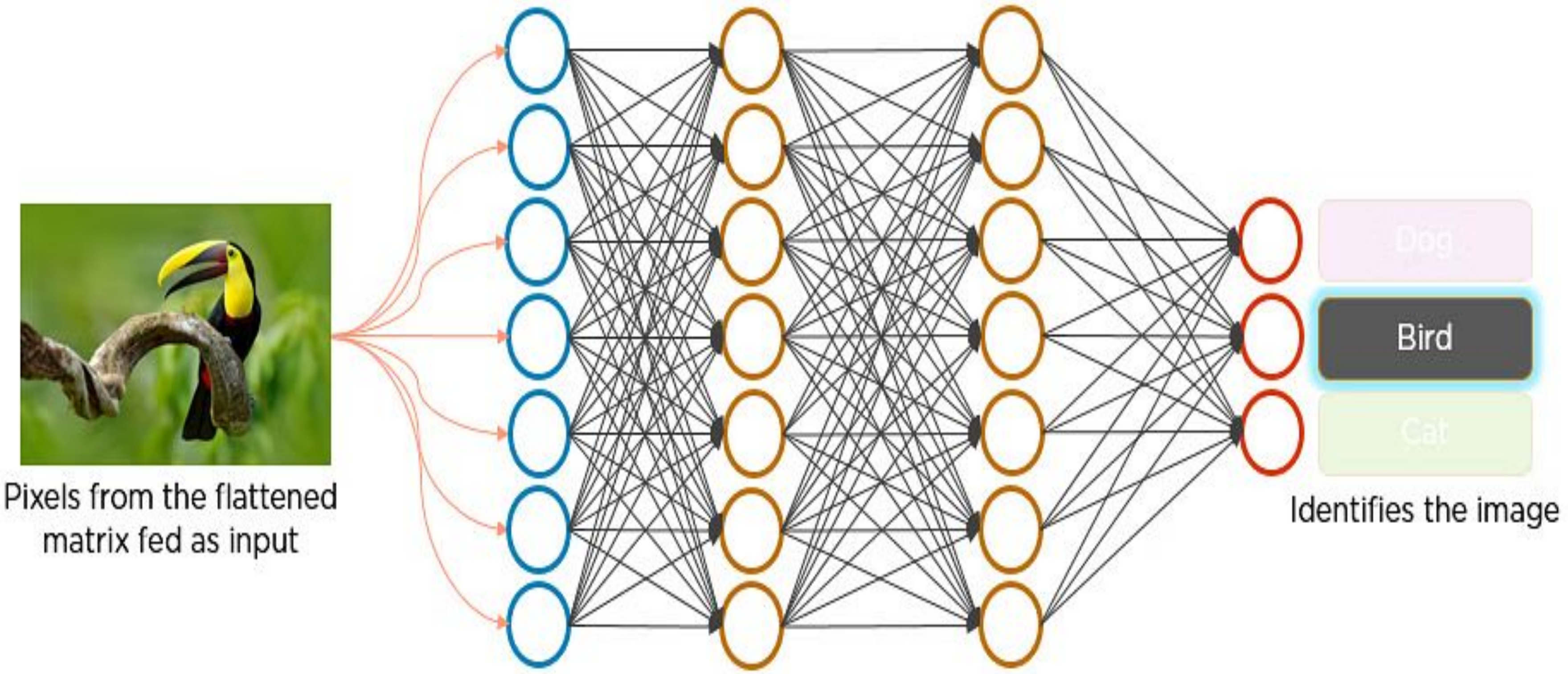
4. Flattening Layer Contd…



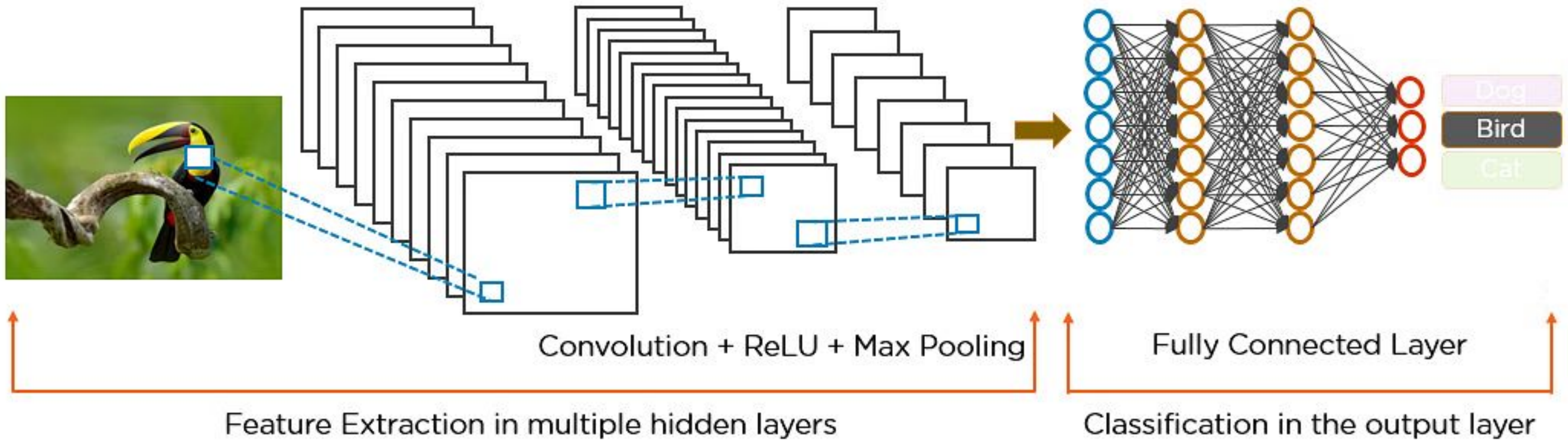
5. Fully Connected Layer



6. Output Layer



Complete Working of CNN


Dog
Bird
Cat
Convolution + ReLU + Max Pooling
Fully Connected Layer
Feature Extraction in multiple hidden layers
Classification in the output layer

VIPS Technical Campus
योगः कर्मसु कौशलम्
IN PURSUIT OF PERFECTION

Practical Implementation of CNN

CNN Evaluation

- **Accuracy :** It 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.

  Accuracy = (TP+TN)/(TP + FP + TN + FN)

- **Precision :** It 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.

  Precision = TP/(TP + FP)

CNN Evaluation Contd…

- **Recall :** It 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.

  Recall = (TP)/(TP+FN)

- **F1 Score :** It 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.

  F1 Score = (2PR)/(P+R)

CNN Evaluation Contd…

SCHOOL OF ENGINEERING AND TECHNOLOGY

True positive rate:

Also called or termed sensitivity. True Positive Rate is considered as a portion of positive data points that are correctly considered as positive, with respect to all data points that are positive.

TPR=TP/TP+FN

True Negative Rate

Also called or termed specificity. False Negative Rate is considered as a portion of negative data points that are correctly considered as negative, with respect to all data points that are negatives.

TNR=TN/TN+FP

False-positive Rate

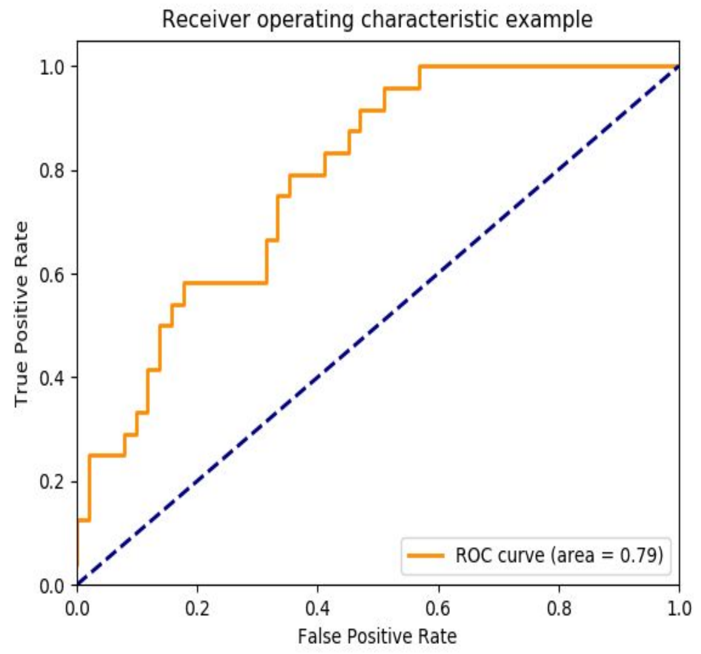
False Negatives rate is actually the proportion of actual positives that are incorrectly identified as negatives

FPR=FP/FP+TN

False Positive Rate and True Positive Rate both have values in the range [0, 1].

CNN Evaluation Contd…

- **ROC-AUC :** It is one of the widely used metrics and basically used for binary classification. The AUC of a classifier is defined as the probability of a classifier will rank a randomly chosen positive example higher than a negative example. Now the thing is what is AUC then? So, AUC is a curve plotted between False Positive Rate Vs True Positive Rate at all different data points with a range of [0, 1]. Greater the value of AUC, better the performance of the model.



Types of CNNs

- **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 2 convolutional and pooling layers followed by subsampling, and 3 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 5 convolutional layers and 3 fully connected layers, with innovations like ReLU activation and dropout. AlexNet demonstrated the power of deep learning, 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.
- **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 CNNs

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.

Transfer Learning

- **In transfer learning,** the knowledge of an already trained machine learning model is applied to a different but related problem. For example, if you trained a simple classifier to predict whether an image contains a backpack, you could use the knowledge that the model gained during its training to recognize other objects like sunglasses.

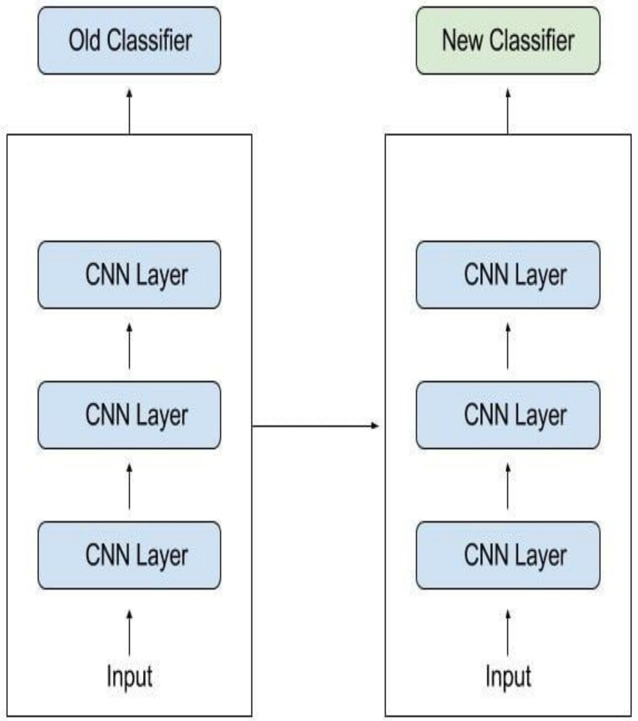
- With transfer learning, we basically try to exploit what has been learned in one task to improve generalization in another. We transfer the weights that a network has learned at "task A" to a new "task B."

- The general idea is to use the knowledge a model has learned from a task with a lot of available labeled training data in a new task that doesn't have much data. Instead of starting the learning process from scratch, we start with patterns learned from solving a related task.

How Transfer Learning Works

In computer vision, for example, neural networks usually try to detect edges in the earlier layers, shapes in the middle layer and some task-specific features in the later layers. In transfer learning, are used and we only retrain the latter layers. It helps leverage the labeled data of the task it was initially trained on. This process of retraining models is known as fine-tuning. In the case of transfer learning, though, we need to isolate specific layers for retraining. There are then two types of layers to keep in mind when applying transfer learning:

- **Frozen layers:** Layers that are left alone during retraining and keep their knowledge from a previous task for the model to build on.
- **Modifiable layers:** Layers that are retrained during fine-tuning, so a model can adjust its knowledge to a new, related task.



Why Transfer Learning ?

- **In transfer learning,** the knowledge of an already trained machine learning model is applied to a different but related problem. For example, if you trained a simple classifier to predict whether an image contains a backpack, you could use the knowledge that the model gained during its training to recognize other objects like sunglasses.

- With transfer learning, we basically try to exploit what has been learned in one task to improve generalization in another. We transfer the weights that a network has learned at "task A" to a new "task B."

- The general idea is to use the knowledge a model has learned from a task with a lot of available labeled training data in a new task that doesn't have much data. Instead of starting the learning process from scratch, we start with patterns learned from solving a related task.

When to use Transfer Learning ?

- **In transfer learning,** the knowledge of an already trained machine learning model is applied to a different but related problem. For example, if you trained a simple classifier to predict whether an image contains a backpack, you could use the knowledge that the model gained during its training to recognize other objects like sunglasses.

- With transfer learning, we basically try to exploit what has been learned in one task to improve generalization in another. We transfer the weights that a network has learned at "task A" to a new "task B."

- The general idea is to use the knowledge a model has learned from a task with a lot of available labeled training data in a new task that doesn't have much data. Instead of starting the learning process from scratch, we start with patterns learned from solving a related task.

Thank You