Anomaly Detection from CCTV Feed

Under the guidance of
Dr. Sujay Deb

Aakash - 2021002
Lakshay Chauhan - 2021060
Shubham Sharma -2021099

INDRAPRASTHA INSTITUTE *of*
INFORMATION TECHNOLOGY
DELHI

Aim *(what?)*

The goal is to develop a simple and cost effective "*Smart CCTV camera*" capable of detecting anomalies in various environments.

For example, in a laboratory, it can identify anomalies such as **explosions** or **vandalism**, while near a road, it can identify anomalies such as **accidents** or **shootings**.

Motivation *(why?)*

While using traditional CCTV cameras, a major drawback is the need to constantly monitor footage 24/7, even when nothing noteworthy occurs. Automation is essential, thus leading to the development of "*Smart CCTV cameras*".

For instance, when placed near a road, these cameras can automatically **detect** accidents and **alert the authorities**.

Neural Networks

Neural Networks can help us solve this problem, thus having a basic understanding of them is essential.

Neural Network

Neural network is a subset of machine learning and a part of deep learning. Inspired by human brain; there structure comprised of layers.

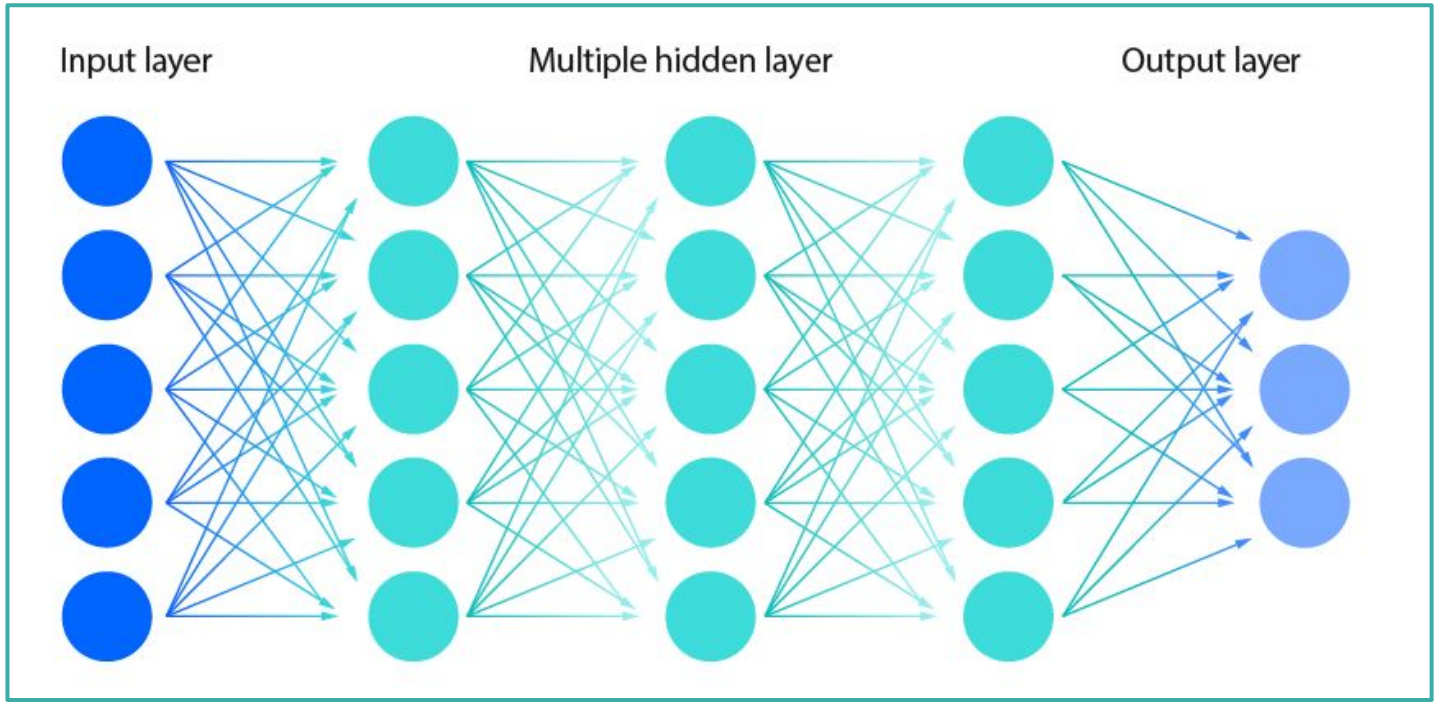
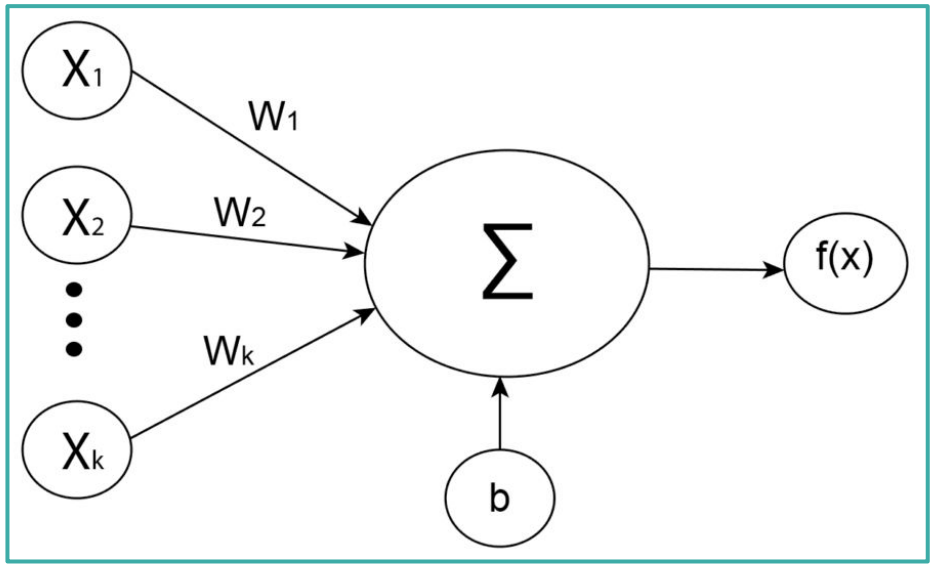


Image source: *What are Neural Networks? | IBM*

The **input** layer has nodes representing features of the input or pixels.
The **output** layer nodes indicate the likelihood of the input belonging to a certain class.
The **hidden** layer doesn't represent anything; it acts as a link between the input and output layers.

Neural Network

It comprises of neurons & weights, structured in layers.



Source: *Convolutional Neural Network vs. Regular Neural Network*

Neuron is a basic computational unit that receives one or more inputs and generates an output.

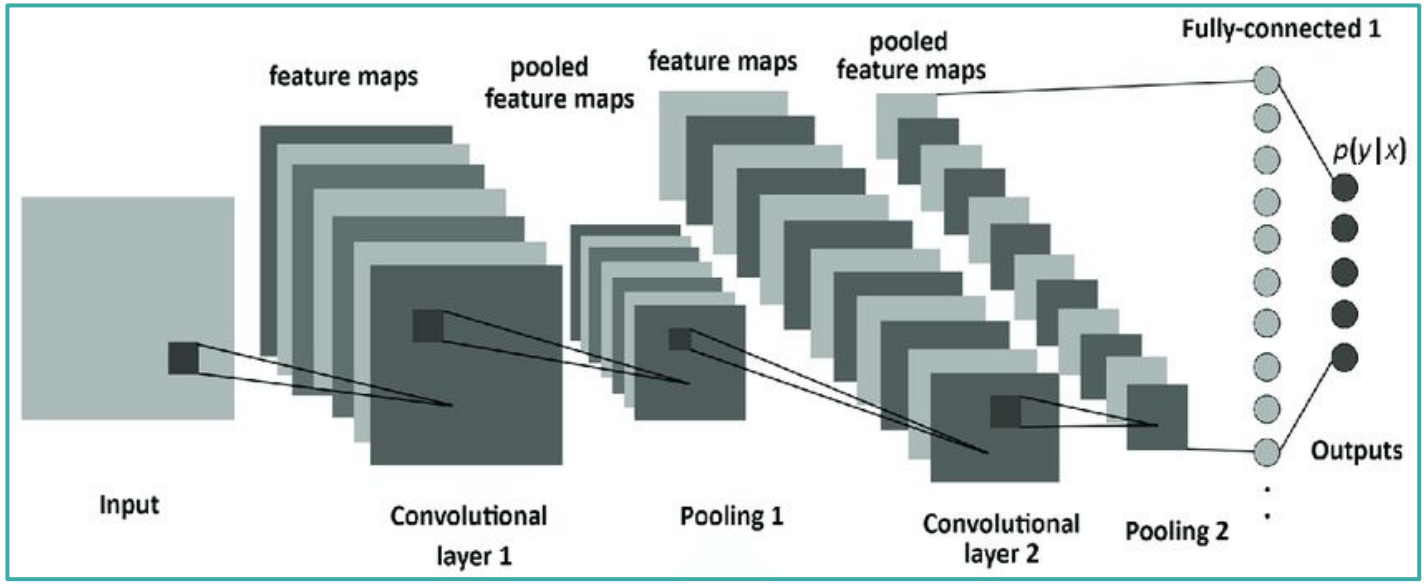
Weights connect successive layers.

Activation Function is used to introduce non linearity, allowing it to learn more complex pattern.

In order to get the output of the neuron, we need to calculate the weighted sum of all the inputs and weights of the connections. After that, we add bias to the sum and apply the activation function.

Convolutional Neural Network

Convolutional neural networks are distinguished from other neural networks by their superior performance with image, speech, or audio signal inputs.



Source: *The structure of a CNN, consisting of convolutional, pooling, and fully-connected layers.*

These have three main types of layers

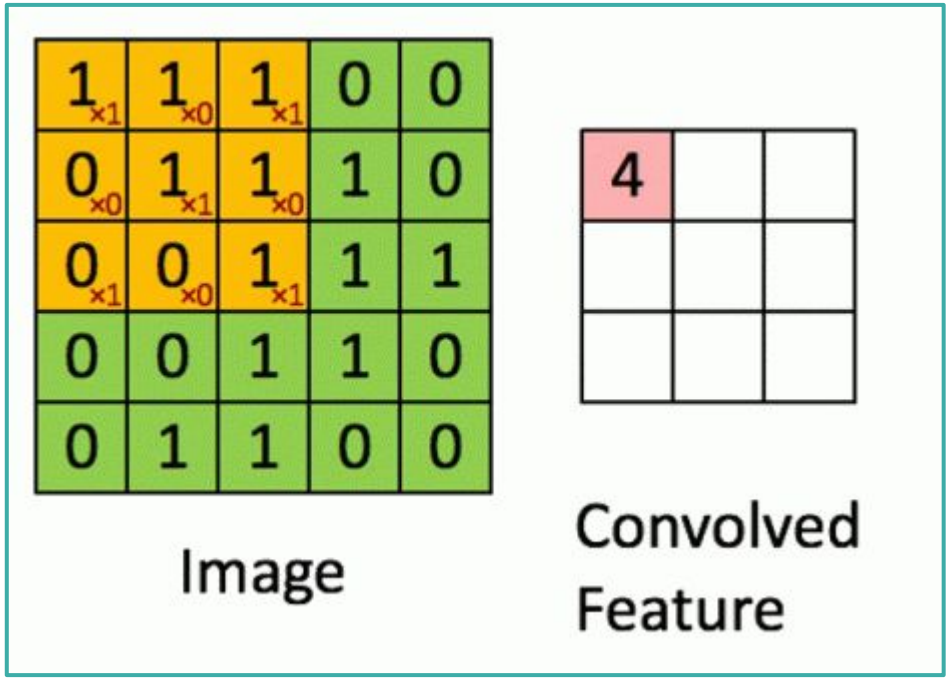
- Convolutional layer
- Pooling layer
- Fully Connected layer

Convolutional Layer

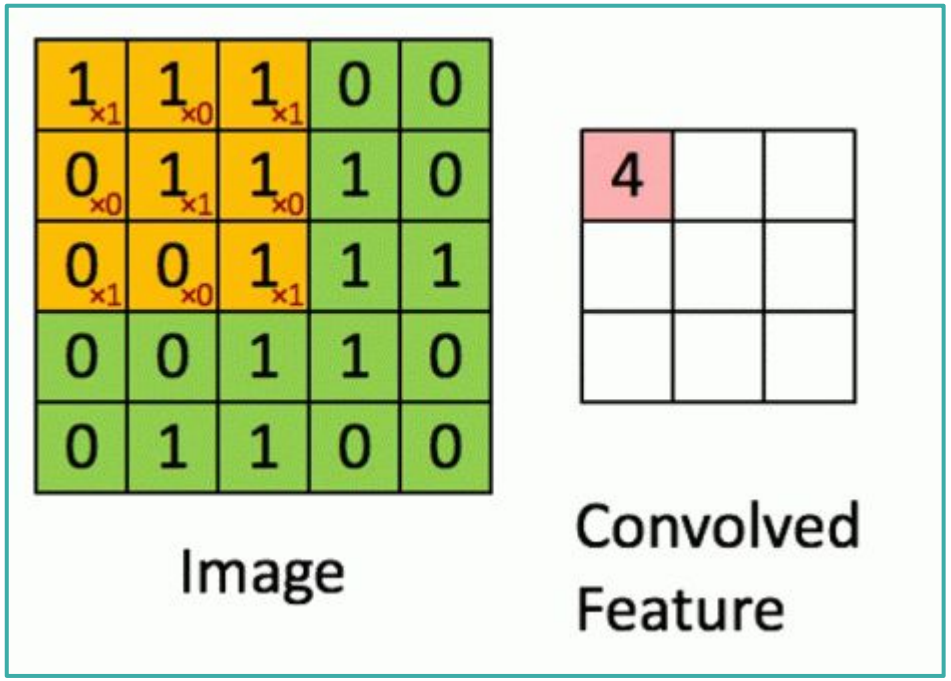
The input will have three dimension which correspond to **RGB** in an image. We also have a feature filter, also known as a **kernel**, which will move across the receptive fields of the image, checking if the feature is present

The kernel is a two-dimensional (2-D) array of weights, which represents part of the image

The final output from the series of dot products from the input and the filter is known as a **feature map**, **activation map**, or a **convolved feature**.



Some parameters, like the weight values, adjust during training through the process of backpropagation



1	0	1
0	1	0
1	0	1

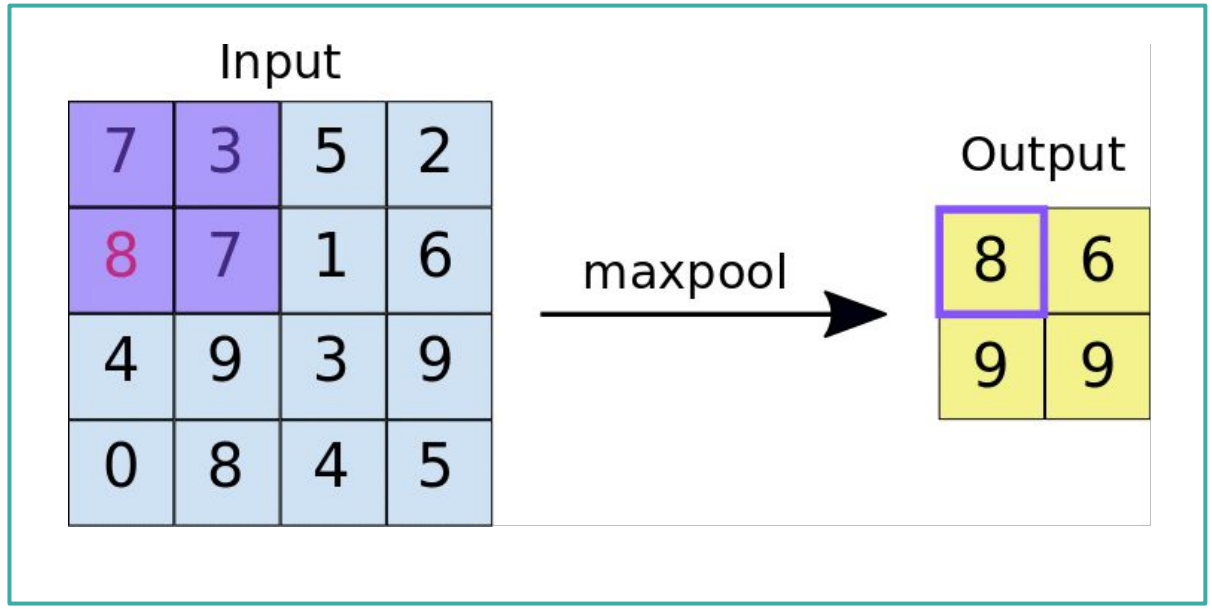
Kernel

- *Shape of image = (m, n)*
- *Shape of kernel = (a, b)*
- *Shape of the new convolved feature = (m+1-a, n+1-b)*

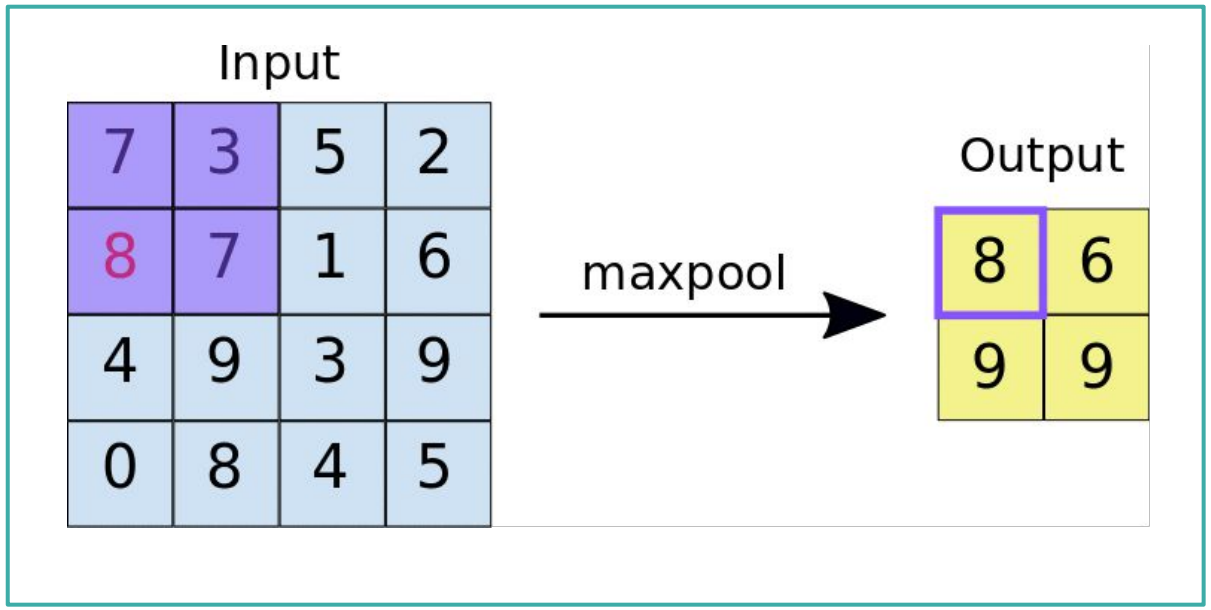
This kernel goes through the image input, multiplies its values with corresponding values in the kernel, and then calculates the sum of those products.

Pooling layer

Pooling layers, also known as downsampling, conducts dimensionality reduction, reducing the number of parameters in the input.



Similarly, there is average pooling where average of all is taken



Max polling takes the maximum value inside the window.

6.25	3.5
5.25	5.25

Similarly taking average value inside the window results in average pooling. Here's the result of average pooling in the provided example

Fully Connected layer

This layer performs the task of classification based on the features extracted through the previous layers *(convolutional and pooling)* and their different filters.

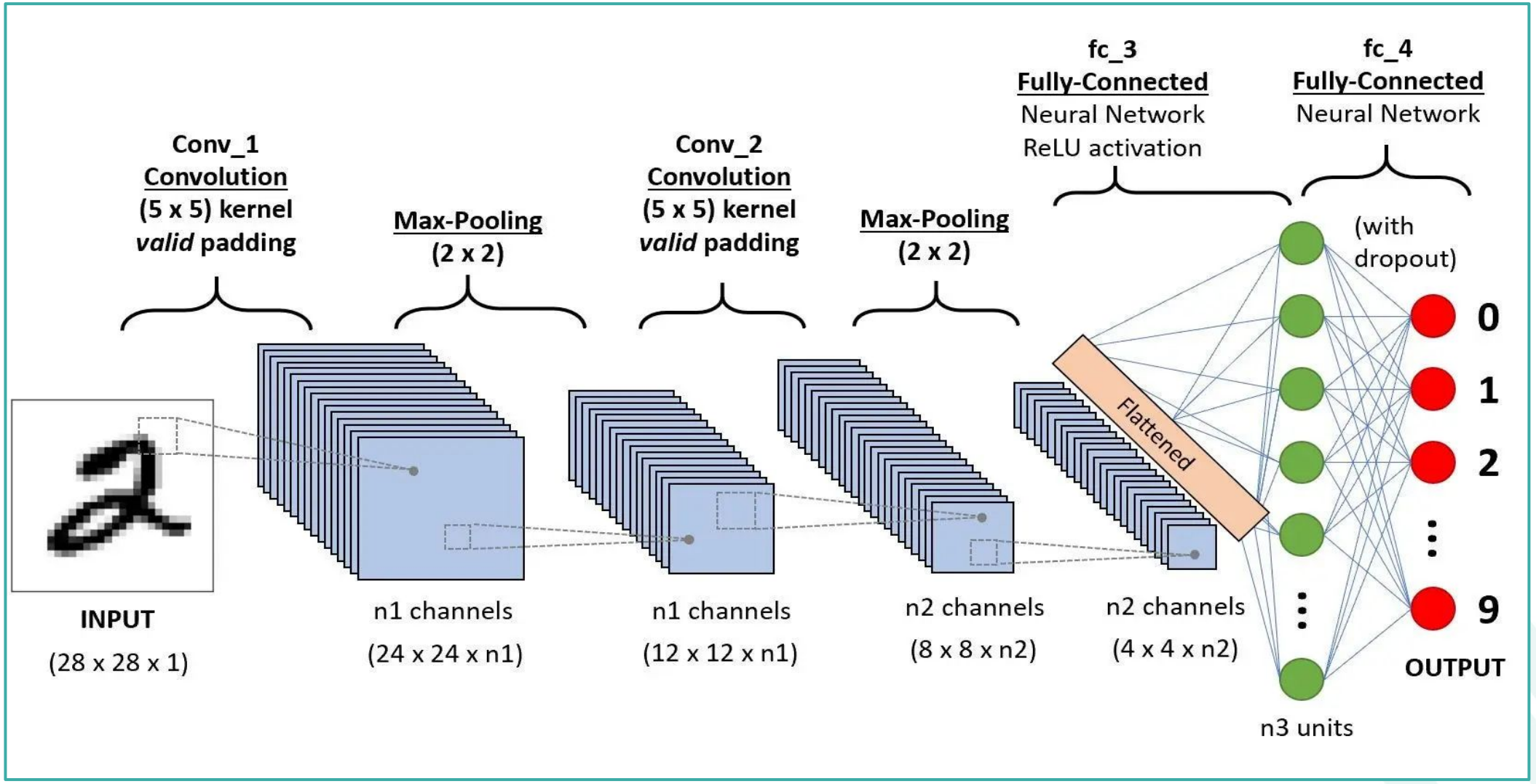
While convolutional and pooling layers tend to use ReLu *(Rectified Linear unit)* functions, fully connected layers *(mainly output layer)* usually leverage a softmax activation function to classify inputs appropriately, producing a probability from 0 to 1.

$$R(z) = \begin{cases} z & z > 0 \\ 0 & z <= 0 \end{cases}$$

$$\text{softmax}(z_j) = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \text{ for j} = 1,\ldots,K$$

Source: Activation Functions: Sigmoid, Tanh, ReLU, Leaky ReLU, Softmax | by Mukesh Chaudhary | Medium

Image Classification using CNN



Source: *A Guide to Convolutional Neural Networks — the ELI5 way | Saturn Cloud Blog*

Overview *(how?)*

1. Capture video stream from the CCTV camera.
2. Break down the video into brief segments *(~ 10 seconds each)*.
3. Use an Anomaly Detection model on these segments.
4. Report any detected anomalies.
    a. Save footage from 1 minute before the anomaly is reported until 1 minute after the anomaly stops being reported.
    b. Immediately notify authorities if the confidence level *(model's certainty about the anomaly)* is above a set threshold.

Dataset *(which?)*

Chen, Chen. “UCF Crime Dataset”. University of Central Florida. *https://www.crcv.ucf.edu/chenchen/dataset.html*

This Dataset contains 13 classes of anomaly:
Abuse, Arrest, Arson, Assault, Burglary, Explosion, Fighting, Road Accidents, Robbery, Shooting, Shoplifting, Stealing, Vandalism

Other links for the same dataset;

- *https://paperswithcode.com/dataset/ucf-crime*
- *https://www.kaggle.com/datasets/mission-ai/crimeucfdataset/*
- *https://www.dropbox.com/sh/75v5ehq4cdg5g5g/AABvnJSwZI7zXb8_myBA0CLHa?dl=0*

Dataset *(why?)*

We chose the "*UCF Crime Dataset*" because it's the only available dataset with substantial CCTV footage *(~ 128 hours of videos)*.

Additionally, we opted for a single dataset to maintain consistency, as using different datasets could introduce various video types, potentially confusing the model.

Despite our search, we couldn't find a dataset specific to an Indian setting.

Implementation

The journey we took to reach the current model

Our initial thought

We can perform anomaly detection if we break down a segment into a sequence of frames and identify objects within those frames.

For example, if we are detecting the anomaly "*Shooting*", we can look for "*guns*" in the frames. Similarly, if the anomaly is "*Explosion*", we can look for the presence of "*fire*" in the frames. We will classify that segment as the corresponding anomaly if we consistently identify a particular object in successive frames.

We can use object detection models for detecting particular object in a frame.

Object Detection *(OD)* models

- Inception v3
- Tensorflow lite & YOLO

** models that we used*

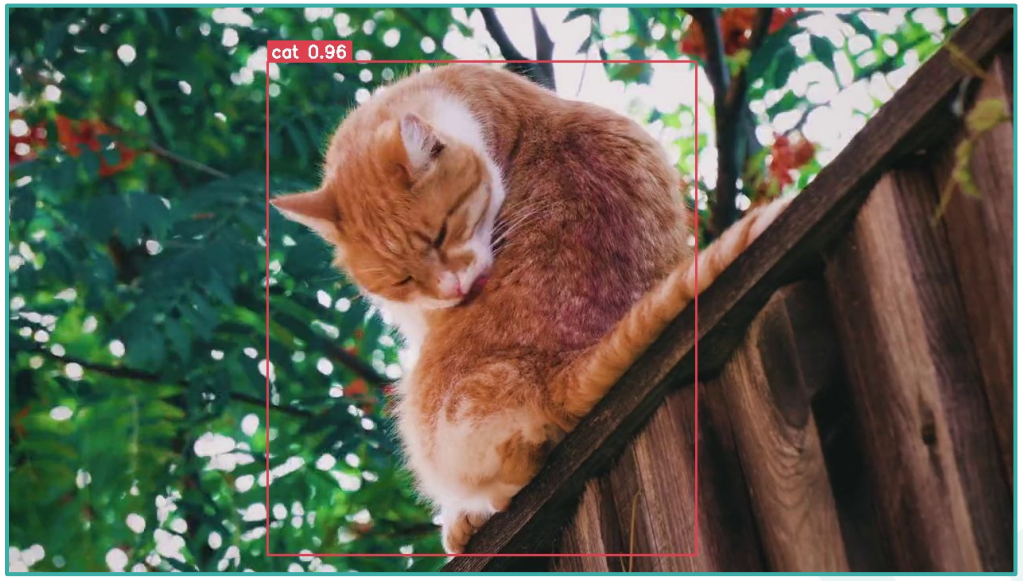
Inception v3 *(OD model)*

Inception v3 is a deep learning model developed by Google for image classification and object recognition tasks. Its primary function is to analyze and categorize input images into different classes.

Tensorflow lite & YOLO *(OD model)*

Tensorflow lite & YOLO are both deep learning object detection models that are lightweight and highly tuned for mobiles and edge devices.

Tensorflow lite:
Source: *TensorFlow-Lite-Object-Detection-on-Android-and-Raspberry-Pi*



YOLO:
Reference: *https://github.com/Hyouteki/btp/tree/main/yolo*

Why we transitioned from OD models?

We realized that object detection would not work in general.

It may work for anomalies such as "*Explosion*", where detecting an object in a frame is enough to catch the anomaly. Also training on custom dataset was not easy in case of YOLO.

But it will not work with anomalies which involve a series of actions such as "*Stealing*". We can't decide if a segment shows "*Stealing*" by looking at one frame. We have to analyze a series of frames in continuation to conclude.

Human Action *(activity)* Recognition model

This model analyzes and interprets human actions in videos. The primary goal is to automatically identify and classify different actions or activities performed by individuals. These actions include gestures, movements, or behaviors such as walking, running, sitting, waving, etc.

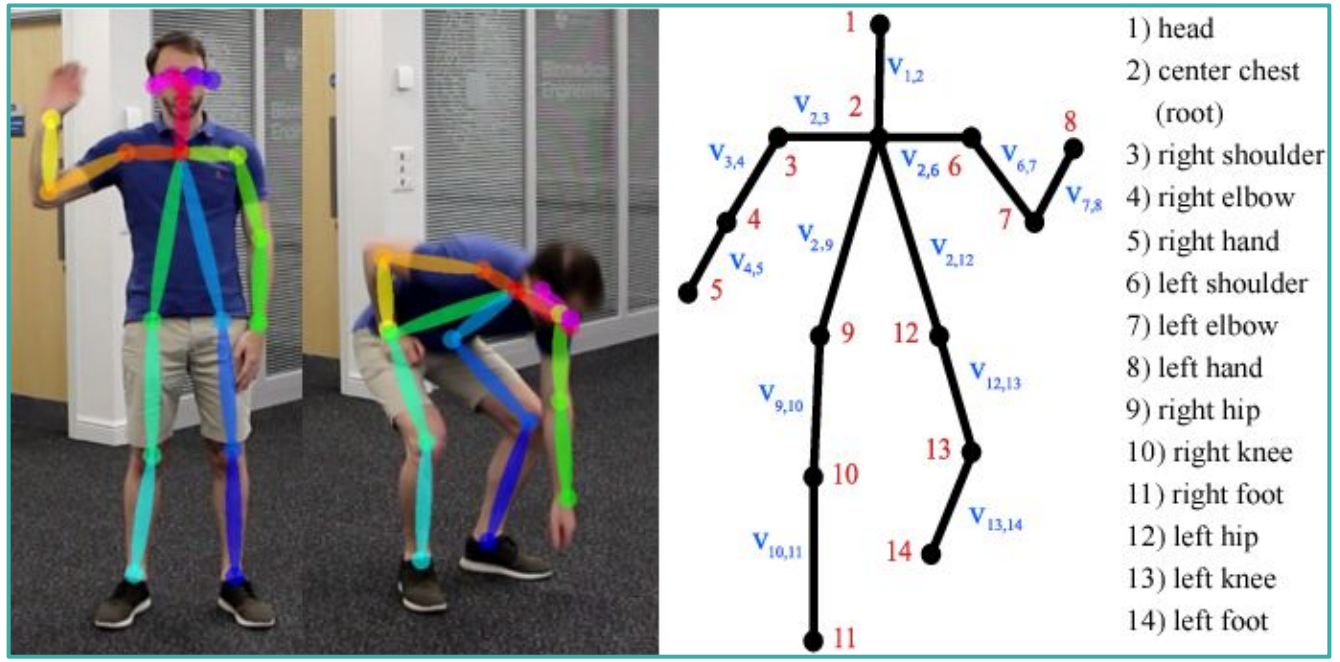


Image source: *[PDF] ActionXPose: A Novel 2D Multi-view Pose-based Algorithm for Real-time Human Action Recognition | Semantic Scholar*

Why did we transition from action recognition model?

This model may not be good at spotting non-human-related anomalies like “*Explosion*” or “*Car accidents*” because it's specifically designed to recognize human actions.

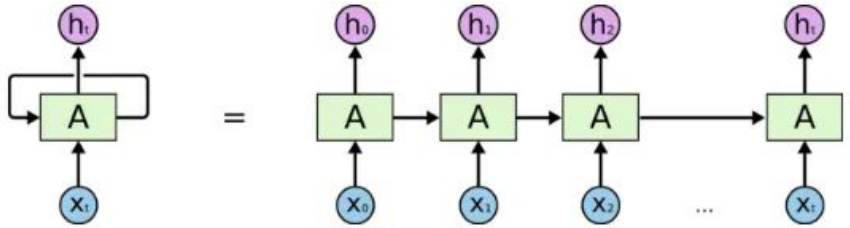
Also, it might make mistakes by giving false positives *(incorrectly saying something is there when it's not)* because this model primarily focuses on the actions rather than the visual information. For instance, it may not find the difference between yoga and dancing as many actions are similar but the context is different. *(Although this may not be that big of an issue in our case, but still good to know)*

LRCN *(Long-term Recurrent Convolutional Network)* model

This is the current model that we are using.

It comprises of Convolutional Neural Network *(CNN)* and LSTM *(Long Short Term Memory)*.

LSTM is a type of Recurrent Neural Network made for handling sequences of data. While a Convolutional Neural Network (CNN) can classify a single frame, LSTM helps us analyze multiple frames in a row to make a meaningful decision



An unrolled recurrent neural network.

Source: [Understanding RNN and LSTM. What is Neural Network? | by Aditi Mittal | Medium]

LRCN model Architecture and Parameters

Layer (type)	Output Shape
conv_lstm2d (ConvLSTM2D)	(None, 92, 90, 90, 4)
max_pooling3d (MaxPooling3D)	(None, 92, 45, 45, 4)
time_distributed (TimeDistributed)	(None, 92, 45, 45, 4)
conv_lstm2d_1 (ConvLSTM2D)	(None, 92, 43, 43, 8)
max_pooling3d_1 (MaxPooling3D)	(None, 92, 22, 22, 8)
time_distributed_1 (TimeDistributed)	(None, 92, 22, 22, 8)
conv_lstm2d_2 (ConvLSTM2D)	(None, 92, 20, 20, 14)
max_pooling3d_2 (MaxPooling3D)	(None, 92, 10, 10, 14)
time_distributed_2 (TimeDistributed)	(None, 92, 10, 10, 14)
conv_lstm2d_3 (ConvLSTM2D)	(None, 92, 8, 8, 16)
max_pooling3d_3 (MaxPooling3D)	(None, 92, 4, 4, 16)
flatten (Flatten)	(None, 23552)
dense (Dense)	(None, 6)

```
SEED = 666
DATASET_NAME = "UCFCrimeDataset"
DATASET_PATH = "/home/abhishek/SmartCCTV/UCFCrimeDataset/"
TRAIN_CLASSES = [
    "Arson",
    "RoadAccidents",
    "Explosion",
    "Vandalism",
    "Shooting",
    "Normal",
]
# Feature Parameters
IMAGE_WIDTH = 92
IMAGE_HEIGHT = 92
IMAGE_DIMENSION = (IMAGE_WIDTH, IMAGE_HEIGHT)
SEQUENCE_LENGTH = 92
# dataset partitions
TRAIN_TEST_SPLIT = 0.25
TRAIN_VALID_SPLIT = 0.25
# early stopping callback parameters
EARLY_STOPPING_CALLBACK_MONITOR = "val_loss"
EARLY_STOPPING_CALLBACK_MIN_DELTA = 0.001
EARLY_STOPPING_CALLBACK_PATIENCE = 15
# optimizer parameters
LEARNING_RATE = 0.001
# training parameters
EPOCHS = 80
BATCH_SIZE = 15
```

LRCN model accuracy

- LOSS = 1.4958889484405518
- ACC. = 0.4838709533214569

Accuracy: ~48.4%
(collective training on 6 classes)

Whole observation report can be found here

Accuracy: On training Individual models i.e. e.g. just (arson & normal) or just (explosion & normal).

Arson

- LOSS = 0.2365783154964447
- ACC. = 0.8979591727256775

RoadAccidents

- LOSS = 0.5021342635154724
- ACC. = 0.7432432174682617

Explosion

- LOSS = 0.29725438356399536
- ACC. = 0.8979591727256775

Vandalism

- LOSS = 0.5319076180458069
- ACC. = 0.795918345451355

Shooting

- LOSS = 0.603776216506958
- ACC. = 0.7142857313156128

Future Works

- Improve the accuracy.
- Dataset filtering *(removing samples that may be confusing to the model)*.
- Add more anomalies *(if found suitable dataset for some)*.
- Monitoring and training side by side.
- Testing on RPI *(if not feasible then we will use server for monitoring and training side by side)*.

Thank You

References

1. *Real-Time Anomaly Recognition Through CCTV Using Neural Networks - ScienceDirect*
2. *Human Activity Recognition using TensorFlow (CNN + LSTM)*
3. *Anomaly Detection in Video Surveillance using Object Detection*
4. *Real-world Anomaly Detection in Surveillance Videos*