SUMMER OF CODES

SOC ‘25

REAL TIME ANOMALY DETECTION IN SURVEILLANCE VIDEOS

AARON BENSIER (24B2270)

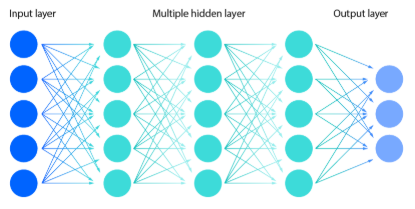
MENTOR : SHAIK REHNA AFROZ

**Computer Vision**

**Deep learning and neural network**

Deep learning is a subfield of machine learning that uses artificial neural networks with multiple layers (or "deep" networks) to learn and make predictions from data. These networks, inspired by the human brain, analyse data through layers of interconnected nodes, automatically extracting patterns and features.

**Relation between AI,ML,DL**

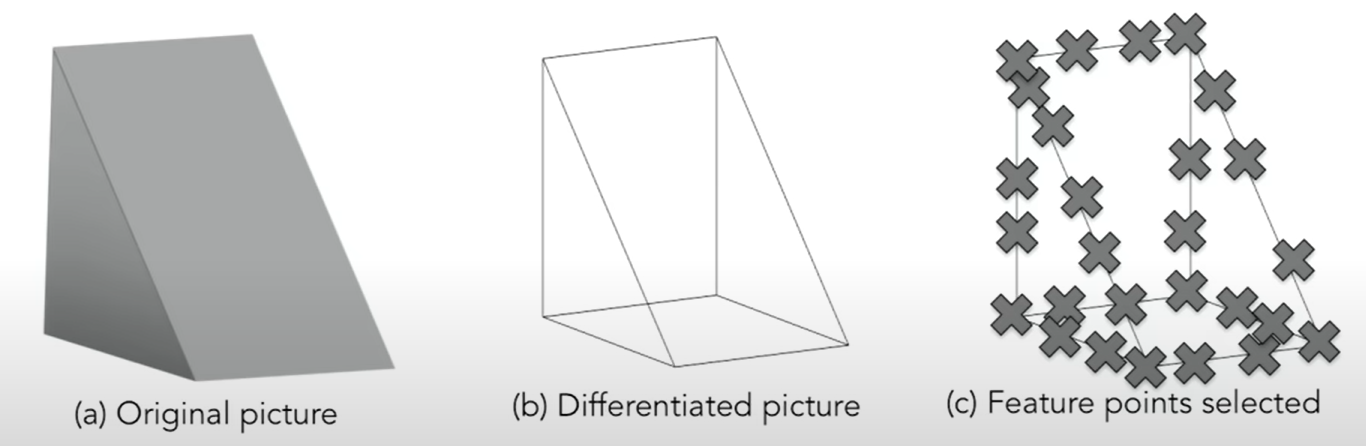


**Neural Network**

Neural networks are a type of machine learning algorithm inspired by the structure and function of the human brain. In neural networks, nodes (also called neurons) are organized into layers. These layers are primarily the input layer, hidden layers, and the output layer. Each node in a layer processes data received from the previous layer and passes its output to the next layer.

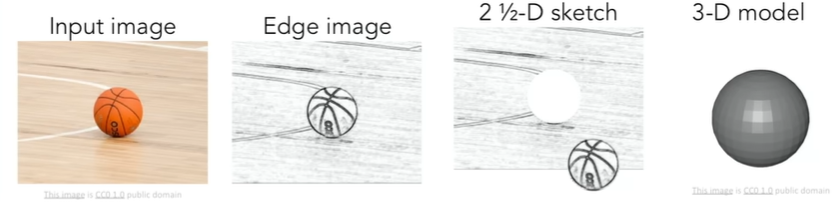
**How an image is processed before recognizing it?**

According to Larry Roberts, an image is recognised by first converting an original picture into a differentiated picture. Then converting it into image where feature points are highlighted.



This was one of the first step taken towards image recognition.

In the book ‘Vision’ written by David Marr, he clearly describes the methods of image recognition in his perspective.



According to David Marr, I mage recognition has three methods:

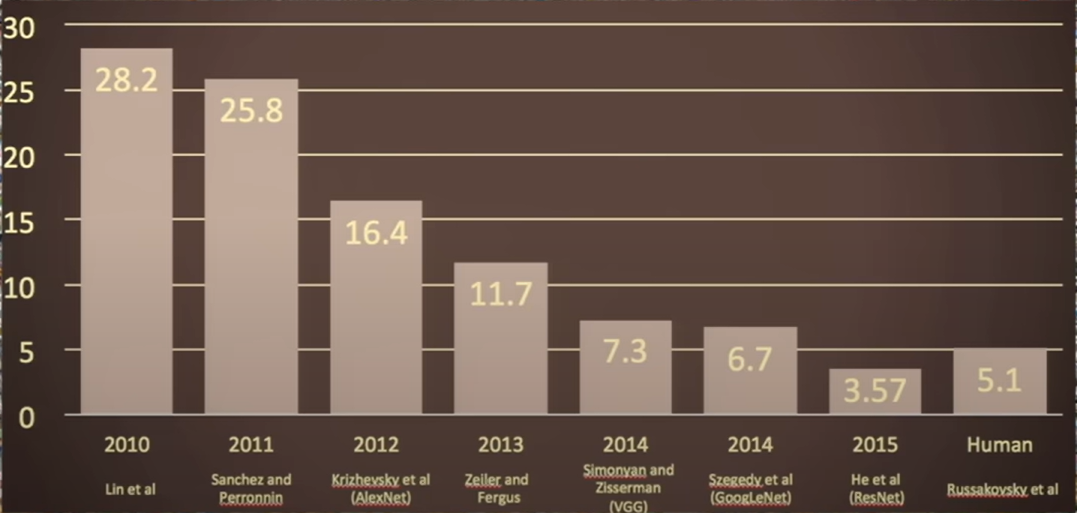
**Primal Sketch :** It notes down primary edges, curves and boundaries.

**2½ - D Sketch :** It notes down the depth of objects and surface orientations.

**3D Model Representation :** 3-D models hierarchically organized in terms of surface and volumetric primitives.

And then they came up with idea of feeding the model with many images(training data).

The main problem was with image classification.



This image summarizes the **performance progression** in the **ImageNet Large Scale Visual Recognition Challenge (ILSVRC)** from 2010 to 2015 for the **Image Classification Challenge**, which involves identifying objects among **1,000 categories** using over **1.4 million images**.

The **y-axis** shows the **Top-5 error rate (%)** and the **x-axis** shows the **year and the best-performing team** of that year.

This demonstrates how AI improved in imageunderstanding rapidly within 5 years. Hence, it marked the beginning of deep learning’s

dominance in computer vision.

Before going into further detail, let’s focus on the image.

An image is first represented as set of pixels

(each carrying a value).

This 6×6 grid is a grayscale image, where:

* Each cell (pixel) contains an intensity value between 0 (black) and 9 (white).
* The matrix is a simplified image — a very low-resolution grayscale input.

**An image represented as a matrix of integers**

Edge detection :

* Vertical edge detection

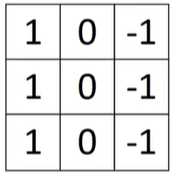


* Horizontal edge detection



When we convolute the image matrix with a specific filter, we get these kind of output.

Let us note that, the filter matrix for vertical and horizontal detection are both different. So what are those matrices?



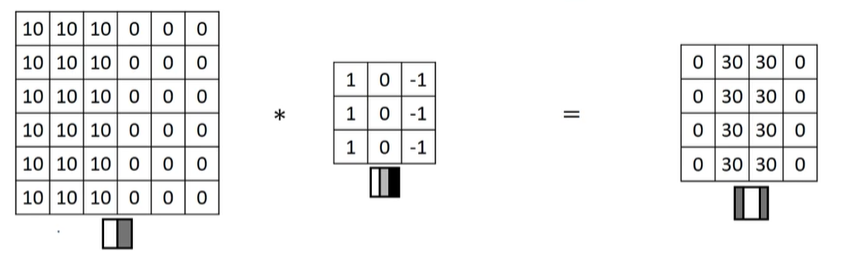
**Filter matrix for vertical**

**Filter matrix for horizontal edge detection**

**edge detection**

The basic operation of convolution of filter matrix with image matrix produce the matrix showing the desired detected edges.

Check out the following example to understand it better.



As we can see, the output depends on the filter matrix. Like making horizontal and vertical edges detections, we can make detectors for lines of every angle by changing the filter.

Therefore, our job is to find what values we should assign for this purpose in the place of filter matrices.

**Some important terms related to CV and their …………definition**

* **Padding -** Adding extra pixels (usually zeros) around the borders of an image or feature map before applying convolution. This helps preserve the spatial dimensions of the output and ensures edge pixels are processed equally.
* **Valid convolutions -** No padding is added, so the output size becomes smaller than the input.
* **Same convolutions -** Adding padding so the output size remains the same as the input.

**Strided convolutions –** It is a type of convolution where the filter moves over the input with steps larger than 1 (called the **stride**). Instead of sliding the filter one pixel at a time, it jumps by the stride value (e.g., stride = 2). This results in a **smaller output**, effectively down sampling the input.

**Python, tensor flow, keras integration**

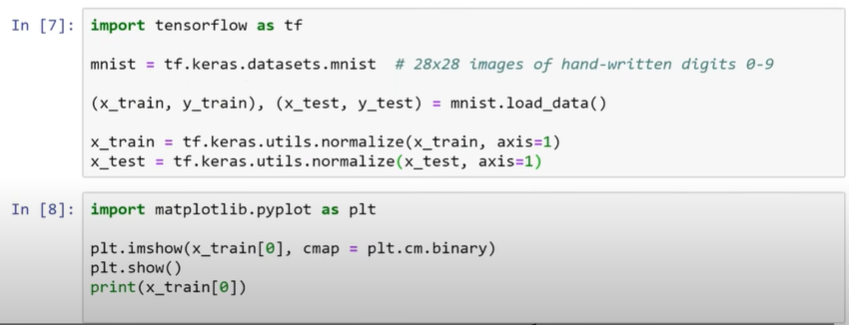
**TensorFlow** is a tool (a Python library) that helps us build and train machine learning models. With TensorFlow, you can classify images, recognize text or speech, and make predictions based on patterns in data.

**Keras** is a high-level API (a simpler interface) that runs on top of machine learning libraries like **TensorFlow**.

You can think of Keras as a shortcut or user-friendly layer that makes it much easier to build and train deep learning models. Instead of writing a lot of complex code, Keras lets you create powerful models with just a few lines.

2.5 **Functions used in python(in context to computer vision) and their explanations.**

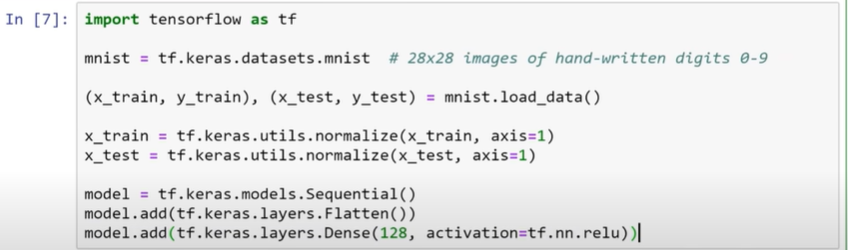
Let’s check a code that aims to load handwritten digit images (MNIST dataset), prepare the data, visualize one sample image, and print its pixel values.



Note :

* The third line is to “# Split the dataset into training and testing sets”
* The “normalize” function aims to “# Normalize the image data (scale pixel values to range 0 to 1)”
* In the input 8, the imshow function is used to “# Show the first image in the training set using a binary (black & white) color map”

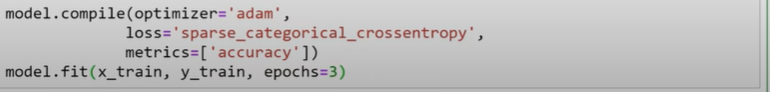
Now let’s check a little modification made in the code:



Note :

* The line “model = tf.keras.models.Sequential()” is to creates an empty model where you can stack layers in order.
* The line “model.add(tf.keras.layers.Flatten())” is to flatten the input, ie, this lets the next layer treat the image as a list of pixel values instead of a 2D image.
* The line “model.add(tf.keras.layers.Dense(128, activation=tf.nn.relu))” is to add a fully connected (hidden) layer with 128 neurons.

Now let’s check another addition made in the code:



Note :

* The compile function prepares the model for training. It is like when you tell TensorFlow how the model should learn.
* The optimizer controls how the model updates its weights during training. 'adam' is a popular and efficient choice — it adjusts learning rates automatically.
* The function “loss='sparse\_categorical\_crossentropy' “ is the function the model tries to minimize.
* The function “metrics=['accuracy']” tells the model to display accuracy (how many predictions are correct) as it trains.

**Special OpenCV functions**

| **Function Name** | **Purpose / Description** |
| --- | --- |
| **cv2.imread()** | **Loads an image from a specified file path.** |
| **cv2.imshow()** | **Displays an image in a window.** |
| **cv2.imwrite()** | **Saves an image to a specified file path.** |
| **cv2.cvtColor()** | **Converts an image from one color space to another (e.g., BGR to Grayscale, HSV, etc.).** |
| **cv2.resize()** | **Resizes an image to a specified width and height.** |
| **cv2.getRotationMatrix2D()** | **Creates a matrix for rotating an image around a point.** |
| **cv2.warpAffine()** | **Applies an affine transformation (e.g., rotation, translation) to an image.** |
| **cv2.GaussianBlur()** | **Applies a Gaussian blur to reduce image noise and detail.** |
| **cv2.blur()** | **Applies a normalized box filter (simple averaging blur).** |
| **cv2.medianBlur()** | **Applies a median blur filter, good for removing salt-and-pepper noise.** |

**Localisation**

**Some processes associated with localization are :**

* **Image classification – When you classify entire image as classes**
* **Object detection – When you detect objects within the image with rectangular boxes**
* **Image segmentation – When you classify pixels as classes**

**Sliding window object detection**

This demonstrating how to systematically scan an image with a fixed-size window to detect objects at different positions and scales. It's a hands-on walkthrough using TensorFlow and Keras, showing how to crop, classify, and filter potential detections across an image.



The red box moves around till it detects the dog completely inside it. The main problem while working on it is, what if there multiple boxes that detects the dog as per the model. Well, this is solved with the emerging algorithm called YOLO(You Only Look Once).

**Neural Network Output :**

(Suppose the output is to detect pedestrian class and dog class)

Then it will be of the format :

Here Pc represent the probability of getting the correct detection. C1 and C2 represents if a dog or a person is detected or not respectively.

Suppose the model detected multiple boxes around object. So which box will it chose depends on the probability of the image matching with the actual object. That is, if it is trying to detect the image of a person, then it will compare it with then image of actual people with which it was trained with and note down the probabilities of it for each rectangular bound. Then it will chose the best out of it by intersection over union method.