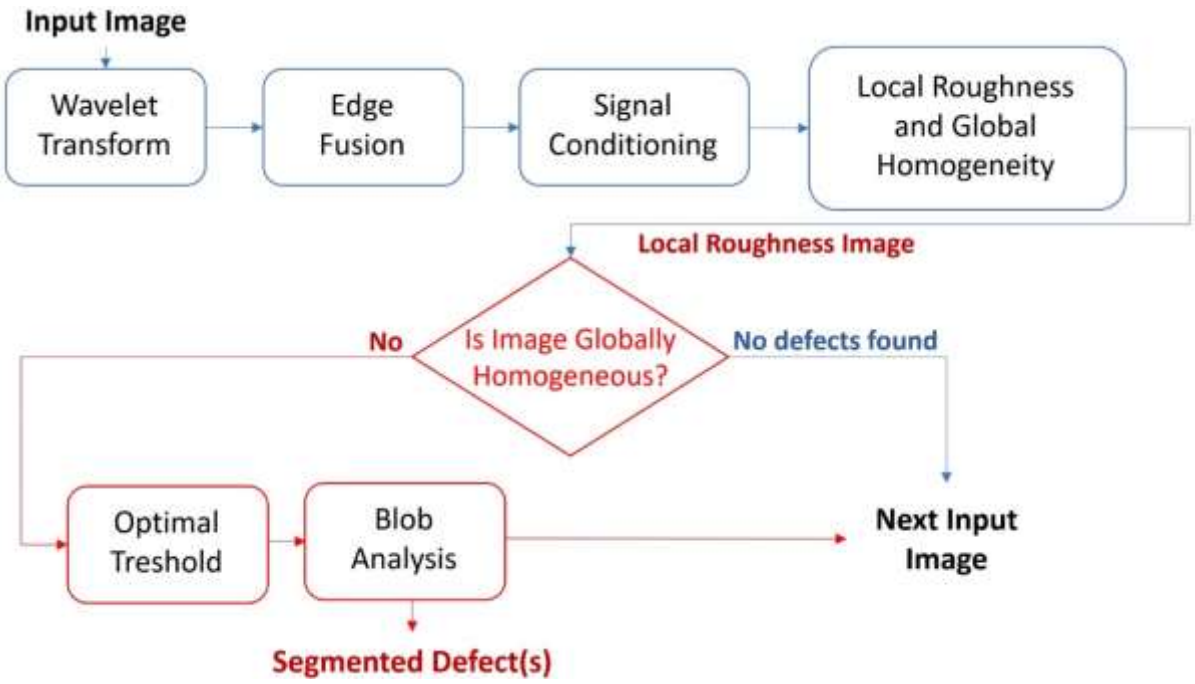
Dhvani Research and Development Solutions

- Harish B

1. Detecting Defects in the given Images :

Flowchart:



Edge Detection is a method of segmenting an image into regions of discontinuity. It is a widely used technique in digital image processing like

- ✓ pattern recognition
- ✓ image morphology
- ✓ feature extraction

Edge detection allows users to observe the features of an image for a significant change in the gray level. This texture indicating the end of one region in the image and the beginning of another. It reduces the amount of data in an image and preserves the structural properties of an image.

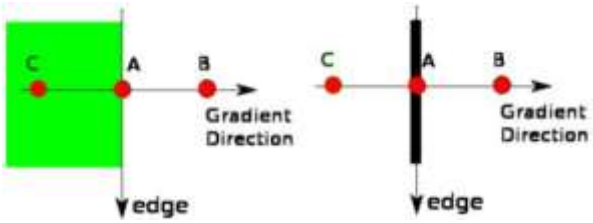
Canny Edge Detector Algorithm Stages:

- **Noise Reduction**: Since edge detection is susceptible to noise in the image, first step is to remove the noise in the image with a 5×5 Gaussian filter. Additionally, the localization error to detect the edge will slightly increase with the increase of the Gaussian filter kernel size.
- **Finding Intensity Gradient of the Image**: Smoothened image is then filtered with a Sobel kernel in both horizontal and vertical direction to get first derivative in horizontal direction (G_x) and vertical direction (G_y). From these two images, we can find edge gradient and direction for each pixel:

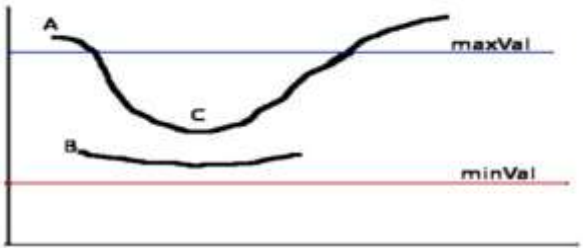
$$Edge\_Gradient\ (G) = \sqrt{G_x^2 + G_y^2}$$

$$Angle\ (\theta) = \tan^{-1}\left(\frac{G_y}{G_x}\right)$$

- **Non-maximum Suppression**: Non-maximum suppression is applied to find "the largest" edge. After applying gradient calculation, the edge extracted from the gradient value is still quite blurred.



- **Hysteresis Thresholding**: This stage decides which are all edges are really edges and which are not. For this, we need two threshold values, minVal and maxVal. Any edges with intensity gradient more than maxVal are sure to be edges and those below minVal are sure to be non-edges, so discarded.



This stage also removes small pixels noises on the assumption that edges are long lines. So, what we finally get is strong edges in the image.

Localizing the Defects:

Incorrect localization of composite edges: The Canny edge detector has a systematic error in localization whenever there is a composite edge. For example, if the edge is a composite of a step and a roof the maxima of $I * G'_\sigma$ are at

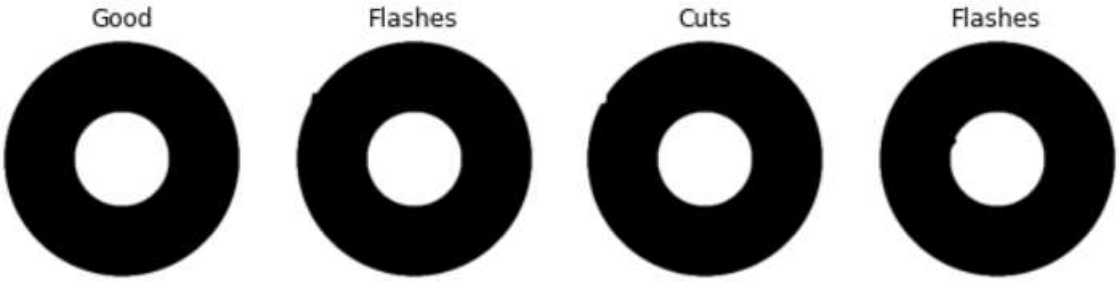
$x_\sigma = \sigma^2(k_2 - k_1)/h$ which is at the origin only if $k_1 = k_2$. Therefore, there is a systematic error for any non-zero value of a whenever the two ramps have differing slopes k_1, k_2. Such asymmetric edges are reasonably common in real images in the presence of shading gradients in the two regions bounding the edge.

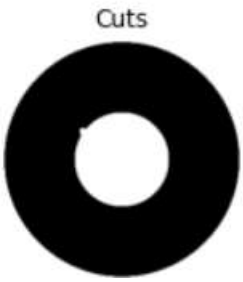
** To ensure correct localization, there should be a maximum at x = 0 for any combination of c_1, c_2 **

Classify the defect to flashes and cut marks:

A deep learning-based artificially intelligent system that can quickly train and identify faulty images. For this purpose, a pretrained convolution neural network based on the PyTorch framework is employed to extract discriminating features from the dataset, which is then used for the classification task. In order to eliminate the chances of overfitting, the proposed model also employed Dropout technology to adjust the network. The experimental study reveals that the system can precisely classify the normal and defective images.

```
plt.figure(figsize = (10 , 10))
for image_batch , label_batch in dataset.take(1):
    for i in range(5):
        plt.subplot(3 , 4 , i + 1)
        plt.imshow(image_batch[i].numpy().astype("uint8"))
        # print(image_batch.shape)
        # print(label_batch.numpy())
        plt.title(class_names[label_batch[i]])
        plt.axis('off')
```





2) Plot the following equation over time :

$$\dot{x} = a * (y - b)$$

$$\dot{y} = bx - y - xz$$

$$\dot{z} = xy - cz$$

The model is a system of three ordinary differential equations now known as the **Lorenz equations**:

$$\frac{\mathrm{d}x}{\mathrm{d}t} = \sigma(y - x),$$

$$\frac{\mathrm{d}y}{\mathrm{d}t} = x(\rho - z) - y,$$

$$\frac{\mathrm{d}z}{\mathrm{d}t} = xy - \beta z.$$

Hence, the z-motion decouples, leaving

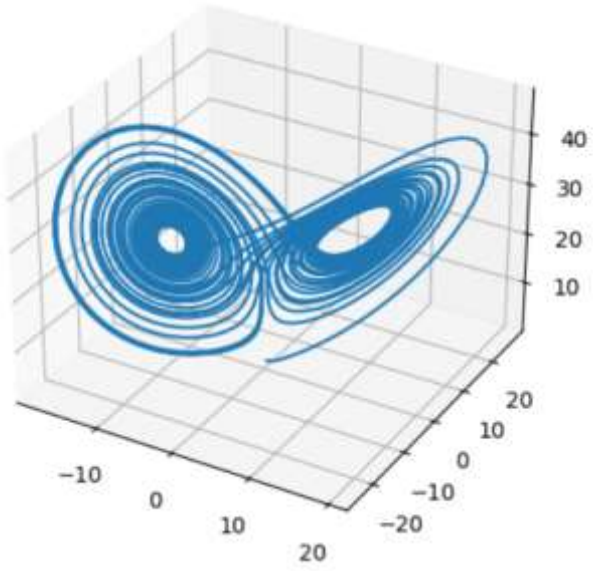
$$\begin{pmatrix} \dot{x} \\ \dot{y} \end{pmatrix} = \begin{pmatrix} -\sigma & \sigma \\ r & -1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$

with trace $\tau = -\sigma - 1 < 0$ and determinant $\Delta = \sigma(1 - r)$.

For $r > 1$, origin is a saddle point since $\Delta < 0$

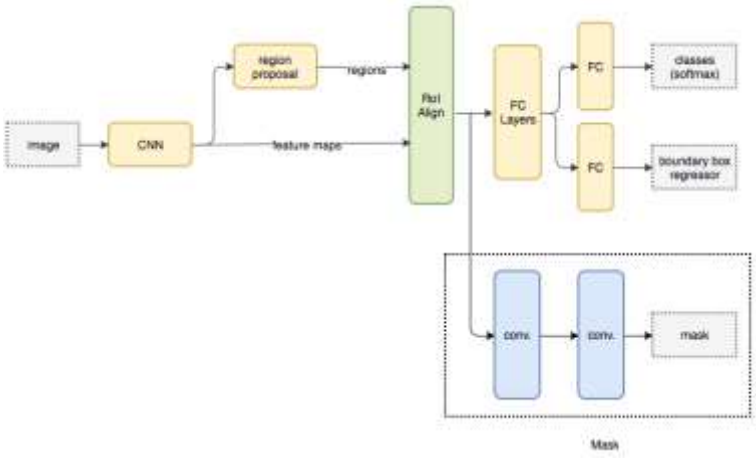
For $r < 1$, origin is a sink since $\tau^2 - 4\Delta = (\sigma + 1)^2 - 4\sigma(1 - r) = (\sigma - 1)^2 + 4\sigma\tau > 0$ → a stable node.

Actually, for $r < 1$ it can be shown that every trajectory approaches the origin as $t \to \infty$ the origin is globally stable, hence there can be no limit cycles or chaos for $r < 1$.



3) Predict location and type of vehicle :

<u>Flowchart:</u>



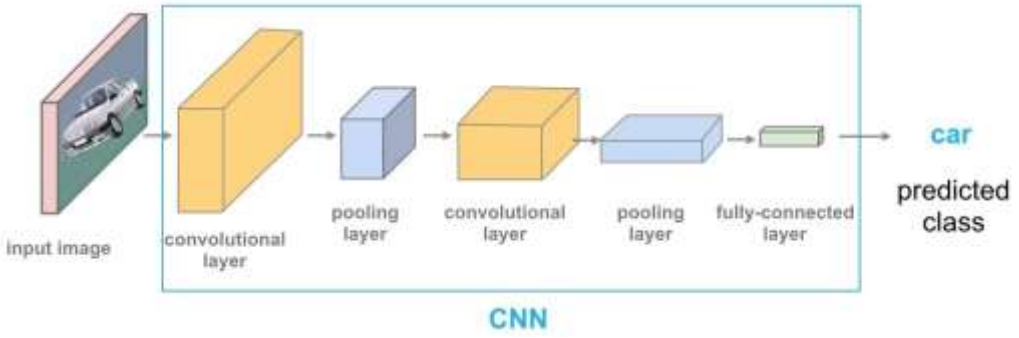
Deep Learning Model to Identify Vehicles:

Mask R-CNN is a Convolutional Neural Network (CNN) and state-of-the-art in terms of image segmentation. This variant of a Deep Neural Network detects objects in an image and generates a high-quality segmentation mask for each instance.

The **Convolutional Neural Network** Architecture consists of three main layers:

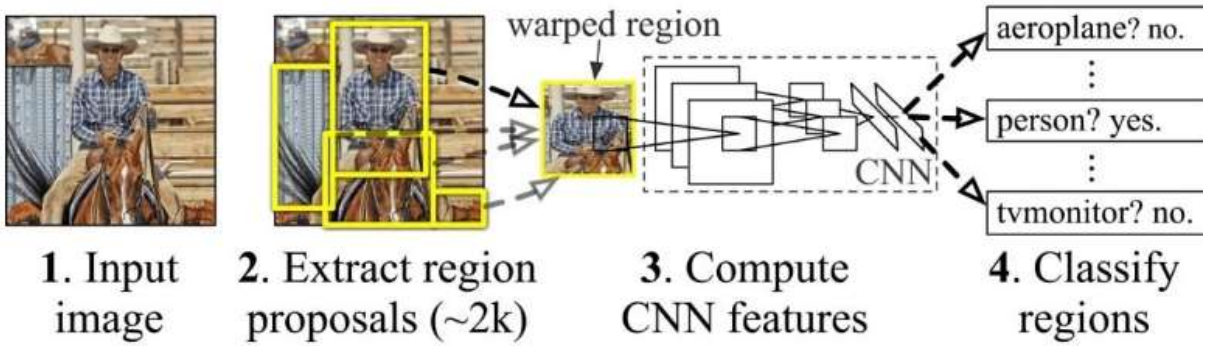
- Convolutional layer: This layer helps to abstract the input image as a feature map via the use of filters and kernels.
- Pooling layer: This layer helps to down sample feature maps by summarizing the presence of features in patches of the feature map.
- Fully connected layer: Fully connected layers connect every neuron in one layer to every neuron in another layer.

Combining the layers of a CNN enables the designed neural network to learn how to identify and recognize the object of interest in an image. Simple Convolutional Neural Networks are built for image classification and object detection with a single object in the image.



R-CNN or RCNN, stands for Region-Based Convolutional Neural Network, it is a type of machine learning model that is used for computer vision tasks, specifically for object detection. The image below depicts the concept of region-based CNN (R-CNN). This approach utilizes bounding boxes across the object regions, which then evaluates convolutional networks independently on all the

Regions of Interest (ROI) to classify multiple image regions into the proposed class. The RCNN architecture was designed to solve image detection tasks.



Mask R-CNN, or Mask RCNN, is a Convolutional Neural Network (CNN) and state-of-the-art in terms of image segmentation and instance segmentation.

Mask R-CNN was built using Faster R-CNN. While Faster R-CNN has 2 outputs for each candidate object, a class label and a bounding-box offset, Mask R-CNN is the addition of a third branch that outputs the object mask. The additional mask output is distinct from the class and box outputs, requiring the extraction of a much finer spatial layout of an object. Mask R-CNN is an extension of Faster R-CNN and works by adding a branch for predicting an object mask (Region of Interest) in parallel with the existing branch for bounding box recognition.

Evaluation Metrics:

1) Choosing Output Format

Tracking with bounding boxes

Just like in object detection with bounding boxes one could track objects by finding the center of the rectangle. Following output file looks like this: **(frame number, class, left, top, right, bottom, confidence)**

where (left, top) is the topmost-left and (right, bottom) is the bottommost-right point coordinates of the bounding box

Tracking with center (centroid) coordinates:

Finding the center of binary mask was done using **cv2.findContours()** and **cv2.moments()**. Here's the following output file format: **(frame number, class, center_left, center_top, confidence)**

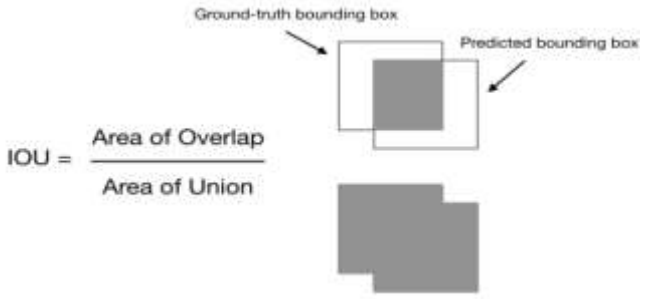
2) Evaluating COCO-trained Model:

Intersection over Union (IoU)

Evaluates the overlap between two bounding boxes. To apply IoU we need:

1. Ground-truth bounding boxes (Annotations)
2. Predicted bounding boxes from our model

IoU is measured by dividing the overlapping area by the area of union between them.



True Positive, False Positive, False Negative and True Negative

- ✓ **True Positive (TP):** A correct detection. Detection with IOU ≥ threshold
- ✓ **False Positive (FP):** A wrong detection. Detection with IOU < threshold
- ✓ **False Negative (FN):** A ground truth not detected
- ✓ **True Negative (TN):** Not used by the metrics.

Precision: It shows how the model is able to identify only the relevant objects.

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

Recall: It shows how the model is able to find all relevant cases (all ground truth bounding boxes).

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

Note:

** Though the output is not yet compiled properly, it results in error. This is the idea upon my knowledge to use Mask RCNN for classifying the vehicle detection from the given datasets **

THANK YOU