# Image Edge Detection Using Convolutional Neural Network (CNN) Layers with TensorFlow

## 1. Introduction

Edge detection is a fundamental task in image processing used to identify object boundaries by detecting discontinuities in pixel intensity. Traditional edge detectors using fixed kernels. In contrast, Convolutional Neural Networks (CNNs) can learn edge features automatically or apply predefined convolution filters efficiently.

This project demonstrates how CNN convolution layers in TensorFlow can be used to perform edge detection on images.

## 2. Objectives :-

- To understand convolution operations in CNNs
- To implement edge detection using CNN layers
- To visualize detected edges using TensorFlow and Matplotlib

## 3. Tools and Technologies:-

**Programming Language:-** Python

**Libraries:-** TensorFlow, NumPy, Matplotlib

## 4. Convolution in CNN:-

**Convolution** is the core operation in a CNN. It allows the network to automatically extract meaningful features (such as edges, corners, textures) from input data—especially images.

- **Key Components of Convolution in CNN:-**
  - ✓ Kernel (Filter)
  - ✓ Stride
  - ✓ Padding
  - ✓ Feature Maps

## 5. Applying CNN To an Image :-

#Implement image edge detection using Convolution network layer (CNN) layers with Tensorflow.

```
import numpy

import tensorflow as tf

import matplotlib.pyplot as plt

from itertools import product

#set the param

plt.rc('figure',autolayout=True)

plt.rc('image',cmap='magma')

#define the kernel

kernel = tf.constant([[-1,-1,-1],[-1,8,-1],[-1,-1,-1],])

#loading image

image=tf.io.read_file('mk.jpg')

image=tf.io.decode_jpeg(image, channels=1)

image=tf.image.resize(image, size=[300,300])

#plot the image

img = tf.squeeze(image).numpy()

plt.figure(figsize=(5,5))

plt.imshow(img,cmap='gray')

plt.axis('off')

plt.title('original gray scale image')

plt.show();

#Reformat

image=tf.image.convert_image_dtype(image,dtype=tf.float32)

image=tf.expand_dims(image,axis=0)
```

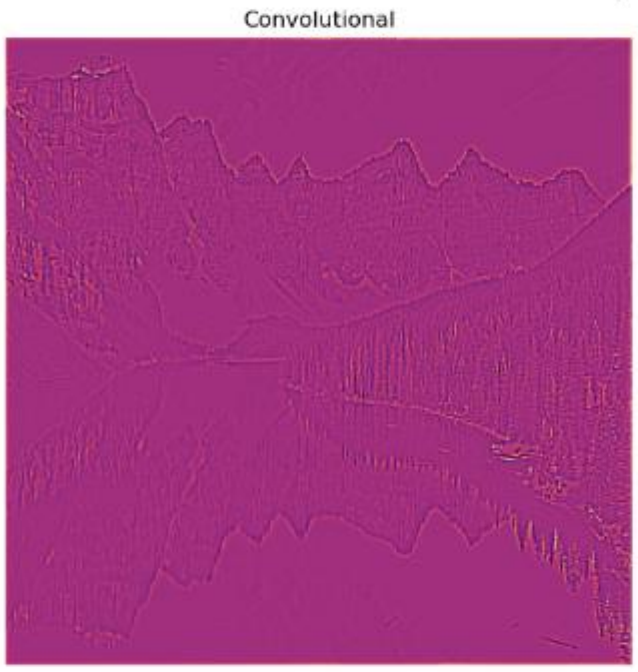
```
kernel=tf.reshape(kernel,[*kernel.shape,1,1])
kernel=tf.cast(kernel,dtype=tf.float32)
#Convolution layer
conv_fn=tf.nn.conv2d
image_filter=conv_fn(input=image,filters=kernel,strides=1,padding='SAME',)
plt.figure(figsize=(15,5))
#Plot convolved image
plt.subplot(1,3,1)
plt.imshow(tf.squeeze(image_filter))
plt.axis('off')
plt.title('Convolutional')
#Activation Layer
relu_fn = tf.nn.relu
#image detection
image_detect=relu_fn(image_filter)
plt.subplot(1,3,2)
plt.imshow(tf.squeeze(image_detect))
plt.axis('off')
plt.title('Activation')
#Pooling Layer(fixed)
image_condense=tf.nn.pool(
    input=image_detect,
    window_shape=(2,2),
    pooling_type='MAX',
    strides=(2,2),
    padding='SAME')
#Display all results
```

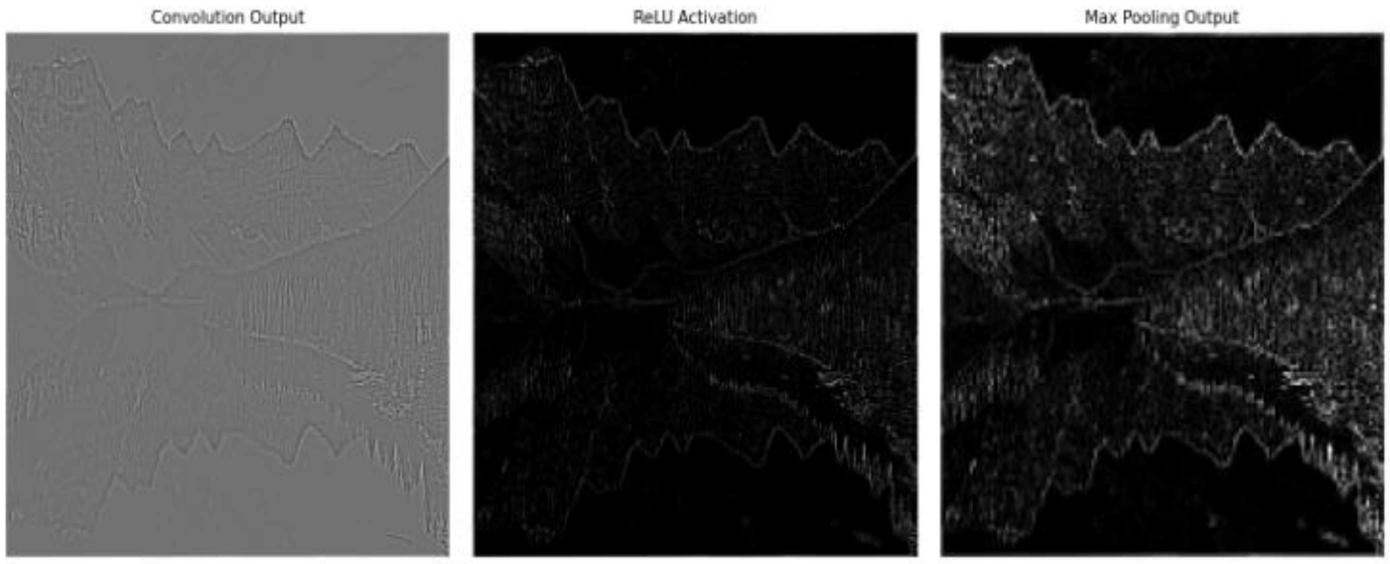
```
plt.figure(figsize=(15,5))
plt.subplot(1,3,1)
plt.imshow(tf.squeeze(image_filter),cmap='gray')
plt.title('Convolution Output')
plt.axis('off')
plt.subplot(1,3,2)
plt.imshow(tf.squeeze(image_detect),cmap='gray')
plt.title('ReLU Activation')
plt.axis('off')
plt.subplot(1,3,3)
plt.imshow(tf.squeeze(image_condense),cmap='gray')
plt.title('Max Pooling Output')
plt.axis('off')
plt.show()
```

- Load an image

- Convert it to grayscale

- Convert into convolution kernels





## 6. Implementation (TensorFlow Code):-

**Install Required Libraries:-**

pip install tensorflow , matplotlib , numpy

## 7. Results:-

The CNN convolution layers successfully highlight object boundaries.

Horizontal and vertical edges are clearly detected.

CNN-based convolution provides flexibility and efficiency over manual pixel processing.

## 8. Advantages:-

Efficient computation using GPU acceleration

Can be extended to learn filters automatically

Integrates easily with deep learning pipelines

## 9. Applications:-

- Medical image analysis

- Object detection
- Facial recognition
- Image segmentation

## 10. Conclusion:-

This project demonstrates that CNN convolution layers in TensorFlow can effectively perform image edge detection. By using predefined kernels or trainable filters, CNNs provide a powerful and scalable approach compared to traditional edge detection methods.

## 11. Future Scope:-

Train CNN to learn edge filters automatically

Use deeper CNN architectures

Combine with object detection models