Edge Detection Using CNN with TensorFlow

1. Introduction

Edge detection is a fundamental computer vision task that identifies object boundaries in images. Traditional methods include Sobel and Canny operators, but Convolutional Neural Networks (CNNs) provide a learnable and flexible approach. This project demonstrates edge detection using CNN layers in TensorFlow.

2. Objective

Implement edge detection using CNN layers.
Visualize the effects of convolution, activation, and pooling.
Understand how CNNs extract and condense image features.

3. Methodology

3.1 Tools

TensorFlow for convolution operations
NumPy for numerical handling
Matplotlib for visualization

3.2 Process

1.Load and preprocess image:

Grayscale conversion, resizing to 500x400, normalization to [0,1].

2.Define convolution kernel:

Laplacian kernel emphasizes edges:
[[-1, -1, -1],
 [-1, 8, -1],
 [-1, -1, -1]]

3.Convolution layer:

Highlights intensity changes via tf.nn.conv2d.

4.Activation layer:

ReLU applied to remove negative values (tf.nn.relu).

5.Pooling layer:

Max pooling (tf.nn.pool) reduces dimensions while retaining strong edges.

6.Visualization:

Outputs of convolution, activation, and pooling plotted using Matplotlib.

Example: Applying CNN to an Image

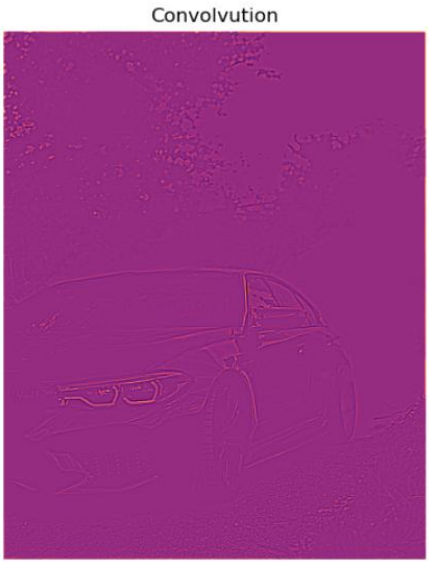
Input Image:

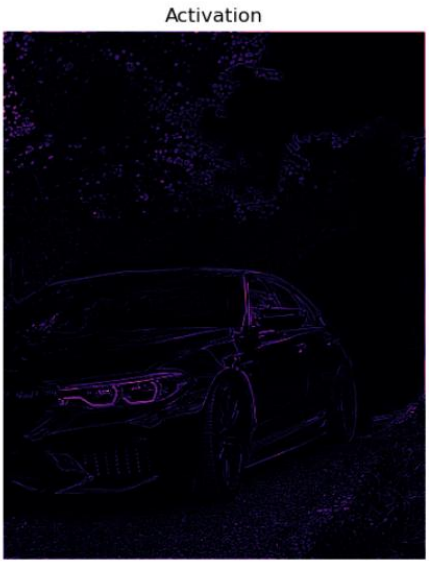
```python
#implement image edge detection using convolution neural network(CNN) layers with tensorflow
import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt
from itertools import product

#set the parameter
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]])

#load image
image=tf.io.read_file('car.jpg')
image=tf.io.decode_jpeg(image,channels=1)
image=tf.image.resize(image,size=[500,400])

#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)
```

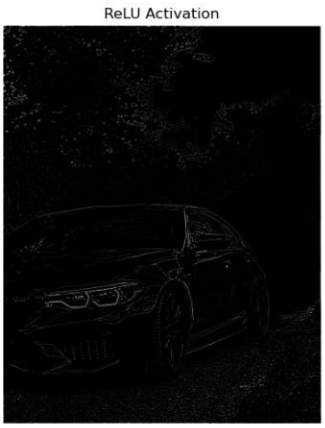
```
#Convolvution Layer
conv_fn=tf.nn.conv2d
image_filter=conv_fn(input=image,filters=kernel,strides=1,padding='SAME')
plt.figure(figsize=(15,5))

#plot the convolved image
plt.subplot(1,3,1)
plt.imshow(tf.squeeze(image_filter))
plt.axis('off')
plt.title("Convolvution")

#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
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("Convolvution 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()
```

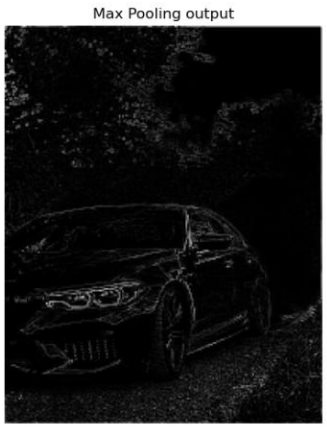
4. Results

Original Image: Grayscale representation of input.
Convolution Output: Edges and contours become prominent.
Activation Output: Negative values removed; edges appear sharper.
Pooling Output: Spatial dimensions reduced; strong edges preserved.









5. Discussion

Convolution highlights high-intensity changes (edges).
ReLU retains strong positive features.

Max pooling condenses spatial information without losing important edges.
This pipeline demonstrates fundamental CNN operations on images.

6. Conclusion

The project successfully implements edge detection using CNN layers:
Convolution extracts edges.

ReLU enhances features.
Pooling reduces dimensionality.
This pipeline can be extended to more complex image processing tasks like segmentation and object detection.

7. Future Work

Experiment with different kernels (Sobel, Prewitt).
Apply multiple convolutional layers for hierarchical feature extraction.
Extend to colour images and larger datasets.

8. References

A. Deep Learning, MIT Press, 2016,

Gonzalez, R. C., & Woods, R. E. Digital Image Processing, 4th Edition, 2018.

TensorFlow Documentation: https://www.tensorflow.org