Image Processing Using Convolution, ReLU, and Pooling in TensorFlow

1. Objective

The purpose of this project is to implement basic image processing using a simple Convolutional Neural Network (CNN) pipeline. Specifically, the project aims to:

1. Load and preprocess a grayscale image.
2. Apply convolution using a kernel to detect edges.
3. Use the ReLU activation function to highlight important features.
4. Apply max pooling to reduce spatial dimensions while retaining key features.
5. Visualize the outputs at each stage.

2. Introduction

Image processing and feature extraction are essential steps in computer vision. Convolutional Neural Networks (CNNs) are widely used for these tasks, with convolution layers extracting features, activation layers introducing non-linearity, and pooling layers condensing information.

In this project, a grayscale image is processed using a **manual CNN-like pipeline** in TensorFlow. This demonstrates the fundamental operations of convolution, activation, and pooling used in modern neural networks.

3. Tools and Technologies Used

- **Programming Language:** Python
- **Libraries:**
  - TensorFlow (for image processing and convolution operations)
  - NumPy (for array manipulation)
  - Matplotlib (for visualization)
- **Image Format:** JPG
- **Development Environment:** Jupyter Notebook

4. Theory

4.1 Convolution

Convolution is a mathematical operation used to extract features from images. A convolution kernel (filter) slides over the image, computing weighted sums to highlight specific patterns like edges or textures.

Kernel used in this project:
[
\begin{bmatrix}
-1 & -1 & -1 \
-1 & 8 & -1 \
-1 & -1 & -1
\end{bmatrix}
]

This kernel is a high-pass filter that emphasizes regions with high intensity changes, effectively detecting edges.

4.2 Activation Function (ReLU)

The ReLU (Rectified Linear Unit) function is used to introduce non-linearity by setting all negative values to zero:
[
f(x) = \max(0, x)
]
This allows the network to focus on significant features while ignoring irrelevant negative values.

4.3 Pooling

Pooling reduces the spatial dimensions of the feature map while retaining important information. Max pooling selects the maximum value in a defined window, reducing computational complexity and making feature detection more robust.

---

5. Methodology

The project was implemented using the following steps:

1. **Image Loading and Preprocessing**
   - Load the image `paju.jpg` using TensorFlow.
   - Convert the image to grayscale and resize it to 300 × 300 pixels.
   - Normalize the image values to [0,1] range.
2. **Convolution**
   - Define a 3×3 edge-detection kernel.

    - Apply convolution to the image using `tf.nn.conv2d`.
3. **Activation (ReLU)**
    - Apply the ReLU function to the convolution output to highlight positive features.
4. **Pooling**
    - Apply max pooling with a 2×2 window to condense the feature map.
5. **Visualization**
    - Plot the original image, convolution output, activation output, and pooled output using Matplotlib.

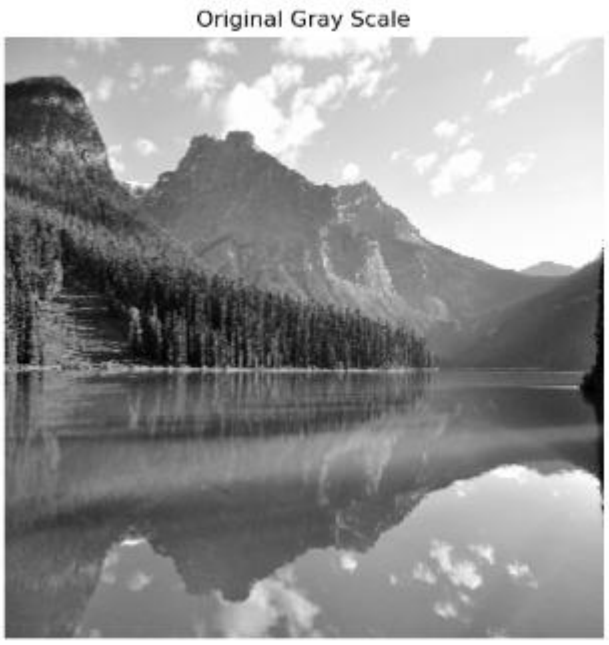
---

6. Implementation

```
import numpy as np

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]])

#Load the Image

image = tf.io.read_file('paju.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.title('Original Gray Scale')
plt.axis('off')
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)

conv_fn=tf.nn.conv2d
image_filter=conv_fn(input=image ,filters=kernel,
strides=1,padding='SAME',)
plt.figure(figsize=(15,5))
#plot the convolution
plt.subplot(1,3,1)
plt.imshow(tf.squeeze(image_filter))
plt.axis('off')
plt.title('convolution')

#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'
)
#Dislaying figure
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(1,3,2)
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')
```

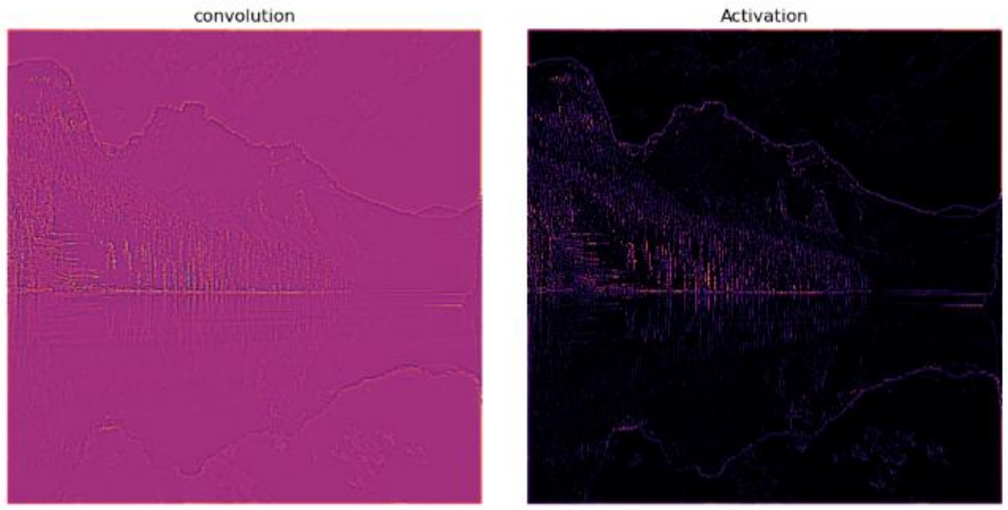
7. Results

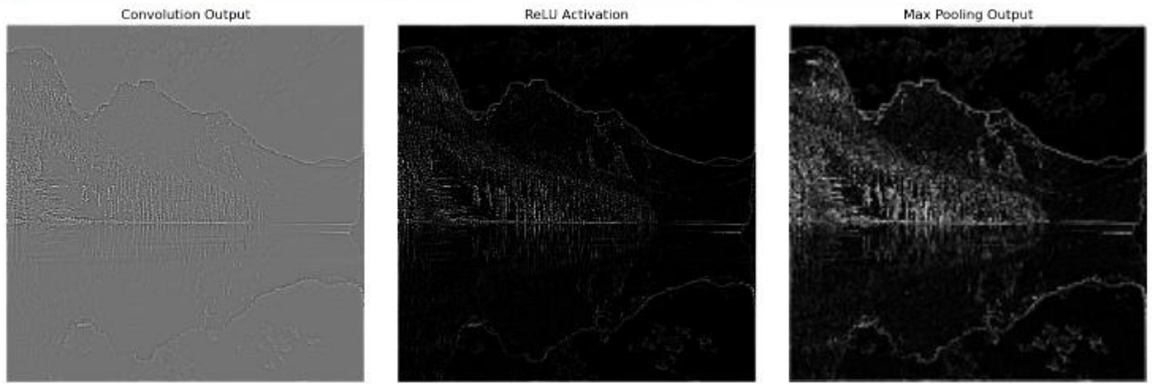
1. **Original Grayscale Image:**



   - Smooth image with visible intensity variations.
2. **Convolution Output:**
   - Edges and boundaries in the image are highlighted.
   - Both horizontal and vertical edges are enhanced by the kernel.
3. **ReLU Activation Output:**
   - Negative values are removed, focusing on significant features.
   - Edge highlights become more prominent.
4. **Max Pooling Output:**
   - Spatial dimensions are reduced.
   - Main features are retained, while fine details and noise are suppressed.

```
[29]:  (np.float64(-0.5), np.float64(149.5), np.float64(149.5), np.float64(-0.5))
```





8. Discussion

- The convolution kernel successfully detected edges in the image.
- ReLU activation enhanced important features by removing irrelevant negative values.
- Max pooling reduced the feature map size while preserving key information, which is useful for efficient computation in CNNs.
- Minor noise is amplified in the convolution output, which is normal for high-pass filters. Preprocessing such as Gaussian smoothing could improve results.

---

9. Conclusion

This project demonstrates the fundamental operations of a CNN pipeline:

1. Edge detection via convolution.
2. Feature enhancement via ReLU activation.
3. Dimensionality reduction via max pooling.

The project highlights how TensorFlow can be used for image processing tasks and provides insight into how CNNs process and extract features from images.

10. Future Scope

- Apply more advanced kernels (Sobel, Prewitt) for edge detection.
- Extend to color images and multi-channel convolutions.
- Incorporate additional CNN layers for more complex feature extraction.
- Use this pipeline as part of a complete image classification model.
- Experiment with different pooling techniques (average pooling, global pooling).

11. References

1. TensorFlow Documentation: https://www.tensorflow.org
2. Gonzalez, R. C., & Woods, R. E. *Digital Image Processing*.
3. Matplotlib Documentation: https://matplotlib.org
4. NumPy Documentation: https://numpy.org