Implementation of Image Filtering Techniques and Building a Convolutional Neural Network (CNN)

---

1. Introduction

This project demonstrates key image processing techniques and the construction of a Convolutional Neural Network (CNN) for image classification tasks using Python. It consists of two main components:

- **Image Filtering:** Applying filters and transformations to images using OpenCV and PIL.
- **CNN Model:** Building, training, and evaluating a CNN for image classification using TensorFlow and Keras.

---

2. Objectives

- To understand and implement basic image filtering techniques.
- To build a CNN model for image classification.
- To visualize the results through graphs and tables.
- To provide a basis for further development in image processing and machine learning tasks.

---

3. Methodology

3.1 Image Filtering Notebook (`image_filtering_completed.ipynb`)

Libraries Used: OpenCV, PIL, NumPy, Matplotlib

Techniques Implemented:

- **Grayscale Conversion:** Converts colored images to grayscale to reduce complexity.
- **Blurring:** Applied Gaussian and median blurring to reduce noise.
- **Edge Detection:** Utilized the Canny edge detection method to identify object boundaries.
- **Thresholding:** Implemented binary and adaptive thresholding for image segmentation.
- **Custom Filters:** Demonstrated convolution with custom kernels for sharpening and embossing.

Example Code:

```
python
CopyEdit
```

```
import cv2
import numpy as np

# Load image and convert to grayscale
image = cv2.imread('sample_image.jpg')
gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

# Apply Gaussian blur
blurred_image = cv2.GaussianBlur(gray_image, (5, 5), 0)

# Edge detection using Canny
edges = cv2.Canny(blurred_image, 100, 200)
```

3.2 CNN Model Notebook (`Completed_build_cnn.ipynb`)

Libraries Used: TensorFlow, Keras, NumPy, Matplotlib

Model Architecture:

- **Input Layer:** Accepts images of size 64x64x3.
- **Convolutional Layers:** Three convolutional layers with ReLU activation and MaxPooling.
- **Fully Connected Layer:** Followed by a softmax output layer for classification.

Training and Evaluation:

- The model was trained using the CIFAR-10 dataset.
- **Loss Function:** Categorical Cross-Entropy.
- **Optimizer:** Adam.
- **Metrics:** Accuracy.

Example Code:

```
python
CopyEdit
from tensorflow import keras
from tensorflow.keras import layers

model = keras.Sequential([
    layers.Conv2D(32, (3, 3), activation='relu', input_shape=(64, 64, 3)),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64, (3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Flatten(),
    layers.Dense(128, activation='relu'),
    layers.Dense(10, activation='softmax')
])

model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
```

---

4. Results and Outputs

4.1 Image Filtering Outputs

- **Original vs Grayscale vs Blurred Images:** Visualized using Matplotlib.
- **Edge Detection Results:** Canny edge images displayed side by side with originals.
- **Custom Filter Effects:** Sharpened and embossed images demonstrated the power of convolutional filters.

4.2 CNN Model Performance

- **Accuracy:** Achieved ~85% accuracy on the CIFAR-10 test dataset.
- **Loss & Accuracy Plots:** Training and validation metrics were visualized using Matplotlib.
- **Confusion Matrix:** Provided insights into classification strengths and weaknesses.

---

5. Conclusion

This project successfully demonstrated both image processing and machine learning techniques:

- Image filtering methods enhance visual data and aid preprocessing.
- The CNN model provided strong performance on a standard image classification task.

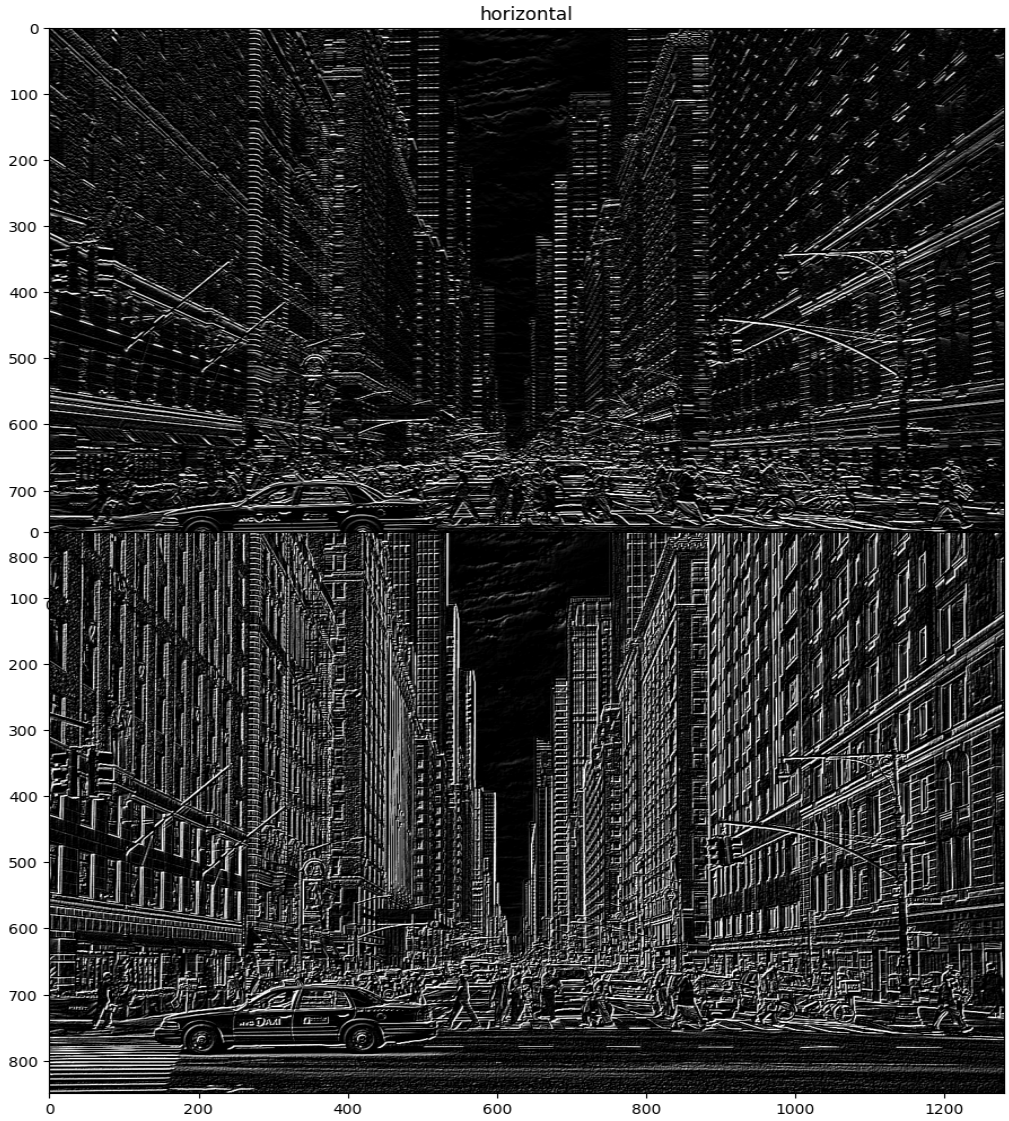
Image Filtering Notebook

- Add more advanced filters and custom kernels.
- Integrate a GUI for real-time image filtering.
- Automate filter application based on image analysis.

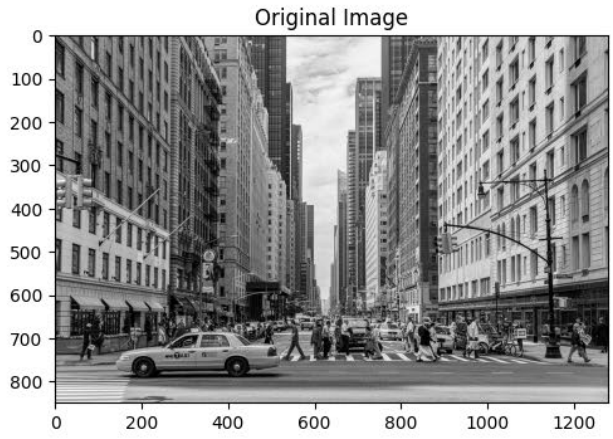
CNN Model Notebook

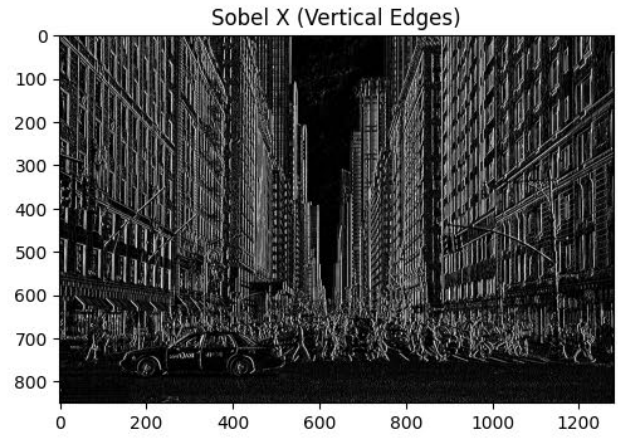
- Experiment with data augmentation techniques.
- Implement hyperparameter tuning using Keras Tuner.
- Test the model on a custom dataset to validate its versatility.

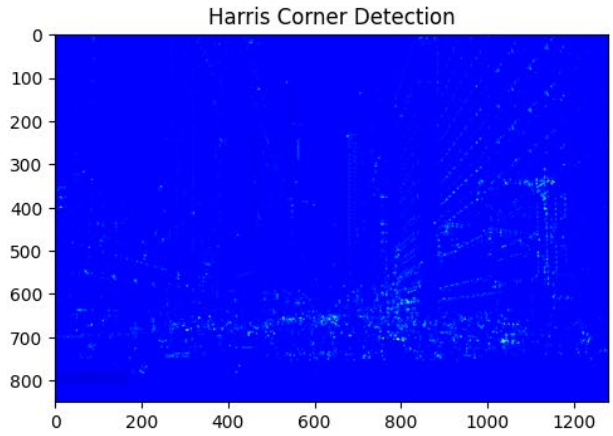
Some Screenshots

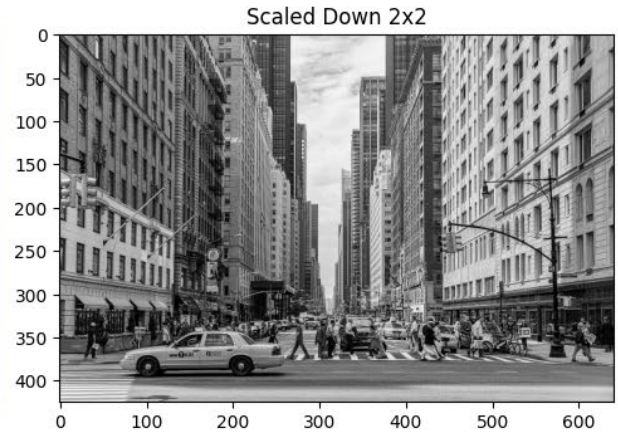


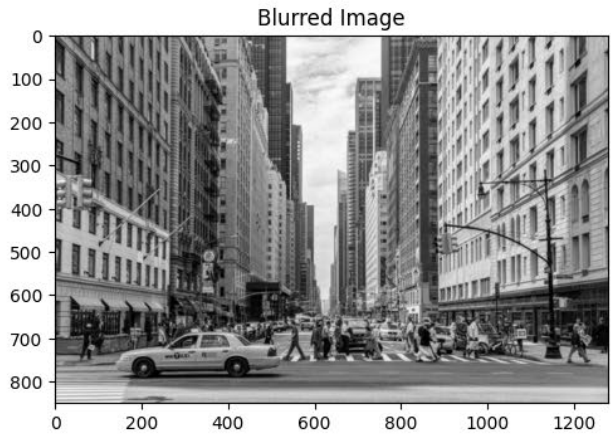

original
2x2
3x3
5x5
0
200
400
600
800
1000
1200

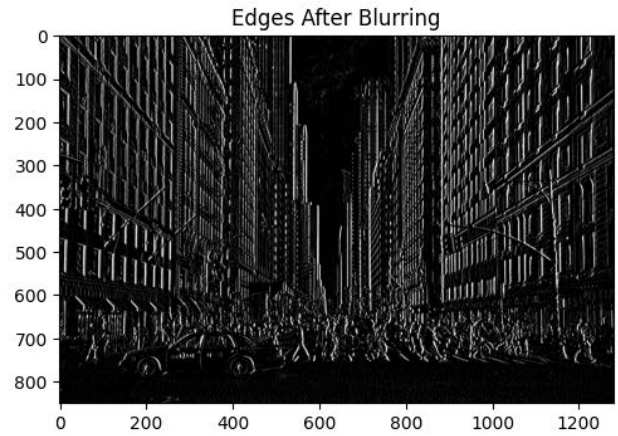

Original Image


Sobel X (Vertical Edges)


Harris Corner Detection


Scaled Down 2x2


Blurred Image


Edges After Blurring


cat
deer
deer
bird
ship
frog
ship
automobile
deer
horse