Combined Report: Image Filtering and Convolutional Neural Network (CNN) Implementation

1. Overview

This project encompasses two key components:

1. **Image Filtering using Convolution:** Various image processing techniques such as edge detection, blurring, and corner detection were implemented to preprocess images for feature extraction.
2. **Building a Convolutional Neural Netw ork (CNN):** A CNN was designed and trained on the CIFAR-10 dataset to classify images into one of ten categories. The architecture, training, and evaluation procedures were completed with an emphasis on comparing optimization techniques and analysing the results.

---

2. Part 1 – Image Filtering Using Convolution

Objectives

- To explore convolution operations for edge detection, blurring, and corner detection.
- To preprocess images effectively, improving the performance of downstream tasks like image classification.

Implementation

Edge Detection:

- **Horizontal, Vertical, and Diagonal Filters:** Various 3x3 kernels were created for edge detection. These kernels highlighted different edges in the image, including horizontal and vertical edges using the Sobel operator, as well as diagonal edges.

Blurring:

- **Averaging Filters (2x2, 3x3, 5x5):** These filters were used to blur images by averaging pixel values in their neighborhood, reducing noise and smoothing the image.

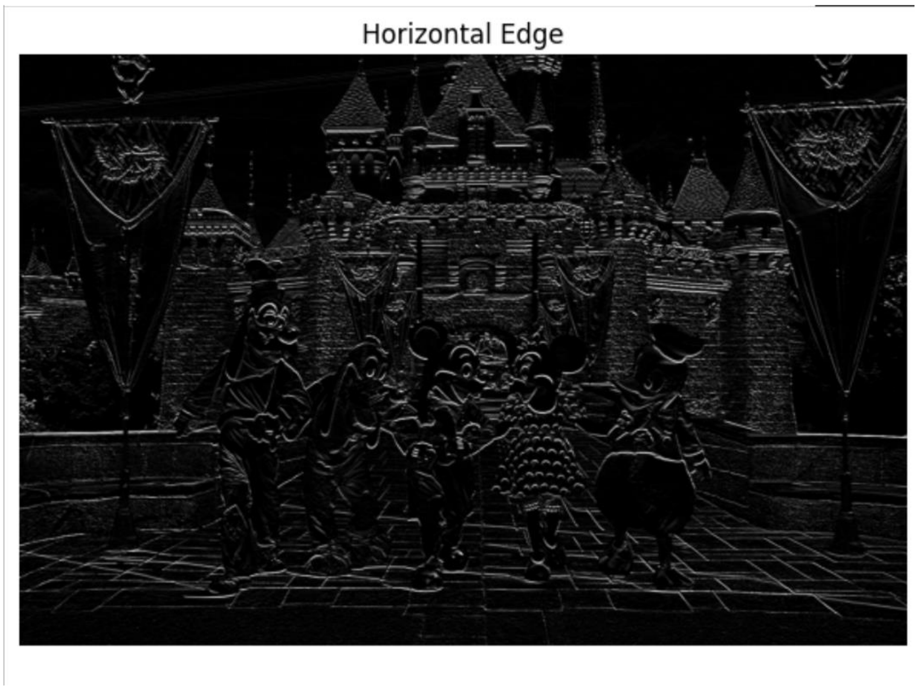
Corner Detection:

- **Harris Corner Detection:** This technique was implemented to identify corners in the image, which can be useful in image matching and recognition.

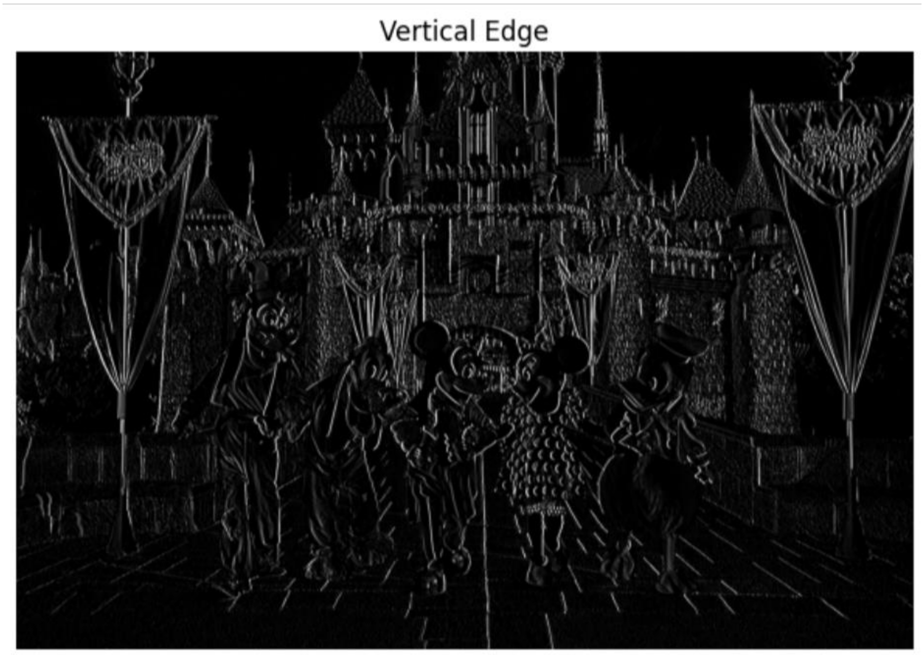
Visualizations

Edge Detection

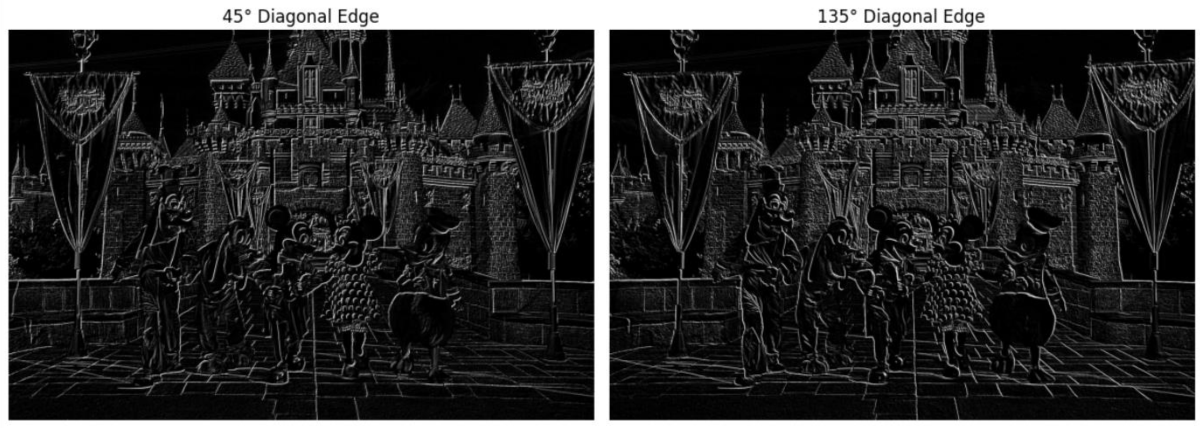
- **Horizontal Edge Detection:**



- **Vertical Edge Detection:**

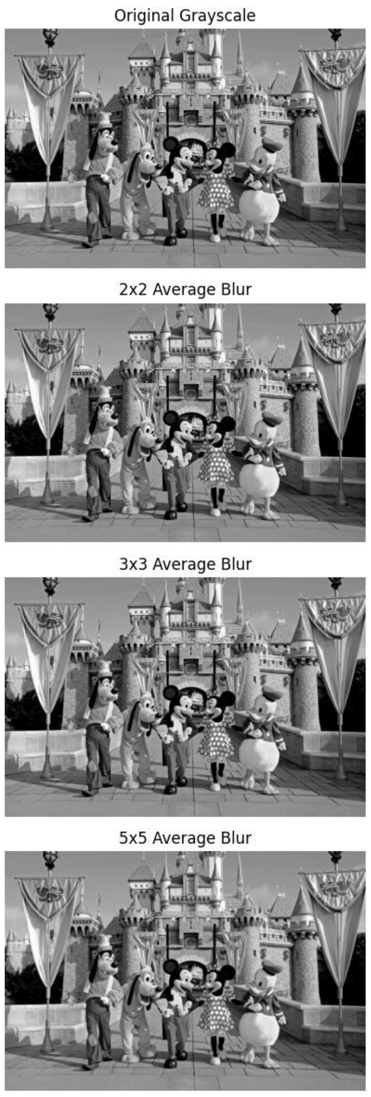


- **Diagonal Edge Detection (45°):**



Blurring

- **2x2 Average Blur**
- **3x3 Average Blur**
- **5x5 Average Blur**



Corner Detection

- **Harris Corner Detection:**

3. Part 2 – Convolutional Neural Network (CNN) Implementation

Objectives

- To design and implement a CNN model using PyTorch for image classification on the CIFAR-10 dataset.
- To evaluate the performance of the CNN using training, validation, and test data.
- To compare different optimizers, particularly SGD and Adam.

CNN Architecture

The model consists of the following layers:

1. **Convolutional Layers:**
    - conv1: 32 filters, 3x3 kernel.
    - conv2: 64 filters, 3x3 kernel.
    - conv3: 128 filters, 3x3 kernel.
2. **Max Pooling Layers:**
    - Applied after each convolutional layer to reduce the spatial dimensions of the image, making the model more computationally efficient.
3. **Fully Connected Layers:**
    - fc1: Fully connected layer with 512 neurons.
    - fc2: Output layer with 10 neurons, corresponding to the 10 classes of CIFAR-10.
4. **Dropout Layer:**
    - Dropout with a probability of 0.25 was applied to reduce overfitting.

Optimizers

Two optimizers were tested:

1. **SGD (Stochastic Gradient Descent)**: Learning rate of 0.01 with momentum.
2. **Adam**: Learning rate of 0.001 (generally performs better).

Training and Validation

The model was trained over 5 epochs using the training data. The validation loss was monitored to prevent overfitting, and the model with the lowest validation loss was saved.

Test Results

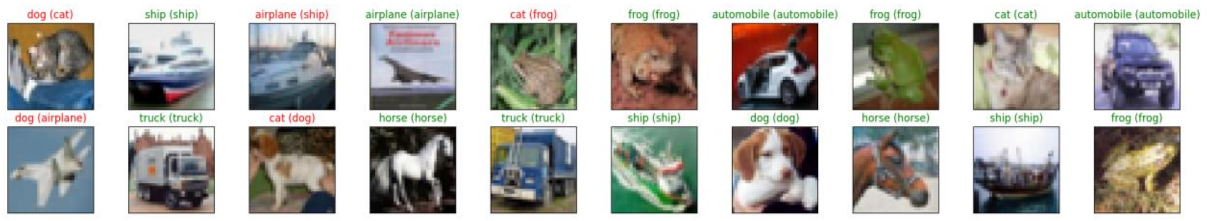
After training, the model was evaluated on the test data, achieving the following results:

- **Test Accuracy (Overall):** 73.64%
- **Test Accuracy per Class:**
  - Airplane: 79.7%
  - Automobile: 81.4%
  - Bird: 59.3%
  - Cat: 55.9%
  - Deer: 69.4%
  - Dog: 68.0%
  - Frog: 83.9%
  - Horse: 78.0%
  - Ship: 83.0%
  - Truck: 77.8%

Visualizations

Sample Test Results

- **Test Images:** The following visualizations show the predicted and true labels for 20 test images. Correct predictions are shown in green, while incorrect predictions are shown in red.



4. Analysis and Findings

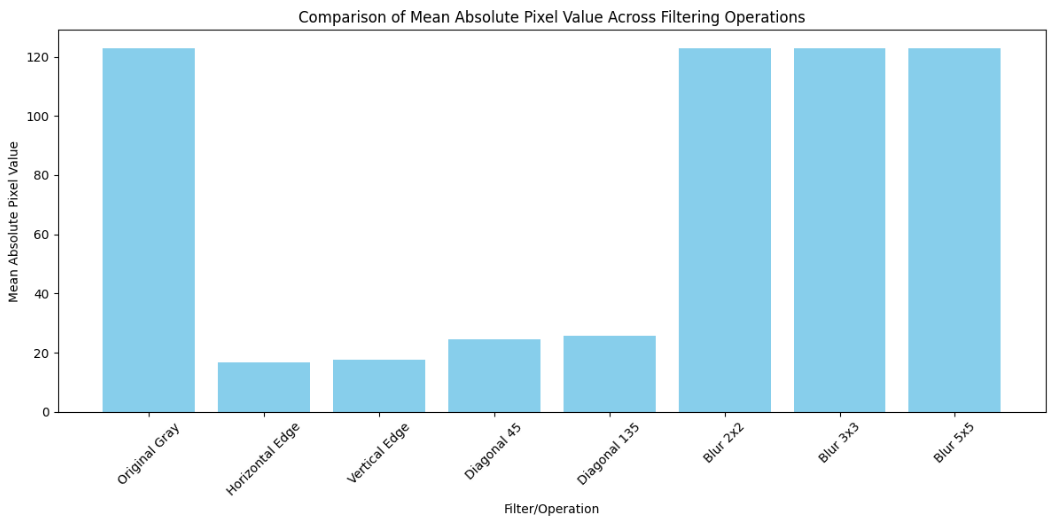
Image Processing

- **Edge Detection:**
  The custom-designed kernels effectively highlighted the edges in the image, which is useful for tasks like object detection and recognition.
- **Blurring:**
  The blurring filters worked well for noise reduction, though excessive blurring (e.g., using a 5x5 kernel) resulted in the loss of finer details.
- **Corner Detection:**
  The Harris corner detection technique successfully identified corners, useful for key point detection and image matching.

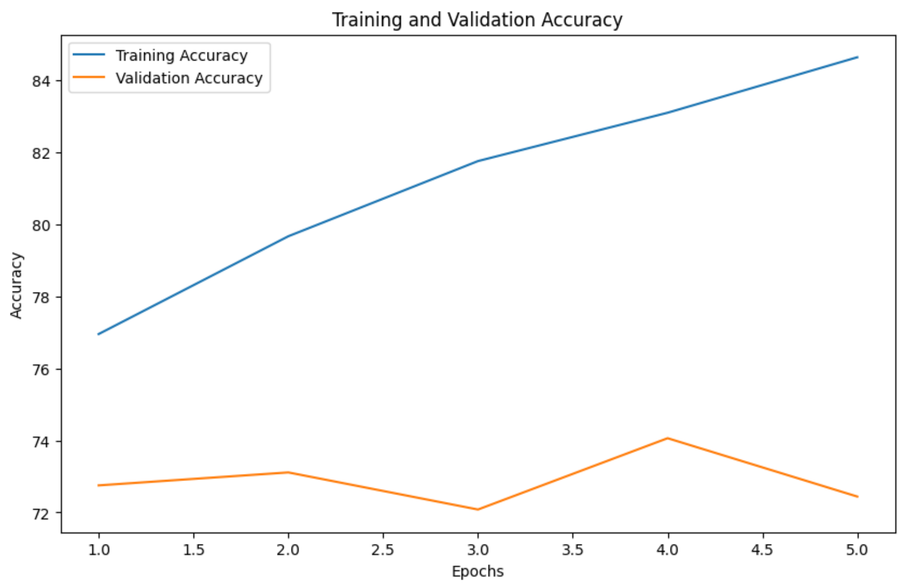
CNN Model

- **Performance:**
  The CNN achieved an overall accuracy of 73.64% on the test dataset. This result is quite good for the CIFAR-10 dataset, considering its complexity.
- **Optimizer Comparison:**
  The Adam optimizer performed slightly better than SGD in terms of training speed and final accuracy. It converged faster and provided more stable results.
- **Overfitting:**
  The dropout layer helped mitigate overfitting, as seen by the decrease in validation loss over the epochs.

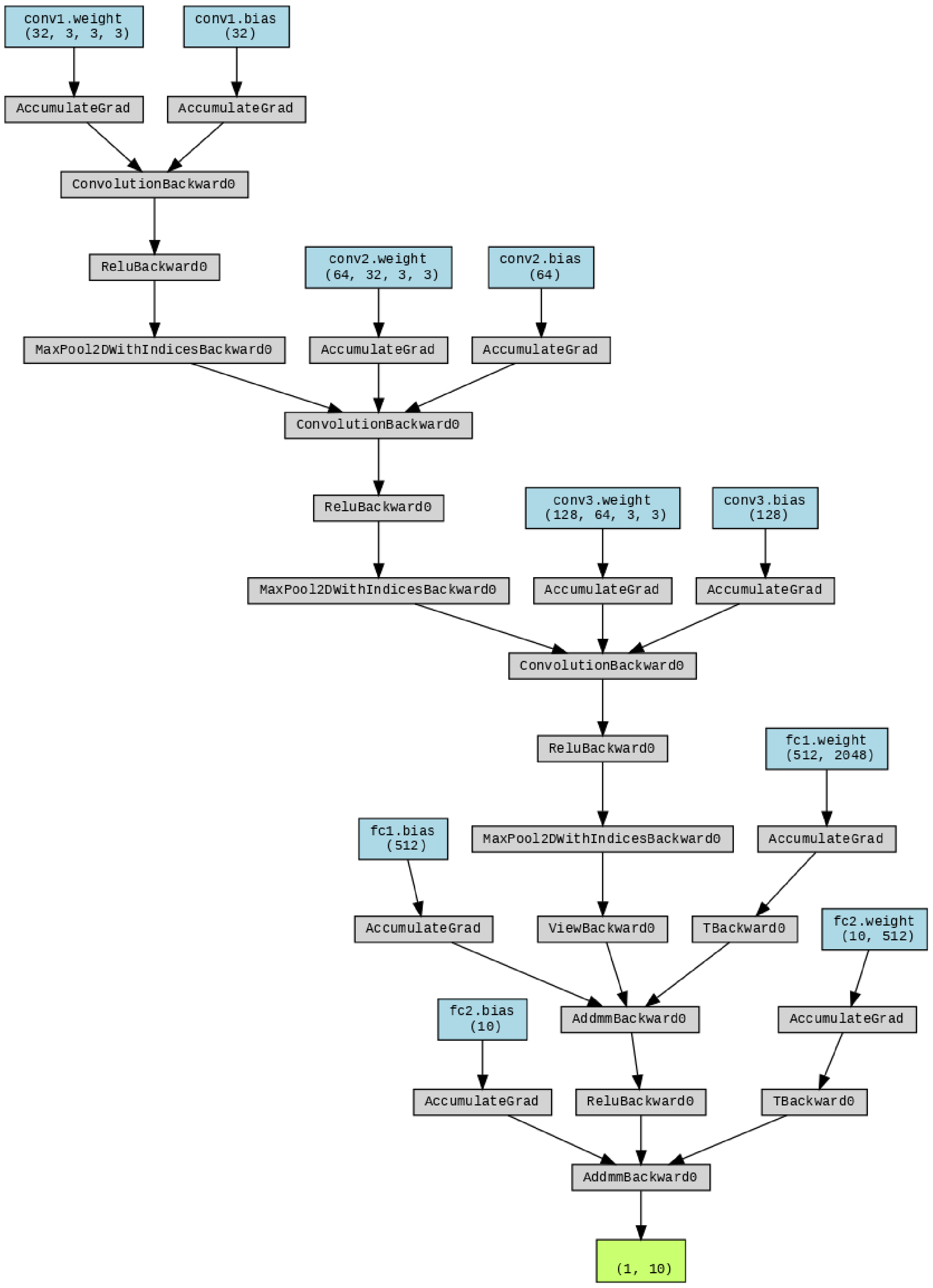
5. **Screenshots of how the code is successfully executed and generating required outputs, graphs and tables**
    a. Image filtering: Comparison of Mean Absolute Pixel Value Across Filtering Operations graph



    b. CNN: Training and Validation Accuracy graph



CNN architecture graphical illustration:


conv1.weight (32, 3, 3, 3)
conv1.bias (32)
AccumulateGrad
AccumulateGrad
ConvolutionBackward0
ReluBackward0
conv2.weight (64, 32, 3, 3)
conv2.bias (64)
MaxPool2DWithIndicesBackward0
AccumulateGrad
AccumulateGrad
ConvolutionBackward0
ReluBackward0
conv3.weight (128, 64, 3, 3)
conv3.bias (128)
MaxPool2DWithIndicesBackward0
AccumulateGrad
AccumulateGrad
ConvolutionBackward0
ReluBackward0
fc1.weight (512, 2048)
fc1.bias (512)
MaxPool2DWithIndicesBackward0
AccumulateGrad
AccumulateGrad
ViewBackward0
TBackward0
fc2.weight (10, 512)
fc2.bias (10)
AddmmBackward0
AccumulateGrad
AccumulateGrad
ReluBackward0
TBackward0
AddmmBackward0
(1, 10)

6. Conclusion

This project demonstrated the power of convolutional neural networks in image classification tasks. The image processing techniques (such as edge detection and blurring) were instrumental in extracting essential features from the images, making the training process more effective. By examining the bar chart of mean absolute pixel values, we can see how each filter operation whether edge detection or blurring uniquely alters the distribution of pixel intensities, providing insights into how these transformations might be used to highlight or suppress different image features.

Furthermore, the line chart of Training vs. Validation Accuracy illustrates that while training accuracy steadily improved over five epochs (from ~77% to ~85%), validation accuracy fluctuated in the low-to-mid 70% range. This discrepancy indicates a degree of overfitting and suggests that additional regularization strategies, data augmentation, or more sophisticated architectures could help close the gap between training and validation performance. Overall, the CNN architecture achieved a decent performance on the CIFAR-10 dataset, and further improvements could be made by experimenting with deeper models or more advanced techniques like transfer learning.