**Image processing and recognition**

**Image Processing Using OpenCV and Matplotlib**

**Objective:** The objective of this section is to demonstrate basic image processing techniques using OpenCV and visualize the results using Matplotlib.

**Methods:**

1. **Loading and Displaying the Image:**
   * Load an image from the file system using cv2.imread.
   * Convert the image from BGR to RGB format (since OpenCV loads images in BGR format and Matplotlib expects RGB).
   * Display the RGB image using Matplotlib.
2. **Converting to Grayscale:**
   * Convert the RGB image to a grayscale image using cv2.cvtColor with the cv2.COLOR\_BGR2GRAY flag.
   * Display the grayscale image using Matplotlib.
3. **Applying Gaussian Blur:**
   * Apply Gaussian blur to the original image using cv2.GaussianBlur to smooth the image.
   * Convert the blurred image from BGR to RGB format.
   * Display the blurred image using Matplotlib.
4. **Edge Detection:**
   * Perform Canny edge detection on the original image using cv2.Canny.
   * Display the edges detected using Matplotlib.

**Image Classification Using a Convolutional Neural Network (CNN)**

**Objective:** The objective of this section is to build, train, and evaluate a Convolutional Neural Network (CNN) for image classification using the CIFAR-10 dataset.

**Methods:**

1. **Loading and Preprocessing the Data:**
   * Load the CIFAR-10 dataset using tf.keras.datasets.cifar10.
   * Normalize the images by scaling the pixel values to the range [0, 1].
2. **Visualizing the Data:**
   * Display the first 16 images of the training set along with their corresponding labels using Matplotlib.
3. **Building the CNN Model:**
   * Build a sequential CNN model using Keras with the following layers:
     + Convolutional layer with 32 filters, 3x3 kernel size, and ReLU activation.
     + Max pooling layer with 2x2 pool size.
     + Convolutional layer with 62 filters, 3x3 kernel size, and ReLU activation.
     + Max pooling layer with 2x2 pool size.
     + Convolutional layer with 62 filters, 3x3 kernel size, and ReLU activation.
     + Flatten layer to convert 2D feature maps to 1D feature vectors.
     + Dense layer with 64 units and ReLU activation.
     + Output dense layer with 10 units (one for each class) and softmax activation.
4. **Compiling and Training the Model:**
   * Compile the model with Adam optimizer, sparse categorical cross-entropy loss, and accuracy as the metric.
   * Train the model for 10 epochs with training and validation data.
5. **Evaluating the Model:**
   * Load a pre-trained model.
   * Load a new image, preprocess it, and make a prediction using the trained model.
   * Display the image along with the predicted class.

### Results

Loss:1.0301839113235474

Accuracy:0.6522499918937683

### The prediction is not always correct. There are cases where cat is identified as birds, truck is identified as plane and frog is identified as birds.

### We can improve the accuracy by training more models.

### Conclusion

In this report, we demonstrated basic image processing techniques using OpenCV, including loading, displaying, converting to grayscale, applying Gaussian blur, and performing edge detection. Additionally, we built, trained, and evaluated a CNN for image classification using the CIFAR-10 dataset. The trained model was successfully used to classify a new image, predicting it as a specific class from the dataset. The results showcase the effectiveness of convolutional neural networks for image classification tasks.