Waste and Recycling Classification DIP Project Report

Fahad Kamran 20i-0983

December 3, 2023

1 Introduction

The waste classification project aims to automatically categorize waste items into organic and recyclable classes using machine learning techniques. Proper waste classification is crucial for effective recycling and waste management.

2 Data Collection and Preprocessing

The training and testing datasets were collected from Kaggle and saved the specified directories: "D:/DATASET/DATASET/TRAIN" for training and "D:/DATASET/DATASET/TEST" for testing. Images were read using OpenCV, converted to grayscale, and their histograms were equalized to enhance features.

3 Step by Step Method Guide

3.1 Introduction

The following document provides a step-by-step guide to a waste classification project. This project involves loading and preprocessing image data, splitting it into training and validation sets, and training a Support Vector Machine (SVM) classifier for waste classification.

3.2 Data Collection and Preprocessing

The training dataset is collected and preprocessed using the following steps:

1. Iterate through subdirectories ('O' and 'R').
2. Read each image using OpenCV.
3. Convert the image to grayscale.
4. Equalize the histogram of the grayscale image.

3.3 Splitting Data and Displaying Images

After successfully collecting and preprocessing the training data, the next crucial step is to split it into training and validation sets. This process is essential for assessing the model's performance and preventing overfitting. The `train_test_split` function from the `sklearn.model_selection` module is employed for this purpose.

The data is divided into training (`X_train` and `y_train`) and validation (`X_val` and `y_val`) sets. The split is performed with a test size of 20% and a random seed of 42 for reproducibility.

To gain insight into the training images, a random subset is selected and displayed. This provides a visual representation of the dataset, helping to verify that the images are correctly labeled and possess the desired characteristics.

Similarly, a subset of random validation images is displayed to ensure that the validation set is representative of the overall dataset. This step aids in verifying that the model is trained on diverse data, enhancing its ability to generalize to new, unseen images.

3.4 Testing Data Collection and Preprocessing

The testing dataset is collected from Kaggle Waste Classification Dataset and preprocessed following the same steps as in the training dataset:

1. Iterate through subdirectories ('O' and 'R').
2. Read each image using OpenCV.
3. Convert the image to grayscale.
4. Equalize the histogram of the grayscale image.

3.5 Displaying Random Testing Images

The effectiveness of the waste classification model relies not only on the training and validation sets but also on its ability to accurately classify new, unseen images. To assess the model's performance on the testing set, a subset of random testing images is selected and displayed.

Similar to the training and validation sets, the testing dataset is collected, preprocessed, and then a random subset is chosen for visualization. The displayed images, along with their corresponding labels, provide a qualitative assessment of the model's ability to generalize to new data.

3.6 HOG Feature Extraction and SVM Training

To enhance the model's ability to distinguish between different waste categories, Histogram of Oriented Gradients (HOG) feature extraction is employed. HOG is a widely-used technique for object detection and classification, particularly in image processing.

The process involves extracting features from the preprocessed images to capture information about the distribution of gradients and edges. These features are then fed into a Support Vector Machine (SVM) classifier for training. SVM is a robust and effective machine learning algorithm suitable for binary classification tasks, making it well-suited for waste classification.

HOG feature extraction is performed on the training, validation, and testing sets separately. This ensures that the SVM model is trained and evaluated on data with consistent feature representations. The features obtained from the HOG extraction process serve as inputs to the SVM classifier, allowing it to learn the patterns and characteristics necessary for accurate waste classification.

The integration of HOG feature extraction with SVM training enhances the model's discriminative capabilities, enabling it to make informed decisions based on the extracted features. This combination forms a powerful framework for waste classification, ultimately contributing to the model's accuracy and robustness.

3.7 Model Evaluation and Metrics

Once the SVM model is trained on the extracted HOG features, it is crucial to evaluate its performance on both the validation and testing sets. This section focuses on assessing the model's classification accuracy and effectiveness in making predictions.

The evaluation metrics employed include:

- **Confusion Matrix:** A table that summarizes the model's performance by comparing predicted and actual class labels. It provides insights into the number of true positives, true negatives, false positives, and false negatives.

- **Accuracy:** The proportion of correctly classified instances among the total instances. It serves as a general indicator of the model's correctness.

- **Precision:** The ratio of true positives to the sum of true positives and false positives. Precision measures the accuracy of positive predictions.

- **Recall (Sensitivity):** The ratio of true positives to the sum of true positives and false negatives. Recall quantifies the model's ability to capture all positive instances.

- **F1 Score:** The harmonic mean of precision and recall, providing a balanced measure that considers both false positives and false negatives.

By systematically evaluating the model using these metrics, we gain a comprehensive understanding of its strengths and weaknesses. The confusion matrix visualizes the distribution of predictions, while accuracy, precision, recall, and F1 score offer nuanced insights into specific aspects of the model's performance.

3.8 Displaying Confusion Matrix

To gain a visual understanding of the model's classification performance, a confusion matrix is employed. The confusion matrix is a table that summarizes the model's predictions against the actual class labels for both the validation and testing sets.

The confusion matrix visually presents the following key metrics:
- **True Positives (TP):** The instances correctly predicted as positive.
- **True Negatives (TN):** The instances correctly predicted as negative.
- **False Positives (FP):** The instances incorrectly predicted as positive.
- **False Negatives (FN):** The instances incorrectly predicted as negative.

These metrics are crucial for assessing the model's performance across different classes and understanding potential misclassifications. The confusion matrix is particularly effective in identifying specific areas where the model may need improvement.

By displaying the confusion matrix, one can easily interpret the distribution of predictions and misclassifications. This visual representation enhances the model evaluation process, providing valuable insights into its strengths and weaknesses.

3.9 Model Saving

After successfully training the SVM model and evaluating its performance, the final step is to save the trained model for future use. Model saving is a critical aspect of the machine learning workflow as it preserves the trained weights, parameters, and architecture, allowing for easy deployment and reuse.

In this section, the trained SVM model is saved using the `pickle` library, a widely-used tool for serializing Python objects. The model is serialized into a binary file, preserving its state and ensuring that it can be loaded and utilized later without the need for retraining.

By saving the model, one can avoid the time-consuming process of training the model every time it needs to be deployed or applied to new data. Additionally, model saving facilitates model sharing and collaboration, enabling others to utilize the trained model for similar tasks.

It is crucial to choose an appropriate file name and location for saving the model. This ensures that the saved model can be easily located and loaded when needed.

This section serves as the final step in the waste classification pipeline, ensuring that the knowledge acquired during training is stored in a reusable and deployable format.

4 Algorithms

4.1 Data Loading and Processing

The following pseudocode outlines the process of loading and processing the image data:

Listing 1: Data Loading and Processing Pseudocode

```
# Iterate through subdirectories (O and R)
for folder_name in os.listdir(train_dataset_dir):
     folder_path = os.path.join(train_dataset_dir, folder_name)

     if os.path.isdir(folder_path):  # Ensure it's a directory
         for file_name in os.listdir(folder_path):
             file_path = os.path.join(folder_path, file_name)

             # Check if the file exists
             if os.path.isfile(file_path):
                 # Read the image using OpenCV
                 image = cv2.imread(file_path)

                 if image is not None:
                     # Convert the image to grayscale
                     grayscale_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

                     # Equalize the histogram
                     enhanced_image = cv2.equalizeHist(grayscale_image)

                     train_images.append(enhanced_image)
                     train_labels.append(extract_label(folder_name))
                 else:
                     print(f"Failed to read image: {file_path}")
             else:
                 print(f"File not found: {file_path}")
     else:
         print(f"Not a directory: {folder_path}")
```

4.2 HOG Feature Extraction and SVM Training

The following pseudocode outlines the process of extracting HOG features and training an SVM classifier:

Listing 2: HOG Feature Extraction and SVM Training Pseudocode

```
# Extract HOG features
def extract_hog_features(images, target_size=(128, 128)):
    hog_features = []
    for image in images:
        # Resize the image to a fixed size
        resized_image = resize(image, target_size, anti_aliasing=True)
        # Compute HOG features directly for grayscale images
        features = hog(resized_image, orientations=8, pixels_per_cell=(8, 8), ce
        hog_features.append(features)
    return np.vstack(hog_features)

X_train_hog = extract_hog_features(X_train)
```

```
X_val_hog = extract_hog_features(X_val)
X_test_hog = extract_hog_features(X_test)

# Train an SVM classifier
svm_model = SVC(kernel='linear')
svm_model.fit(X_train_hog, y_train)
```

5 Results

5.1 Visualizations

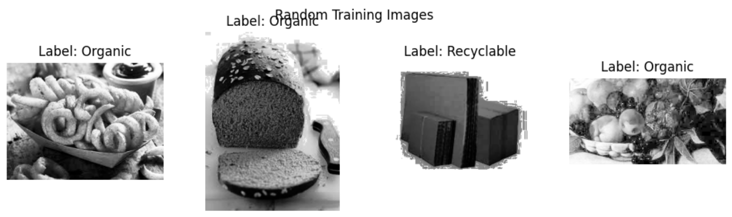


Figure 1: Random Training Images

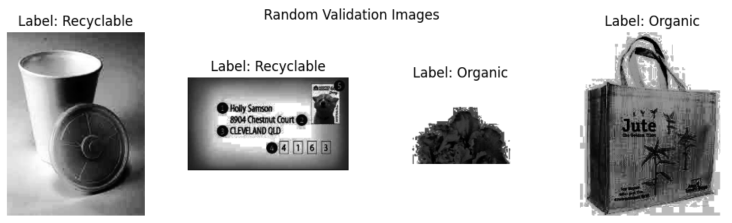


Figure 2: Random Validation Images

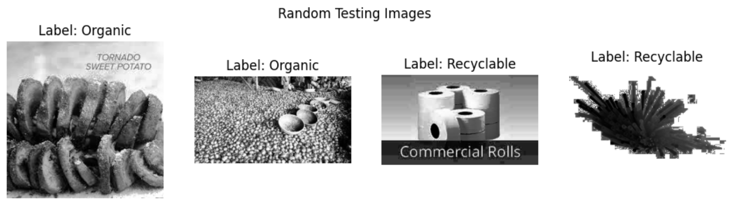


Figure 3: Random Testing Images

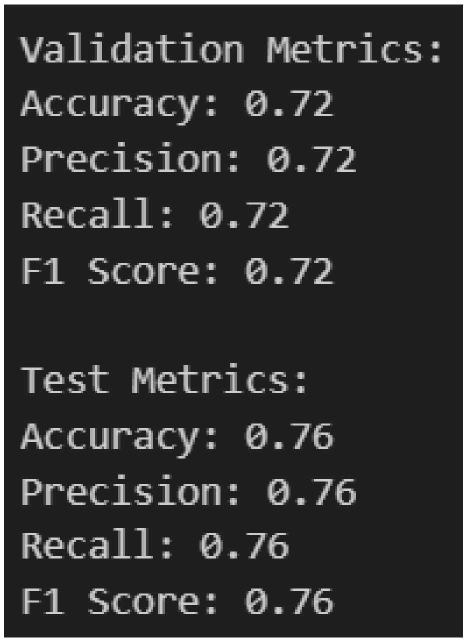
5.2 Tables

Total Samples (Classes)	2 (Organic and Recyclable)
Total Images and Labels	25076, 25076
Training Samples	18050, 18050
Validation Samples	4513, 4513
Testing Samples	2513, 2513

Table 1: Dataset Summary

5.3 Metrics and Analysis

The model is evaluated on the validation and testing sets using metrics such as confusion matrix, accuracy, precision, recall, and F1 score on the SVM Classifier based on HOG feature extraction.



```
Validation Metrics:
Accuracy: 0.72
Precision: 0.72
Recall: 0.72
F1 Score: 0.72

Test Metrics:
Accuracy: 0.76
Precision: 0.76
Recall: 0.76
F1 Score: 0.76
```

Figure 4: Metrics

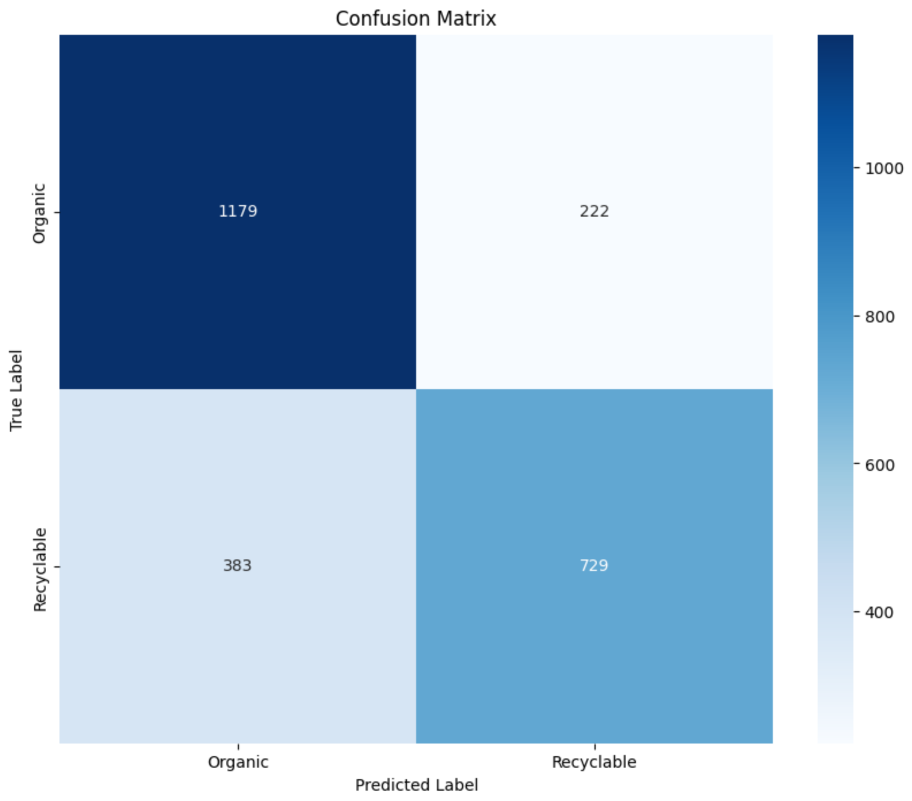
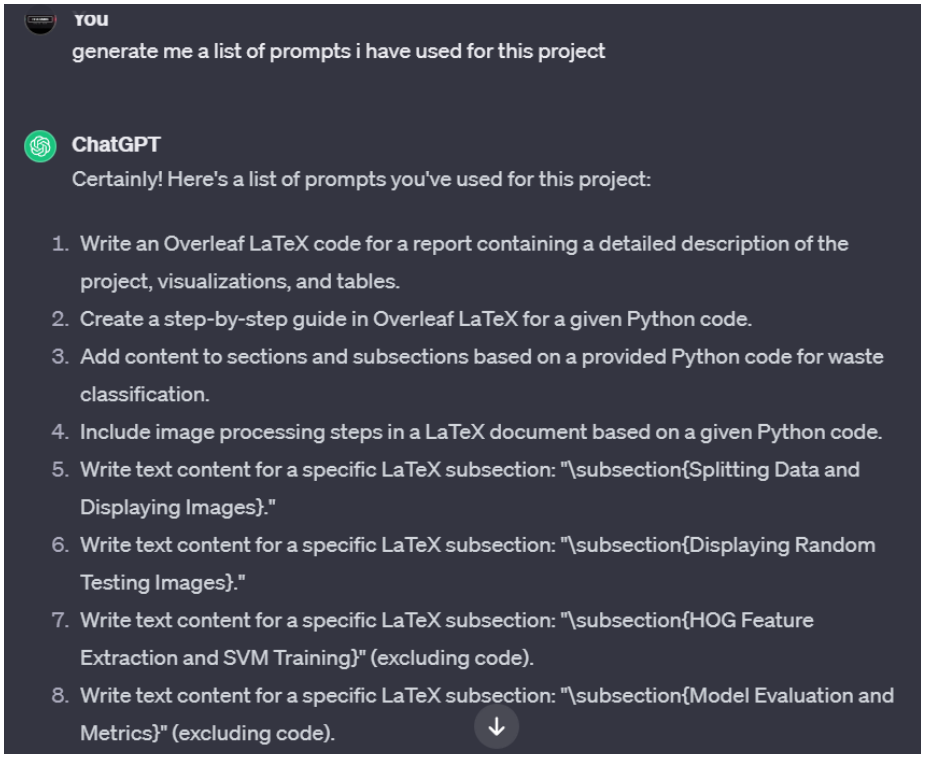


Figure 5: Confusion Matrix

6 Conclusion

The waste classification project demonstrates promising results in distinguishing between organic and recyclable waste. The preprocessing steps, feature extraction, and SVM training contribute to the overall success of the model. In Comparison with a basic CNN structure, my model had a good accuracy. The edge CNN has over SVM classifier is its auto feature extraction and kernel adaptation learning in its training which result in CNN model reaching accuracy of classification greater that 90 percent on the same dataset.

7 CHATGPT Prompts Used

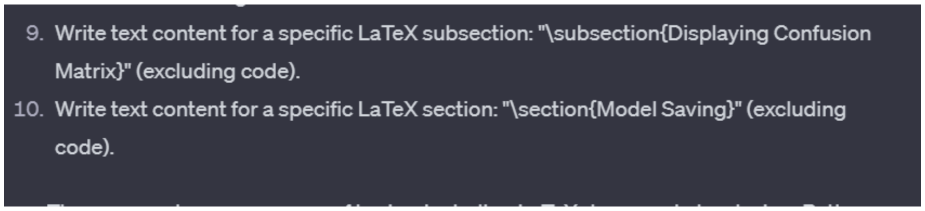


You
generate me a list of prompts i have used for this project

ChatGPT
Certainly! Here's a list of prompts you've used for this project:

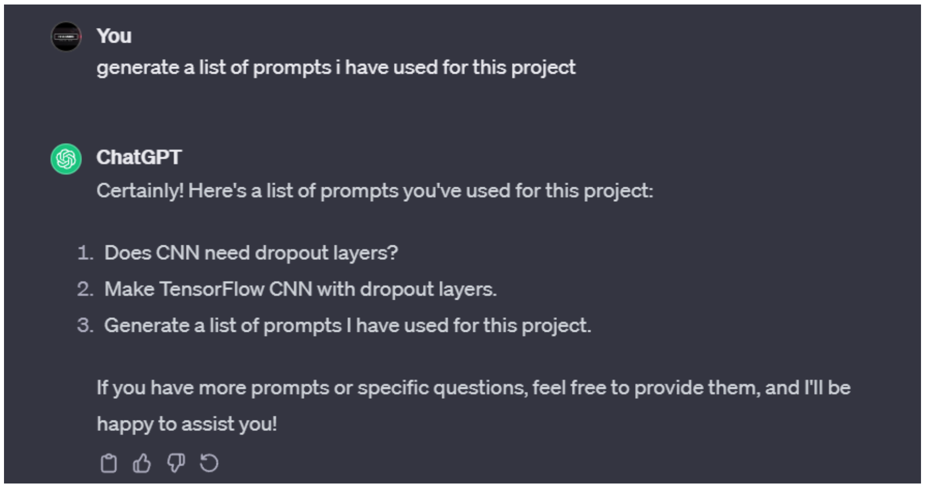
1. Write an Overleaf LaTeX code for a report containing a detailed description of the project, visualizations, and tables.
2. Create a step-by-step guide in Overleaf LaTeX for a given Python code.
3. Add content to sections and subsections based on a provided Python code for waste classification.
4. Include image processing steps in a LaTeX document based on a given Python code.
5. Write text content for a specific LaTeX subsection: "\subsection{Splitting Data and Displaying Images}."
6. Write text content for a specific LaTeX subsection: "\subsection{Displaying Random Testing Images}."
7. Write text content for a specific LaTeX subsection: "\subsection{HOG Feature Extraction and SVM Training}" (excluding code).
8. Write text content for a specific LaTeX subsection: "\subsection{Model Evaluation and Metrics}" (excluding code).

Figure 6



9. Write text content for a specific LaTeX subsection: "\subsection{Displaying Confusion Matrix}" (excluding code).
10. Write text content for a specific LaTeX section: "\section{Model Saving}" (excluding code).

Figure 7

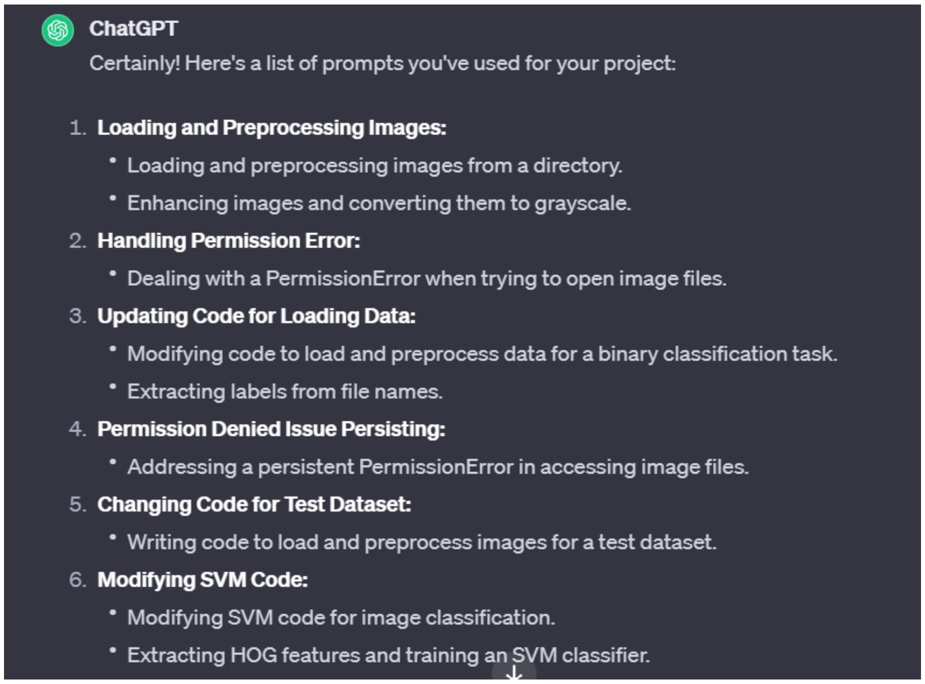

You
generate a list of prompts i have used for this project

ChatGPT
Certainly! Here's a list of prompts you've used for this project:

1. Does CNN need dropout layers?
2. Make TensorFlow CNN with dropout layers.
3. Generate a list of prompts I have used for this project.

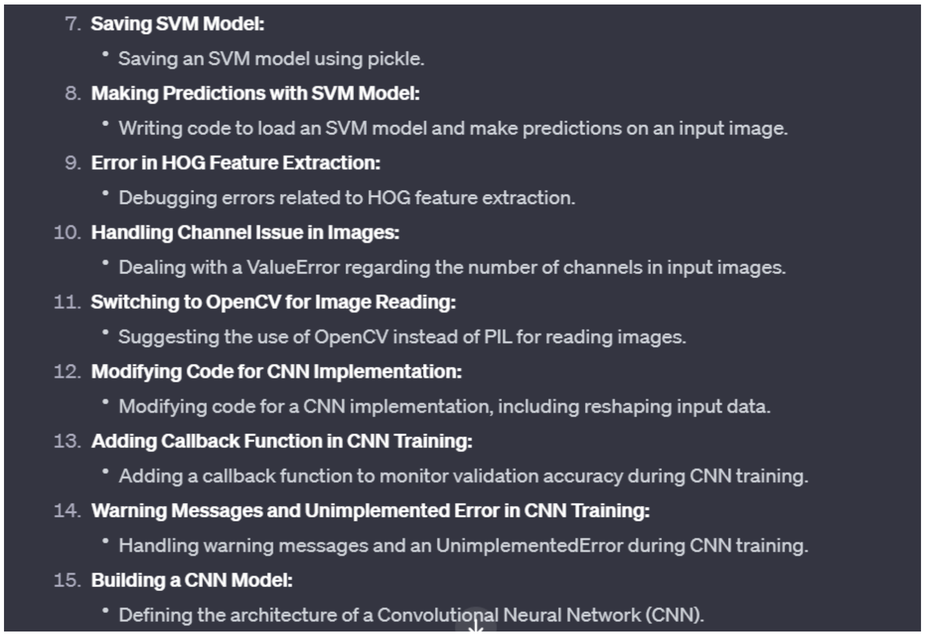
If you have more prompts or specific questions, feel free to provide them, and I'll be happy to assist you!

Figure 8


ChatGPT
Certainly! Here's a list of prompts you've used for your project:

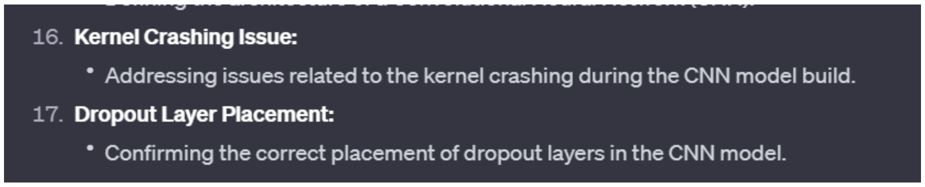
1. **Loading and Preprocessing Images:**
   - Loading and preprocessing images from a directory.
   - Enhancing images and converting them to grayscale.
2. **Handling Permission Error:**
   - Dealing with a PermissionError when trying to open image files.
3. **Updating Code for Loading Data:**
   - Modifying code to load and preprocess data for a binary classification task.
   - Extracting labels from file names.
4. **Permission Denied Issue Persisting:**
   - Addressing a persistent PermissionError in accessing image files.
5. **Changing Code for Test Dataset:**
   - Writing code to load and preprocess images for a test dataset.
6. **Modifying SVM Code:**
   - Modifying SVM code for image classification.
   - Extracting HOG features and training an SVM classifier.

Figure 9



7. **Saving SVM Model:**
   - Saving an SVM model using pickle.
8. **Making Predictions with SVM Model:**
   - Writing code to load an SVM model and make predictions on an input image.
9. **Error in HOG Feature Extraction:**
   - Debugging errors related to HOG feature extraction.
10. **Handling Channel Issue in Images:**
    - Dealing with a ValueError regarding the number of channels in input images.
11. **Switching to OpenCV for Image Reading:**
    - Suggesting the use of OpenCV instead of PIL for reading images.
12. **Modifying Code for CNN Implementation:**
    - Modifying code for a CNN implementation, including reshaping input data.
13. **Adding Callback Function in CNN Training:**
    - Adding a callback function to monitor validation accuracy during CNN training.
14. **Warning Messages and Unimplemented Error in CNN Training:**
    - Handling warning messages and an UnimplementedError during CNN training.
15. **Building a CNN Model:**
    - Defining the architecture of a Convolutional Neural Network (CNN).

Figure 10



16. **Kernel Crashing Issue:**
    - Addressing issues related to the kernel crashing during the CNN model build.
17. **Dropout Layer Placement:**
    - Confirming the correct placement of dropout layers in the CNN model.

Figure 11

8 References

- Kaggle Dataset: Waste Classification Data
  `https://www.kaggle.com/datasets/techsash/waste-classification-data/code`