This project focuses on the classification of mushroom images into nine distinct classes: Agaricus, Amanita, Boletus, Cortinarius, Entoloma, Hygrocybe, Lactarius, Russula, and Suillus. The training images are organized in a folder-based structure under the `Mushrooms/` directory, with each subfolder representing a different class. For testing purposes, a separate directory `sample_test_data/` contains a CSV file (`mushrooms_test.csv`) listing test image filenames and their corresponding true labels, along with a folder of the actual test images.

Two models were implemented and evaluated. The first model is a convolutional neural network (CNN) with a softmax classifier, trained end-to-end to predict mushroom classes directly from image data. The architecture consists of three convolutional layers with ReLU activations, each followed by max pooling, then a flattening layer, dropout for regularization, a dense layer with 128 units, and a final dense layer with softmax activation for classification into 9 categories. The model was trained using data augmentation techniques such as rotation, shift, zoom, and flipping, with images resized to 128x128 pixels. The training was done over 15 epochs using the Adam optimizer with a learning rate of 1e-4, and the data was split into 80% training and 20% validation. The model achieved 98–99% training accuracy and 95–98% validation accuracy. The trained model was saved as `softmax_model.keras` and the class index mapping as `class_indices.pkl`.

For testing, the CNN model was evaluated using the provided test CSV. Each test image was preprocessed, resized, and fed into the trained softmax model. The predicted class indices were mapped back to class names using `class_indices.pkl`, and predictions were compared against the true labels to calculate test accuracy. The output included per-image predictions and a final overall accuracy score.

The second model took a hybrid approach, combining a CNN-based feature extractor with a support vector machine (SVM) classifier. In this pipeline, the CNN was either randomly initialized or trained briefly to produce meaningful features. The output from the final dense layer of the CNN (prior to the softmax layer) was used as a feature vector. These feature vectors were then used to train an `sklearn.svm.SVC` model with a linear kernel. When features were extracted from an untrained CNN, the SVM achieved only about 30% accuracy. However, after training the CNN and re-extracting features, SVM accuracy significantly improved, reaching 70–90% depending on the quality of the CNN features. The SVM model was saved as `svm_model.pkl`, and its class mapping as `class_indices_svm.pkl`.

Testing for the SVM pipeline followed a similar structure. Each test image was preprocessed and passed through the trained CNN to extract a feature vector. The SVM then predicted the class label, which was mapped back to a readable label using `class_indices_svm.pkl`. Predictions were compared to ground truth labels from the CSV, and overall accuracy was reported.

The outputs from both models included saved model files, class index mappings, and logs of prediction accuracy. Key artifacts produced include `softmax_model.keras`, `class_indices.pkl`, `feature_extractor_model.keras`, `svm_model.pkl`, and `class_indices_svm.pkl`.

Future improvements for this project include using pretrained CNNs such as MobileNetV2 for better feature extraction, augmenting the dataset with more images and class variations, employing cross-validation to assess model stability, and tuning hyperparameters using grid search techniques.

In conclusion, this project successfully built two classification systems: a CNN with softmax classifier and a CNN-SVM hybrid pipeline. Both models were tested using real-world image data, showing the effectiveness of end-to-end learning as well as modular feature-based classification approaches for multi-class image recognition tasks.