

Data Collection and Preprocessing Phase

Date	28 January 2026
Student Name	Sammed Sunil Awati
Project Title	Uncovering the Hidden Treasures of the Mushroom Kingdom A Classification Analysis
Maximum Marks	6 Marks

Data Preprocessing

The images will be preprocessed by resizing, normalizing, augmenting, denoising, adjusting contrast, detecting edges, converting color space, cropping, batch normalizing, and whitening data. These steps will enhance data quality, promote model generalization, and improve convergence during neural network training, ensuring robust and efficient performance across various computer vision tasks.

Section	Description

Data Overview	This project uses image datasets of three mushroom species: Boletus , Lactarius , and Russula . The images are collected from various sources including SmartInternz , custom fieldcaptured images , and platforms like Kaggle and Wikimedia . This ensures rich visual diversity and robust generalization during training.
Resizing	All images are resized to 224×224 pixels using OpenCV's cv2.resize() function to ensure uniform input dimensions for CNN-based models.
Normalization	Pixel values are normalized to the range [0, 1] by dividing by 255.0, improving convergence during model training.
Data Augmentation	Using ImageDataGenerator, images are augmented with random rotation , shifts , zoom , horizontal/vertical flips , and fill modes to avoid overfitting.
Denoising	OpenCV's fastNlMeansDenoisingColored() is applied to reduce environmental noise and improve image clarity, especially for field-captured data.

Edge Detection	cv2.Canny() is used for edge detection, helping to emphasize structure, texture, and contour features of different mushroom species.
Color Space Conversion	Images are converted from BGR to HSV color space using cv2.cvtColor() to better capture color-based patterns across lighting variations.
Image Cropping	Manual center cropping is done on some images to focus on the mushroom body and reduce irrelevant background noise, enhancing object recognition.

Batch Normalization

BatchNormalization() is applied in the neural network model to stabilize and accelerate the learning process by reducing internal covariate shift.

Data Preprocessing Code Screenshots

```

1  import os
2  import cv2
3  import numpy as np
4  from tensorflow.keras.preprocessing.image import ImageDataGenerator
5
6  data_dir = "path/to/mushroom_dataset"
7  categories = ["Boletus", "Lactarius", "Russula"]
8
9  data = []
10 IMG_SIZE = 224
11
12 for category in categories:
13     folder = os.path.join(data_dir, category)
14     label = categories.index(category)
15     for img_name in os.listdir(folder):
16         img_path = os.path.join(folder, img_name)
17         img = cv2.imread(img_path)
18         if img is not None:
19             img = cv2.resize(img, (IMG_SIZE, IMG_SIZE))
20             data.append([img, label])
21

```

Loading Data

Resizing

```
img = cv2.resize(img, (224, 224))
```

	<pre> data = np.array(data, dtype=object) X = np.array([i[0] for i in data]) / 255.0 # Normalize pixel values y = np.array([i[1] for i in data]) </pre>
Normalization	
Data Augmentation	<pre> train_datagen = ImageDataGenerator(rotation_range=30, width_shift_range=0.1, height_shift_range=0.1, zoom_range=0.2, horizontal_flip=True, vertical_flip=True, fill_mode='nearest') </pre>

Color Space Conversion	<pre>hsv_img = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)</pre>
Image Cropping	<pre>cropped_img = img[30:194, 30:194] # Manual center crop</pre>
Batch Normalization	<pre>from tensorflow.keras.layers import BatchNormalization model.add(BatchNormalization())</pre>
Denoising	<pre>denoised_img = cv2.fastNlMeansDenoisingColored(img, None, 10, 10, 7, 21)</pre>
Edge Detection	<pre>edges = cv2.Canny(img, threshold1=100, threshold2=200)</pre>