Data Collection and Preprocessing Phase

Date	20 July 2024
Team ID	SWTID1720163161
Project Title	Hydration Essentials: Classifying Water Bottle Images
Maximum Marks	6 Marks

Preprocessing Template

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	The dataset consists of images of water bottles at various water levels. Each category represents a different water level (e.g., overflow, half, full). The dataset is organized into directories named after these categories.
Resizing	Images are resized to 224x224 pixels to be compatible with the chosen model architecture .
Normalization	Normalization is performed by dividing the pixel values by 255, bringing them to the range [0, 1].
Data Augmentation	Data augmentation techniques such as flipping, rotation, and zooming can be applied using the ImageDataGenerator from Keras to enhance the dataset's diversity.
Image Cropping	Images can be cropped to focus on the water levels in the bottles, ensuring that the model learns relevant features.
Batch Normalization	Batch normalization is applied to the inputs of each layer in the neural network to stabilize and accelerate training.

Data Preprocessing Code Screenshots	
Loading Data	[3]: `dataset = tf.keras.preprocessing.image_dataset_from_directory("archivek", seed=123, shuffle=True,# image_size=(IMAGE_SIZE,IMAGE_SIZE), batch_size=BATCH_SIZE )`
Resizing	[12]: `resize_and_rescale = tf.keras.Sequential([ layers.Resizing(IMAGE_SIZE, IMAGE_SIZE), layers.Rescaling(1./255), ])`
Normalization	[12]: `resize_and_rescale = tf.keras.Sequential([ layers.Resizing(IMAGE_SIZE, IMAGE_SIZE), layers.Rescaling(1./255), ])`
Data Augmentation	Data Augmentation [14]: `data_augmentation = tf.keras.Sequential([ layers.RandomFlip("horizontal_and_vertical"), layers.RandomRotation(0.2), ])` Applying Data Augmentation to Train Dataset [16]: `train_ds = train_ds.map( lambda x, y: (data_augmentation(x, training=True), y) ).prefetch(buffer_size=tf.data.AUTOTUNE)`
Image Cropping	[12]: `resize_and_rescale = tf.keras.Sequential([ layers.Resizing(IMAGE_SIZE, IMAGE_SIZE), layers.Rescaling(1./255), ])`

Batch Normalization	```
	input_shape = (BATCH_SIZE, IMAGE_SIZE, IMAGE_SIZE, CHANNELS)
	n_classes = 3
	
	model = models.Sequential([
	    resize_and_rescale,
	    layers.Conv2D(32, kernel_size = (3,3), activation='relu', input_shape=input_shape),
	    layers.MaxPooling2D((2, 2)),
	    layers.Conv2D(64, kernel_size = (3,3), activation='relu'),
	    layers.MaxPooling2D((2, 2)),
	    layers.Conv2D(64, kernel_size = (3,3), activation='relu'),
	    layers.MaxPooling2D((2, 2)),
	    layers.Conv2D(64, (3, 3), activation='relu'),
	    layers.MaxPooling2D((2, 2)),
	    layers.Conv2D(64, (3, 3), activation='relu'),
	    layers.MaxPooling2D((2, 2)),
	    layers.Conv2D(64, (3, 3), activation='relu'),
	    layers.MaxPooling2D((2, 2)),
	    layers.Flatten(),
	    layers.Dense(64, activation='relu'),
	    layers.Dense(n_classes, activation='softmax'),
	])
	``