Data Collection and Preprocessing Phase

Date	20 October 2024
Team ID	739755
Project Title	Bird Species Classification
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 used is *200 Bird Species with 11,788 Images* from Kaggle, stored in Google Drive at C:/Users/manda/OneDrive/Desktop/Bird Species Classification/major/CUB_200_2011/CUB_200_2011/images. It consists of 200 bird species, each with multiple images. The images are labeled and categorized for classification.
Resizing	The images are resized to *(224 × 224 pixels)* to ensure uniform input size for the CNN model. The resizing is performed using torchvision.transforms.Resize() in PyTorch. Example: transforms.Resize((224, 224)).
Normalization	Normalization scales pixel values to a specific range for stable training. The images are normalized using [0,1] scaling (dividing by 255) or standard normalization (mean=0.5, std=0.5). The torchvision.transforms.Normalize() function is used. Example: transforms.Normalize(mean=[0.5, 0.5, 0.5], std=[0.5, 0.5, 0.5]).
Data Augmentation	Data augmentation techniques such as random rotation, horizontal flipping, brightness cropping are applied to increase dataset diversity. This is done using torchvision.transf transforms.RandomHorizontalFlip(), transforms.RandomRotation(15).

Denoising	Denoising is applied using Gaussian Blurring to remove noise from images for better feature extraction. The cv2.GaussianBlur() function is used. Example: cv2.GaussianBlur(image, (5, 5), 0)
Edge Detection	Edge detection is performed using the Canny Edge Detection technique to highlight bird contours. The cv2.Canny() function is used. Example: cv2.Canny(image, 100, 200).
Color Space Conversion	The images are converted from RGB to Grayscale to highlight different features. The conversion is done using OpenCV's cv2.cvtColor() function. Example: cv2.cvtColor(image, cv2.COLOR_BGR2GRAY).
Image Cropping	Images are cropped to focus on the bird and remove unnecessary background. Bounding box cropping is applied if annotations are available. Cropping is performed using PIL or cv2. Example: cropped_image = image[y1:y2, x1:x2].
Batch Normalization	Batch normalization is applied after convolutional layers in the CNN model to stabilize training and accelerate convergence. This is done using torch.nn.BatchNorm2d(). Example: nn.BatchNorm2d(num_features=64).
Data Preprocessing Code Screenshots	
Loading Data	
Normalization	

Data preprocessing	

Pre Trained	`#model torch.hub.load('facebookresearch/deit:main', 'resnlp 12, pretrained-True)`
	`torch.backends.cudnn.benchmark = True`
	`model= torchvision.models.efficientnet_b0(pretrained=True)`
	
	`/usr/local/lib/python3.10/dist-packages/torchvision/models/_utils.py:208: UserWarning: The parame`
	`warnings.warn(`
	`/usr/local/lib/python3.10/dist-packages/torchvision/models/_utils.py:223: UserWarning: Arguments`
	`warnings.warn(msg)`
	`Downloading: "https://download.pytorch.org/models/efficientnet_b0_rwightman-7f5810bc.pth" to /roo`
	`100%| | 20.5M/20.5M [00:00<00:00, 66.9MB/s]`

Adding Layers	`import torch`
	`device = torch.device("cuda:0" if torch.cuda.is_available() else "cpu")`
	`def count_parameters(model):`
	`return sum(p.numel() for p in model.parameters() if p.requires_grad)`
	
	`# Get in_features before redefining the classifier`
	`n_inputs = model.classifier[1].in_features  # Use the original model to get this value`
	
	`for param in model.parameters():`
	`param.requires_grad = False`
	
	`# Now redefine the classifier`
	`model.classifier = nn.Sequential(`
	`nn.Linear(n_inputs, 2048),`
	`nn.SiLU(),`
	`nn.Dropout(0.3),`
	`nn.Linear(2048, len(classes))`
	`)`
	`model = model.to(device)`
	
	`print(model.classifier)`
	
	`Sequential(`
	`(0): Linear(in_features=1280, out_features=2048, bias=True)`
	`(1): SiLU()`
	`(2): Dropout(p=0.3, inplace=False)`
	`(3): Linear(in_features=2048, out_features=1, bias=True)`
	`)`

Training and Testing	`dataloaders ={`
	`"train": train_loader,`
	`"val": val_loader`
	`}`
	`dataset_sizes= {`
	`"train": train_data_len,  # Use the variable train_data_len assigned earlier`
	`"val": valid_data_len  # Use the variable valid_data_len assigned earlier`
	`}`
	
	`print(len(train_loader))`
	`print(len(val_loader))`
	`print(len(test_loader))`
	
	`59`
	`146`
	`146`
	
	`print(train_data_len, test_data_len, valid_data_len)`
	
	`14876 9298 9298`

Save Model	 model.save('/content/drive/MyDrive/Bird Species Classification/Manasa/Bird_Species_Model.h5') WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save_model(model)`. Th