Data Collection and Preprocessing Phase

Date	15 June 2025
Team ID	SWTID1749820017
Project Title	Dog Breed Identification using Transfer Learning
Maximum Marks	6 Marks

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	Give an overview of the data, which you're going to use in your project.
Resizing	Resize images to a specified target size.
Normalization	Normalize pixel values to a specific range.
Data Augmentation	Apply augmentation techniques such as flipping, rotation, shifting, zooming, or shearing.
Denoising	Apply denoising filters to reduce noise in the images.
Edge Detection	Apply edge detection algorithms to highlight prominent edges in the images.

Color Space Conversion	Convert images from one color space to another.
Image Cropping	Crop images to focus on the regions containing objects of interest.
Batch Normalization	Apply batch normalization to the input of each layer in the neural network.

Data Preprocessing Code Screenshots

Loading Data

```
from pathlib import Path
import urllib.request
import tarfile
target_folder = Path("/content/drive/MyDrive/Dog_Classifier/Dataset")
target_folder.mkdir(parents=True, exist_ok=True)
tar_path = target_folder / "images.tar"
url = "http://vision.stanford.edu/aditya86/ImageNetDogs/images.tar"
if not tar_path.exists():
    print("Downloading...")
    urllib.request.urlretrieve(url, tar_path)
    print("Download completed.")
print("Extracting...")
with tarfile.open(tar_path, "r") as tar:
    tar.extractall(path=target_folder)
print("Extraction completed.")
```

Resizing

```
DATASET_DIR = "/content/drive/MyDrive/Dog_Classifier/Dataset/Images"
IMG_SIZE = (331, 331)
BATCH_SIZE = 128
```

Normalization

```
[3]: datagen = ImageDataGenerator(
         rescale=1./255,
```

Data Augmentation

```
    validation_split=0.2,
    horizontal_flip=True,
    zoom_range=0.2,
    shear_range=0.2
)
```

Denoising

The dataset (Stanford Dogs) contains high-quality, pre-curated images with minimal noise. Deep learning models like NASNet are robust to minor noise and often perform their own feature extraction. Adding

	denoising might blur fine breed-specific textures like fur patterns, which are important for accurate classification.
	`import cv2`
	
	`def denoise_images(input_path, output_path):`
	`  img = cv2.imread(input_path)`
	`  denoised = cv2.fastNlMeansDenoisingColored(img, None, 10, 10, 7, 21)`
	`  cv2.imwrite(output_path, denoised)`

Edge Detection	Edge detection techniques like Canny or Sobel are designed for traditional computer vision pipelines — not deep learning. Convolutional Neural Networks (CNNs) automatically learn edge-like filters in their early layers. Explicit edge detection preprocessing can strip raw pixel information and reduce model performance.
	`def detect_edges(img_path):`
	`  img = cv2.imread(img_path, 0)`
	`  edges = cv2.Canny(img, 100, 200)`
	`  return edges`

Color Space Conversion	NASNet and most pre-trained CNN architectures are designed for RGB inputs (same as ImageNet, which they're trained on). Converting images to grayscale or other color spaces like HSV would remove valuable breed-identifying color patterns (like coat shades or markings), potentially lowering model accuracy.
	`def convert_to_gray(img_path):`
	`  img = cv2.imread(img_path)`
	`  gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)`
	`  return gray`

Image Cropping	The images in the dataset are already fairly centered on the dog. Manual or automatic cropping might accidentally cut off important breed features like ears, tail, or posture. Instead, resizing with preserved aspect ratio and padding ensures that the entire dog remains in view for the model.

	`def crop_center(img, cropx, cropy):`
	`  y, x, _ = img.shape`
	`  startx = x//2 - cropx//2`
	`  starty = y//2 - cropy//2`
	`  return img[starty:starty+cropy, startx:startx+cropx]`

Batch Normalization	**NASNet** already includes batch normalization in its architecture.