Data Collection and Preprocessing Phase

Date	28 Sep 2024
Team ID	724803
Project Title	Railway Sentry: Detecting Workers on Railway Tracks using YOLO V9
Maximum Marks	6 Marks

Preprocessing Template

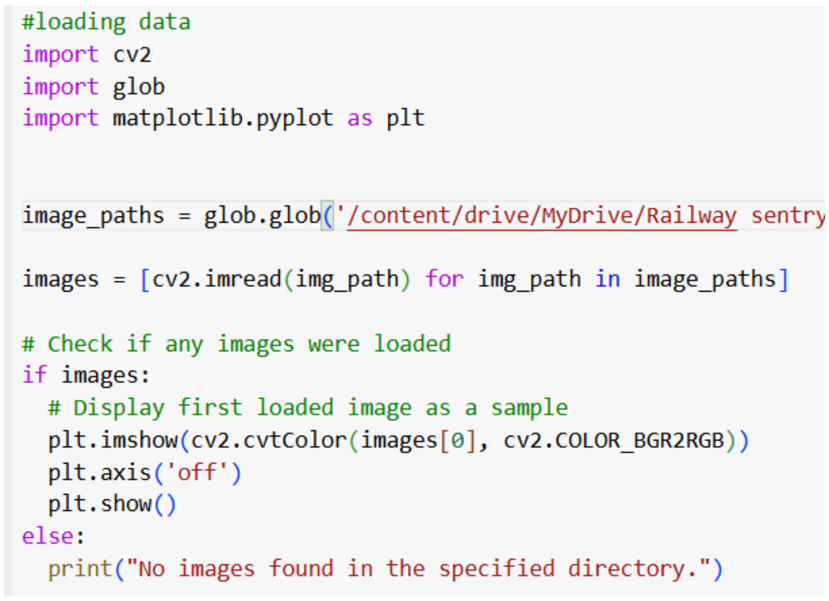
Prepare and annotate a diverse dataset of railway scenes with workers on tracks under various conditions (lighting, weather). Resize, normalize, and split data for training/testing to optimize YOLO V9 worker detection accuracy.

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.

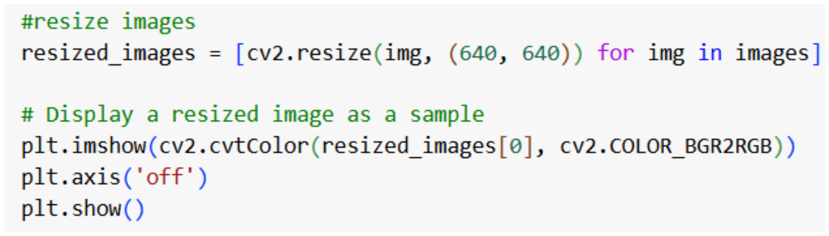
Data Preprocessing Code Screenshots

Loading Data



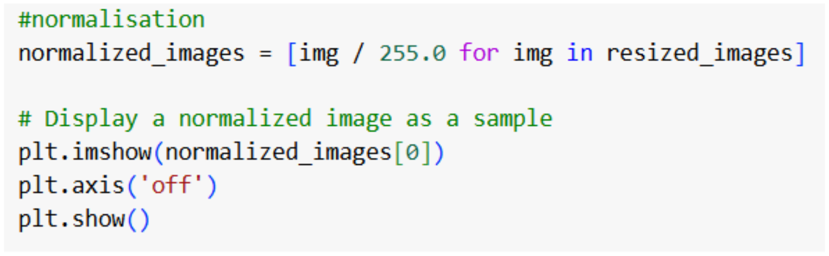
```python
#loading data
import cv2
import glob
import matplotlib.pyplot as plt


image_paths = glob.glob('/content/drive/MyDrive/Railway sentry

images = [cv2.imread(img_path) for img_path in image_paths]

# Check if any images were loaded
if images:
  # Display first loaded image as a sample
  plt.imshow(cv2.cvtColor(images[0], cv2.COLOR_BGR2RGB))
  plt.axis('off')
  plt.show()
else:
  print("No images found in the specified directory.")
```

Resizing

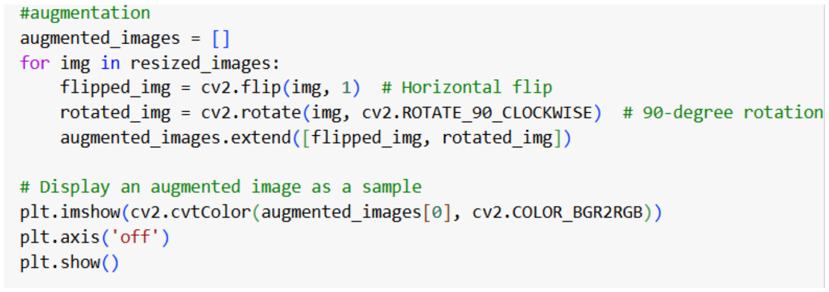


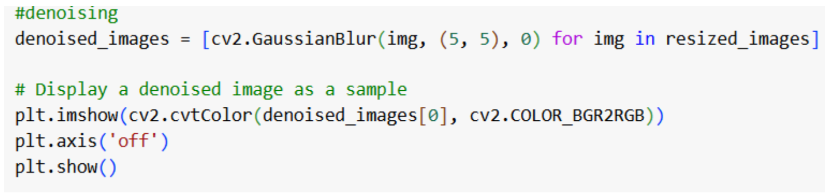
```python
#resize images
resized_images = [cv2.resize(img, (640, 640)) for img in images]

# Display a resized image as a sample
plt.imshow(cv2.cvtColor(resized_images[0], cv2.COLOR_BGR2RGB))
plt.axis('off')
plt.show()
```

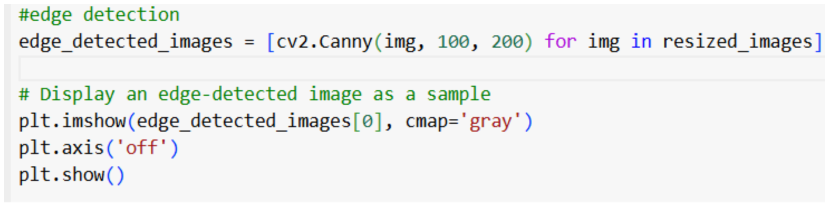
Normalization



```python
#normalisation
normalized_images = [img / 255.0 for img in resized_images]

# Display a normalized image as a sample
plt.imshow(normalized_images[0])
plt.axis('off')
plt.show()
```

Data Augmentation	 `#augmentation`
	`augmented_images = []`
	`for img in resized_images:`
	`    flipped_img = cv2.flip(img, 1)  # Horizontal flip`
	`    rotated_img = cv2.rotate(img, cv2.ROTATE_90_CLOCKWISE)  # 90-degree rotation`
	`    augmented_images.extend([flipped_img, rotated_img])`
	
	`# Display an augmented image as a sample`
	`plt.imshow(cv2.cvtColor(augmented_images[0], cv2.COLOR_BGR2RGB))`
	`plt.axis('off')`
	`plt.show()`

Denoising	 `#denoising`
	`denoised_images = [cv2.GaussianBlur(img, (5, 5), 0) for img in resized_images]`
	
	`# Display a denoised image as a sample`
	`plt.imshow(cv2.cvtColor(denoised_images[0], cv2.COLOR_BGR2RGB))`
	`plt.axis('off')`
	`plt.show()`

Edge Detection	 `#edge detection`
	`edge_detected_images = [cv2.Canny(img, 100, 200) for img in resized_images]`
	
	`# Display an edge-detected image as a sample`
	`plt.imshow(edge_detected_images[0], cmap='gray')`
	`plt.axis('off')`
	`plt.show()`

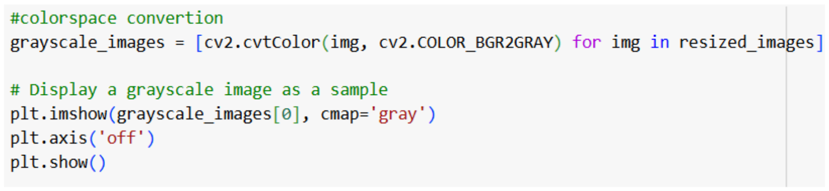
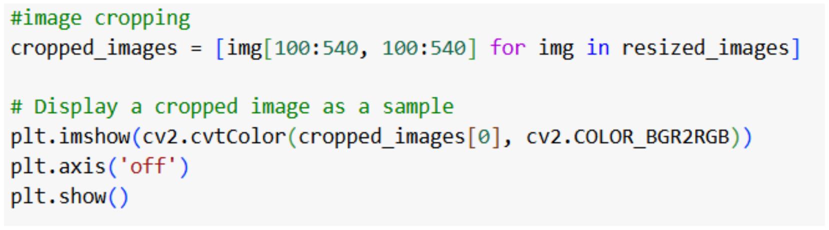
Color Space Conversion	 `#colorspace convertion`
	`grayscale_images = [cv2.cvtColor(img, cv2.COLOR_BGR2GRAY) for img in resized_images]`
	
	`# Display a grayscale image as a sample`
	`plt.imshow(grayscale_images[0], cmap='gray')`
	`plt.axis('off')`
	`plt.show()`

Image Cropping	 `#image cropping`
	`cropped_images = [img[100:540, 100:540] for img in resized_images]`
	
	`# Display a cropped image as a sample`
	`plt.imshow(cv2.cvtColor(cropped_images[0], cv2.COLOR_BGR2RGB))`
	`plt.axis('off')`
	`plt.show()`