

Data Collection and Preprocessing Phase

Date	28 Sep 2024
Team ID	739648
Project Title	Guardian Eye:Yolo-Based Smart Helmet Detection System For Enhanced Safety In Real-Time
Maximum Marks	6 Marks

Preprocessing Template

The preprocessing template for a smart helmet detection system involves capturing real-time video data, followed by image enhancement techniques such as noise reduction and contrast adjustment for clearer visibility. The next step is to apply object detection algorithms to localize and identify helmets in the scene. Finally, region-based processing helps isolate areas of interest for efficient and accurate real-time detection..

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	<pre>from ultralytics import YOLO model = YOLO('yolov5s.pt') results = model.train(data="/content/drive/MyDrive/helmet/Hard-Hat-Sample-1/data.yaml", epochs=100)</pre> <p>Show hidden output</p> <pre>results = model.val(data="/content/drive/MyDrive/helmet/Hard-Hat-Sample-1/data.yaml", epochs=100);</pre> <p>Ultralytics 8.3.40 Python-3.10.12 torch-2.5.1+cu121 CUDA:0 (Tesla T4, 15102MiB) YOLOv5s summary (fused): 193 layers, 9,112,697 parameters, 0 gradients, 23.8 GFLOPs val: Scanning /content/drive/MyDrive/helmet/Hard-Hat-Sample-1/valid/labels.cache... 20 images, 0 backgrounds, 0 corrupt: 100% ██████████ 20/20 [00:00<?, ?it/s]</p> <table><thead><tr><th>Class</th><th>Images</th><th>Instances</th><th>Box(P)</th><th>R</th><th>mAP50</th><th>mAP50-95</th></tr></thead><tbody><tr><td>all</td><td>20</td><td>65</td><td>0.985</td><td>0.6</td><td>0.647</td><td>0.472</td></tr><tr><td>head</td><td>3</td><td>18</td><td>0.981</td><td>0.889</td><td>0.979</td><td>0.7</td></tr><tr><td>helmet</td><td>17</td><td>45</td><td>0.974</td><td>0.911</td><td>0.963</td><td>0.716</td></tr><tr><td>person</td><td>1</td><td>2</td><td>1</td><td>0</td><td>0</td><td>0</td></tr></tbody></table> <p>Speed: 0.2ms preprocess, 16.5ms inference, 0.0ms loss, 2.1ms postprocess per image Results saved to runs/detect/train352</p> <pre>!yolo task=detect mode=predict model=/content/drive/MyDrive/helmet/runs/detect/train20/weights/best.pt</pre> <p>WARNING ⚠ 'source' argument is missing. Using default 'source=/usr/local/lib/python3.10/dist-packages/ultralytics/assets'. Ultralytics 8.3.40 Python-3.10.12 torch-2.5.1+cu121 CUDA:0 (Tesla T4, 15102MiB) YOLOv5s summary (fused): 193 layers, 9,112,697 parameters, 0 gradients, 23.8 GFLOPs</p>	Class	Images	Instances	Box(P)	R	mAP50	mAP50-95	all	20	65	0.985	0.6	0.647	0.472	head	3	18	0.981	0.889	0.979	0.7	helmet	17	45	0.974	0.911	0.963	0.716	person	1	2	1	0	0	0
Class	Images	Instances	Box(P)	R	mAP50	mAP50-95																														
all	20	65	0.985	0.6	0.647	0.472																														
head	3	18	0.981	0.889	0.979	0.7																														
helmet	17	45	0.974	0.911	0.963	0.716																														
person	1	2	1	0	0	0																														
Resizing																																				

Normalization	-----
Data Augmentation	-----
Denoising	-----
Edge Detection	-----
Color Space Conversion	<pre> image_path = "/content/drive/MyDrive/helmet/archive/images/BikesHelmets0.png" results = model(image_path) result = results[0] boxes = result.bboxes.xyxy labels = result.bboxes.cls confidences = result.bboxes.conf for box, label, confidence in zip(boxes, labels, confidences): print(f"Bounding Box: {box.tolist()}") print(f"Class Label: {label.item()}") print(f"Confidence: {confidence.item()}") results[0].plot() </pre>

Image
Cropping

```
image_path = "/content/drive/MyDrive/helmet/archive/images/BikesHelmets0.png"
results = model(image_path)
result = results[0]
boxes = result.bboxes.xyxy
labels = result.bboxes.cls
confidences = result.bboxes.conf
for box, label, confidence in zip(boxes, labels, confidences):
    print(f"Bounding Box: {box.tolist()}")
    print(f"Class Label: {label.item()}")
    print(f"Confidence: {confidence.item()}")
results[0].plot()
```