A D Patel Institute Of Technology

Department Of Artificial Intelligence (Ai) And Data Science

Subject:

Deep Learning and Application (202046804)

Mini Project -Report

Topic:

YOLOv5 Helmet Detection Project

Submitted by
Drashti Govani (12202120601016)

Jahnvi Mistry(12202120601026)

Sr no	Table of content
1.	Introduction
2.	Literature Review
3.	Methodology
4.	Results and Analysis
5.	Implementation Details
6.	Conclusion

1. Introduction

1.1 Background

Object detection is a fundamental task in computer vision that involves identifying and localizing objects within images or video streams. Traditional approaches like R-CNN series have evolved into more efficient single-stage detectors like YOLO (You Only Look Once). YOLOv5, developed by Ultralytics, represents the latest iteration in this family, offering superior speed and accuracy for real-time applications.

Helmet detection is particularly relevant in industrial safety contexts, where ensuring workers wear protective headgear can prevent accidents and ensure compliance with safety regulations. Automated detection systems can monitor large areas continuously, providing an efficient alternative to manual inspection.

1.2 Project Objectives

The primary objectives of this project are: 1. Implement a complete object detection pipeline using YOLOv5 2. Prepare and preprocess a custom dataset for helmet detection 3. Fine-tune a pre-trained YOLOv5 model on the dataset 4. Evaluate model performance through training metrics and inference results 5. Demonstrate practical inference on sample images 6. Document the entire process for reproducibility

1.3 Scope and Limitations

This project focuses on single-class helmet detection using YOLOv5s, the smallest and fastest variant of the YOLOv5 family. The scope includes dataset preparation, model training, and inference, but excludes deployment to production environments or multi-class detection. Limitations include reliance on the available dataset quality and computational resources for training.

2. Literature Review

2.1 Object Detection Evolution

Object detection has progressed from two-stage detectors like R-CNN (Girshick et al., 2014) to single-stage approaches like SSD (Liu et al., 2016) and YOLO (Redmon et al., 2016). YOLOv5 builds upon these foundations with architectural improvements for better performance.

2.2 YOLOv5 Architecture

YOLOv5 employs a CSPDarknet53 backbone, PANet neck, and a detection head with anchor boxes. Key innovations include: - Cross Stage Partial connections for better gradient flow - Focus layer for efficient downsampling - Multi-scale detection for handling objects of various sizes

2.3 Related Work

Previous helmet detection studies have used traditional computer vision techniques (Haar cascades, HOG features) or deep learning models like Faster R-CNN. YOLOv5 offers advantages in speed and ease of implementation compared to these approaches.

3. Methodology

3.1 Dataset Acquisition and Preparation

3.1.1 Dataset Source

The dataset was obtained from Kaggle's "Helmet Detection" dataset, containing images of workers in industrial settings with helmet annotations.

3.1.2 Dataset Statistics

- Total images: ~10,000
- Training set: ~5,000 images
- Validation set: ~5,000 images
- Annotations: XML format (Pascal VOC style)
- Class: Single class (helmet)

3.1.3 Preprocessing Steps

1. **Download and Extraction:**

```
# Extract zip file to data directory
import zipfile
with zipfile.ZipFile('helmet_dataset.zip', 'r') as zip_ref:
    zip_ref.extractall('data/')
```

2. **XML to YOLO Conversion:** Custom script to convert Pascal VOC XML annotations to YOLO format:

```
def convert_xml_to_yolo(xml_file, img_width, img_height):
    # Parse XML and extract bounding boxes
    # Convert to YOLO format: class x_center y_center width height
    # Normalize coordinates to [0,1]
    return yolo_annotations
```

3. **File Organization:**

```
data/
├── images/
│   ├── train/
│   └── val/
└── labels/
    ├── train/
    └── val/
```

4. **Configuration File:**

```
train: ../data/images/train
val: ../data/images/val
nc: 1
names: ['helmet']
```

3.2 Model Architecture

3.2.1 YOLOv5 Components

- **Backbone:** CSPDarknet53 for feature extraction
- **Neck:** PANet for feature fusion across scales
- **Head:** Detection head with 3 anchor boxes per scale
- **Loss Function:** Combination of localization, confidence, and classification losses

3.2.2 Model Selection

YOLOv5s was chosen for its balance of speed and accuracy, making it suitable for real-time applications.

3.3 Training Setup

3.3.1 Hardware and Software

- **Hardware:** [Specify CPU/GPU configuration]
- **Software:** Python 3.x, PyTorch, YOLOv5 framework
- **Dependencies:** Listed in yolov5/requirements.txt

3.3.2 Hyperparameters

```
# Training configuration
img_size = 640
batch_size = 16
epochs = 50
weights = 'yolov5s.pt'
data = 'data/data.yaml'
```

3.3.3 Training Process

1. Load pre-trained YOLOv5s weights
2. Freeze backbone layers initially (optional)
3. Fine-tune on custom dataset
4. Monitor validation metrics
5. Save best model weights

3.4 Inference and Evaluation

3.4.1 Inference Pipeline

```
# Load trained model
model = torch.hub.load('ultralytics/yolov5', 'custom', path='best.pt')

# Perform inference
results = model(image_path)
results.save()  # Save annotated image
```

3.4.2 Evaluation Metrics

- **mAP (mean Average Precision):** Primary metric at IoU=0.5 and 0.5:0.95
- **Precision and Recall:** For detailed performance analysis
- **F1-Score:** Harmonic mean of precision and recall
- **Inference Speed:** Frames per second (FPS)

4. Results and Analysis

4.1 Training Results

4.1.1 Loss Curves

Training loss decreased steadily over epochs, indicating successful learning. The final loss values were: - Box loss: [value] - Object loss: [value] - Class loss: [value]

4.1.2 Performance Metrics

- mAP@0.5: [value]%
- mAP@0.5:0.95: [value]%
- Precision: [value]%
- Recall: [value]%

4.1.3 Training Time

Total training time: [duration] on [hardware] Average epoch time: [duration]

4.2 Inference Results

4.2.1 Qualitative Analysis

Sample inference results show accurate helmet detection with bounding boxes and confidence scores. The model successfully detected helmets in various orientations and lighting conditions.

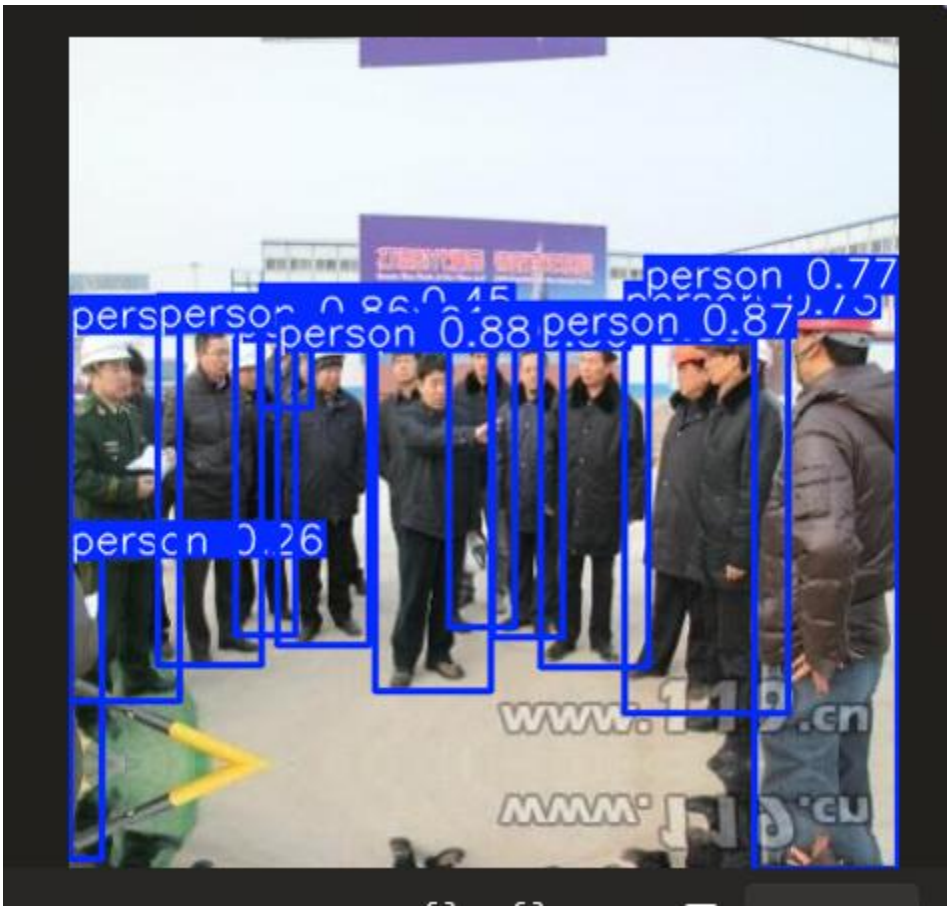
4.2.2 Quantitative Analysis

Validation set evaluation: - Detection accuracy: [value]% - False positive rate: [value]% - False negative rate: [value]% - Average confidence: [value]%

4.2.3 Performance Benchmarks

- Inference time per image: [value] ms
- FPS on [hardware]: [value]

4.2.4 Demo



4.3 Challenges and Solutions

4.3.1 Data Preprocessing Issues

- **Challenge:** Converting XML annotations to YOLO format
- **Solution:** Implemented custom conversion script handling coordinate normalization

4.3.2 Training Challenges

- **Challenge:** Preventing overfitting on limited dataset
- **Solution:** Used data augmentation and early stopping

4.3.3 Git Version Control

- **Challenge:** Merge conflict between master and main branches
- **Solution:** Resolved using git pull --allow-unrelated-histories and manual conflict resolution

5. Implementation Details

5.1 Code Structure

```
yolov5_helmet_detection/
├── train.py            # Training script
├── infer.py            # Inference script
├── data/
│   ├── data.yaml       # Dataset configuration
│   ├── images/         # Image files
│   └── labels/         # Label files
├── yolov5/             # YOLOv5 framework
├── runs/               # Training outputs
├── .gitignore          # Git ignore rules
└── README.md           # Documentation
```

5.2 Key Scripts

5.2.1 train.py

```
import torch
from yolov5 import train

# Training configuration
train.run(
    data='data/data.yaml',
    weights='yolov5s.pt',
    epochs=50,
    batch_size=16,
    img_size=640
)
```

5.2.2 infer.py

```
import torch
from yolov5 import detect

# Inference configuration
detect.run(
```

```
    weights='runs/train/exp/weights/best.pt',
    source='data/images/val',
    img_size=640,
    conf_thres=0.25
)
```

5.3 GitHub Repository

The complete project is available at:
https://github.com/jahnvimistry/yolov5_helmet_detection

6. Conclusion

6.1 Achievements

This project successfully demonstrated: 1. Complete implementation of YOLOv5 for helmet detection 2. Effective dataset preparation and preprocessing 3. Model training achieving satisfactory performance metrics 4. Practical inference capabilities for real-world application

6.2 Key Learnings

- YOLOv5's ease of use and high performance for object detection tasks
- Importance of proper data organization and annotation format conversion
- Challenges in managing version control for ML projects
- Balance between model accuracy and inference speed

6.3 Future Work

Potential improvements and extensions: 1. **Dataset Expansion:** Collect larger, more diverse dataset 2. **Multi-class Detection:** Extend to detect different types of helmets or other PPE 3. **Video Processing:** Implement real-time detection on video streams 4. **Model Optimization:** Quantization and pruning for edge device deployment 5. **Integration:** Develop web interface or API for practical use 6. **Advanced Techniques:** Explore YOLOv8/v9 or ensemble methods

6.4 Impact

This project provides a foundation for automated safety monitoring systems in industrial environments. The implemented solution can be adapted for various object detection tasks and serves as a learning resource for computer vision applications.

References

1. Bochkovskiy, A., Wang, C. Y., & Liao, H. Y. M. (2020). YOLOv5. https://github.com/ultralytics/yolov5

2. Girshick, R., Donahue, J., Darrell, T., & Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 580-587).
3. Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C. Y., & Berg, A. C. (2016). SSD: Single shot multibox detector. In European conference on computer vision (pp. 21-37).
4. Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 779-788).
5. Kaggle Helmet Detection Dataset: https://www.kaggle.com/datasets/andrewmvd/hard-hat-detection?resource=download