**Report**

**Construction Site Safety Object Detection**

Abstract

This report details the development of an object detection system for construction site safety using the Construction Site Safety Image Dataset from RoboFlow and a YOLOv8-Mobilenet model implemented with the Ultralytics library in Python. The project aimed to detect safety-related objects such as hardhats, safety vests, and machinery to enhance automated safety monitoring. Despite challenges with Kaggle API configuration, the training process was completed, achieving moderate performance. This report summarizes the methodology, results, and limitations encountered.

Introduction

Object detection in construction sites is vital for ensuring worker safety by identifying critical safety equipment and hazards. The Construction Site Safety Image Dataset, containing images annotated with 10 classes (e.g., Hardhat, Mask, Person, Safety Cone), provides a foundation for training deep learning models to automate safety monitoring.

Objectives

To develop a YOLOv8-Mobilenet model using the Ultralytics library for detecting safety-related objects in construction site images.

To evaluate the model’s performance using metrics such as precision, recall, and mean Average Precision (mAP).

To address challenges related to dataset setup and training configuration.

Dataset

Construction Site Safety Image Dataset (RoboFlow)

Contains images (exact count not specified, but validation set includes 114 images with 697 instances) sourced from construction sites.

Images are annotated with 10 classes: Hardhat, Mask, NO-Hardhat, NO-Mask, NO-Safety Vest, Person, Safety Cone, Safety Vest, Machinery, Vehicle.

Hosted on Kaggle and downloaded as a zip file for training.

Methodology

Data Preprocessing

Extraction: Dataset unzipped to access training, validation, and test images with corresponding labels.

Augmentation: Applied mosaic augmentation until epoch 40, followed by lighter augmentations (Blur, MedianBlur, ToGray, CLAHE) to enhance dataset diversity.

Model Architecture

YOLOv8-Mobilenet model, configured via `yolov8-mobilenet.yaml`, with 9.27M parameters and 26.4 GFLOPs.

Architecture includes convolutional layers optimized for efficiency, suitable for resource-constrained environments.

Loss functions: Box loss, classification loss, and Distribution Focal Loss (DFL).

Optimizer: Not explicitly stated but typically SGD or Adam in YOLOv8.

Training

Batch Size: Not explicitly specified, but 163 training batches suggest a reasonable batch size (e.g., 8 or 16).

Epochs: 50 epochs, completed in ~1 hour on a Tesla T4 GPU.

Image Size: 640x640 pixels.

Learning Rate: Not specified but typically adaptive in YOLOv8.

- Training monitored with metrics like box loss, classification loss, and DFL loss, with validation after each epoch.

Results

Training Performance

- Training completed 50 epochs, with final validation metrics: Precision = 0.709, Recall = 0.481, mAP50 = 0.55, mAP50-95 = 0.254.

Class-specific performance:

Strong: Hardhat (mAP50: 0.656), Mask (mAP50: 0.818), Safety Cone (mAP50: 0.78).

Weak: Vehicle (mAP50: 0.263), NO-Mask (mAP50: 0.391).

Model weights saved as `last.pt` and `best.pt` (18.9MB each).

Challenges

Kaggle API setup failed due to a missing `kaggle.json` file, though dataset download proceeded.

Moderate model performance suggests potential for further optimization, especially for underrepresented classes like Vehicle.

Discussion

The YOLOv8-Mobilenet model shows promise for construction site safety monitoring, with strong detection for key safety equipment like hardhats and masks. However, lower recall for some classes indicates a need for more balanced data or extended training. The lightweight architecture is well-suited for deployment on edge devices, but performance could be improved with hyperparameter tuning or a larger dataset.

Conclusion

The project successfully implemented an object detection system for construction site safety using the YOLOv8-Mobilenet model and the Construction Site Safety Image Dataset. Despite setup challenges, the model achieved moderate performance, demonstrating its potential for real-world safety applications. Future work should focus on resolving configuration issues, optimizing model performance, and exploring cloud-based training to enhance scalability.