MLHD: Machine Learning for Human Detection — Progress Report

Daveed Vodonenko Lucas Portela Darius Vulturu

Data Analysis

We built the dataset from CCTV factory footage. First, we labeled each frame with a YOLO detector and refined the annotations in LabelMe. We then exported the labels from .json to YOLO .txt format for training. Each 1920×1080 RGB image was normalized, resized to 416×416 with gray padding, and rescaled bounding boxes to preserve spatial consistency. We converted the processed images into PyTorch tensors of shape [3, 416, 416] and encoded the labels into a [26, 26, 5] grid. During training, the DataLoader batches eight samples per iteration, producing tensors of size [8, 3, 416, 416] for the images and [8, 26, 26, 5] for the corresponding labels.

We analyzed the dataset's spatial and statistical structure by aggregating all bounding boxes across the CCTV frames. Heatmaps of bounding-box centers (see Figure 1) showed clustering in the lower and central areas of each frame, consistent with typical worker positioning in factory layouts. A 26×26 grid occupancy analysis identified 64 overlapping detections within single cells, which exposes a key limitation of the YOLO architecture: its one-object-per-cell constraint. This design choice systematically undercounts individuals in crowded scenes, especially when workers stand close together.

We found that the number of individuals per frame remained relatively stable, with most images showing three to five persons. This consistency indicates a steady worker density across the dataset. These findings shape our modeling approach. To reduce positional bias and improve generalization, we plan to implement spatial data augmentation techniques, such as random cropping, shifts, and flips, and to explore attention-based weighting or region-prior regularization to enhance detection accuracy in high-activity zones.

Model Choice and Implementation

We designed a Convolutional Neural Network (CNN) for object detection in CCTV footage, following a YOLO-style dense prediction framework. The model uses a nine-layer backbone with progressive downsampling that increases channel depth from 3 to 512, followed by a 1×1 convolutional head that outputs a [S, S, 5] tensor. Each grid cell predicts objectness and bounding-box offsets through sigmoid activation, allowing spatially distributed detection across the frame. This design captures fine-grained positional information while remaining computationally efficient enough for real-time inference.

We implemented the network in PyTorch, using torchvision for dataset handling and torch.utils.data for batch processing. Custom training utilities manage all key processes, including epoch-level training, evaluation, and fitting. A custom detection loss function aligns predicted boxes with ground truth based on confidence and localization accuracy. The system supports training across CPU, CUDA, and Apple's MPS backends, ensuring flexible deployment on different hardware setups.

To stabilize optimization, we integrated checkpointing and learning-rate scheduling. These components allow the model to resume training efficiently and avoid convergence issues during longer runs. For future work, we plan to replace the PyTorch default convolutional layers with a custom implementation to improve control over kernel behavior and potentially optimize the architecture for edge inference performance.

Preliminary Results

We trained the model for 50 epochs with a batch size of 8 and an input resolution of 416×416. Early in training, we observed clear signs of overfitting as the validation loss began to diverge from the training loss. The limited dataset size, with 1,958 training images, and the absence of sufficient data augmentation contributed to this pattern. To prevent further deterioration, we applied early stopping with a patience parameter of 10 epochs, allowing the model to halt automatically once performance plateaued.

We evaluated the trained network on a validation set containing 218 images and 753 labeled objects. The following metrics summarize its detection performance:

- Precision: 53.6% at IoU threshold 0.5
- Recall: 52.2% at IoU threshold 0.5
- F1 Score: 0.529
- mAP: 0.303 (IoU 0.5) and 0.024 (IoU 0.75)

These results reveal a moderate level of detection capability given the dataset constraints. Precision, which measures the share of correct detections among all predicted boxes, and recall, which captures the proportion of ground-truth objects successfully detected, remain balanced. This indicates that the model identifies objects at a reasonable rate without producing excessive false positives. The F1 score, the harmonic mean of precision and recall, confirms this balance between accuracy and completeness. The mean Average Precision (mAP), which averages detection precision across different recall levels, declines sharply at higher Intersection over Union (IoU) thresholds. IoU quantifies how closely a predicted bounding box overlaps with the ground truth, so a lower mAP at stricter IoU thresholds reveals weak localization accuracy. In practical terms, the model often detects the correct objects but struggles to align bounding-box boundaries precisely. Addressing this issue will require stronger data augmentation and refined loss weighting to penalize poor localization more effectively.

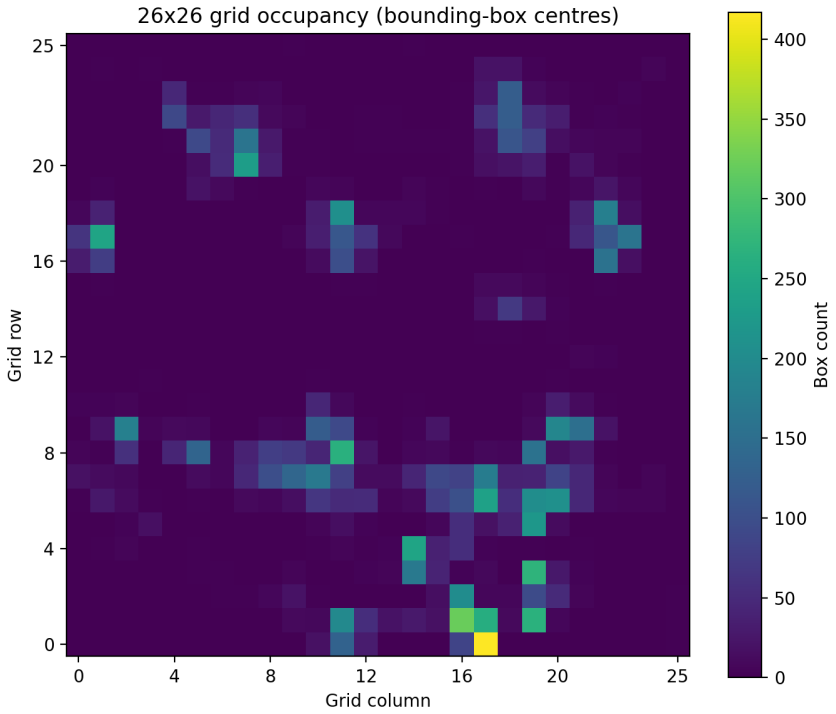

Figure 1: 26×26 grid occupancy heatmap showing clustered detections.

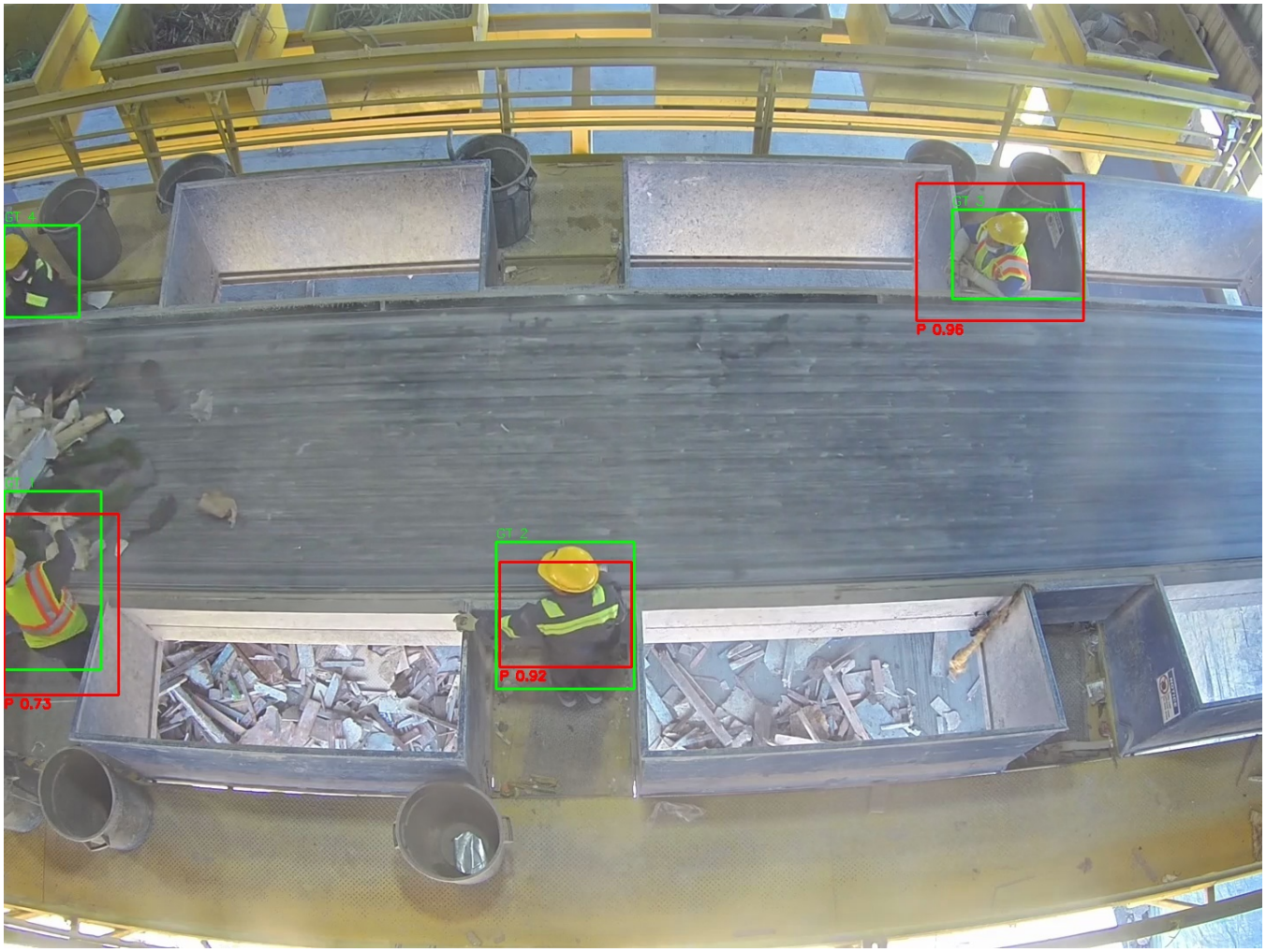

Figure 2: Image 520 with Ground Truth (green box) vs Prediction (red box).