

Assessing the Influence of Learning Rate on Human Detection for Safety Applications Using Deep Learning

Pranav Krishnapuram¹, Atul Dubey²

¹ Tessellations, Cupertino, California

² AIClub, Mountain View, California

SUMMARY

Work-related injuries and fatalities impose significant financial burdens, costing over \$171 billion annually, with direct compensation to victims exceeding \$1 billion. The deployment of automated technological solutions based on deep neural networks could help detect humans on the scene and help authorities reduce workplace injuries. This study investigates the potential of a deep learning-based object detection model, employing the YOLOv8 architecture, to enhance surveillance and safety in work environments. We hypothesized that the YOLOv8 model, adapted via transfer learning, would achieve a mean Average Precision (mAP50) of over 80% in detecting human presence in CCTV footage. We used a publicly available dataset of the images captured by a CCTV camera and labeled them to identify the person in the images. Our experiments resulted in an mAP of 88.5% on the test set, validating our hypothesis and demonstrating the model's efficacy in accurately identifying human figures and their locations within the monitored scenes. To deploy the model, we prepared a Python script that can take a camera stream or a video feed as input and detect a person's location. Once the person is detected, the location of the person is compared with the safety zone and an alert is generated in case of violation. This study demonstrates that deep learning-based object detection models can help build smart camera devices or software that can help detect people in a scene and avoid workplace injuries before they happen.

INTRODUCTION

Workplace fatalities, injuries, and illnesses cost the country billions of dollars every year. As per the studies, work-related injuries and death cost over 171 billion dollars in 2019 and the compensation cost to the victims is over 1 billion dollars (1). The construction industry is considered one of the most hazardous industries in the world. Because of poor occupational safety and lack of safety regulations and standards in developing countries, the number of accidents is higher (2). By employing safety standards, we can reduce injury claims by 9.4% and save 26% in worker's compensation (3).

The government has tried to introduce new rules around safety standards, but they are still not strictly followed. There has been some research done to create solutions that detect safety equipment (Helmets, shoes, glasses, etc.) (4). In (5), researchers have used Bluetooth Low Energy (BLE) technology to create virtual fencing and detect people in dangerous areas. However, that requires facilities to install physical devices which might create resistance in adoption. In (6), researchers have used YOLO (a deep learning technique for object detection) to create solutions that will detect if a worker is wearing safety equipment like a hard hat and vest and also wearing them so that they are positioned correctly and are able to protect the user maximally. In (7), researchers have used motion, EEG, and EMG sensors with deep learning techniques to find if workers are holding the push and pull carts safely. These sensors might not be cost-effective and feasible for every worker to wear. In (8), researchers used the YOLOv3 object detection technique to detect if the workers were wearing helmets. Researchers have also tried a proactive approach to train workers in a simulated environment while assessing their behavior (9). There are many such researches happening that are focused on detecting the safety equipment. However, very little work has been done to enforce the virtual barricades and detect if the person is in a dangerous zone, and stop them before they get injured.

We hypothesized that the YOLOv8 model, adapted via transfer learning, would achieve a mean Average Precision (mAP50) of over 80% in detecting human presence in CCTV footage. We have used YOLOv8 (You Only Look Once) deep learning architecture and created a custom model to detect people in the scene (10). The model is light enough to be deployed in a portable device and can be used with a smart camera for practical deployment. In this paper, we have presented the results achieved for detecting a person in a scene and also test results after deploying the model in a real-world scenario.

RESULTS

The best YOLOv8 model, with a mAP50 value of 88.5%, worked well in detecting people in a scene (epochs = 15 and learning rate = 0.0005, **Table 1**).

We started the training with 5 epochs and a learning rate of 0.01 and that resulted in an accuracy of 59.3%. Then for the same number of epochs, we varied the learning rate up to 0.0005 and that improved the accuracy drastically (85.1%). Then we decided to increase the number of epochs in the steps of 5 and we saw a significant increase in performance until 15 epochs. After 15 epochs, the performance seemed to saturate and prompted us to stop the training. Also, a mAP50 value of 88.5% was enough to detect a person in the scene and worked well in the real-world scenario testing.

The variation of mAP50 and mAP50-95 over various epochs is presented on the chart (**Figure 1**). The mAP50 value is the mean average precision of the detection when the intersection over union (IoU) threshold is set to 50%, while mAP50-95 measures the mean average precision across IoU thresholds ranging from 0.5 to 0.95.

The final model was tested on a recorded video from a school scene where a kid is not supposed to cross the tape. The detection output (**Figure 2**) shows that the model is working well up to a distance of 15 meters.

DISCUSSION

The YOLOv8 model created in this research is performing well and can detect persons in the scene. We tested the model with a recorded video of a real-world scenario, and the model could identify a person even at a distance of 15 meters.

We started with the idea of adding safety for people in workplaces. One of the major issues that we have is people tend to ignore the warning signs and barricades. So we decided to build a solution that can create a virtual barricade using cameras and alert authorities in case of intrusion. To achieve this, we used a dataset that contains CCTV images from different locations (11). To experiment, we started with an object detection technique called YOLO and built a model that can detect people in a scene. We selected the YOLO model for experimentation because it is fast enough to detect objects almost in real-time. While training, we started with 5 epochs and a learning rate of 0.01 and experimented by increasing epochs in

steps of 5 and reducing the learning rate by making it less than half of the one in the previous training cycle. In the process, we realized that these changes were enough to significantly improve the mAP50 value across the training cycles. Now, using the detection information and where exactly they should stop, we have created a solution that can help find intrusions. The next step was to test the solution in a real work scenario. We tried running the model on a local computer with some technical glitches. Ultimately, we recorded a video of a scene where people were supposed to stop before a barricade. We used the video as an input to the model, and the model detected all the people in it properly. We use the barricade information to create a virtual boundary and our code can alert whenever a person crosses it. The next step is to deploy this solution in a portable computer to use it as a smart camera or it can be installed in command and control centers to detect intrusions in the feed from the existing cameras. Also, the model is currently limited to detecting people, however, in the future, it can be extended to detect more objects like vehicles and animals by extending the dataset and training it again. This way, we can selectively stop things from crossing a certain area.

MATERIALS AND METHODS

Here are some of the tools and techniques that we have used in experimentation and building the solution:

Python packages

Ultralytics: A Python package that deals with training the AI, and with most statistical operations for other parameters.

Roboflow: Helps to find a dataset that is large and convenient for usage. We selected one that was CCTV footage of humans to train the AI to detect humans.

OpenCV (12): Used for loading the video and other image processing in the training and running of the model.

Object Detection Model:

YOLOv8: A Deep learning model that is able to process the data quickly and find any humans that are on the footage being processed.

Machine learning pipeline

Figure 3 shows how the deep learning model is trained. We collected the data and converted it into YOLOV8 format using Roboflow. For training the model we need to split the dataset into the

training, test, and validation sets which was also done using Roboflow. Next, we used Ultralytics (a Python module) to import a version of YOLOv8 (i.e. YOLOv8n) and performed transfer learning to create a custom model that can detect humans. We trained the model using the training dataset by varying the hyperparameters such as epochs and learning rate. We validated the model in each training cycle by using the validation dataset to check if mAP50 > 85%. Once we got a model with expected performance on the validation dataset, we tested it using the test dataset.

Inference code

Figure 4 is a flowchart of the code required to deploy and run the model in real-world scenarios. Once the camera is initialized, we load the final object detection model trained earlier. We pass the recorded video frames into the detection model and it responds with the location of humans in the frames. We have also created a virtual barricade by setting a rectangle in the frame within which no human should be present. Once the detection model detects any human in the frame, we compare the location of humans with the location of the virtual barricade. If there is any overlap in human and barricade locations, then it sends an alert to the authority or outputs a signal. The above process repeats until the device is on.

REFERENCES

1. "Business Case for Safety and Health - Overview." Occupational Safety and Health Administration. www.osha.gov/businesscase. Accessed 5 Apr. 2024.
2. Mouleeswaran, Kumar. "Evaluation of safety performance level of construction firms in and around erode zone." International journal of innovative research in science, engineering and technology 3.1 (2014), vol. 3, no. 1, Jan. 2014, https://www.academia.edu/download/46543490/7_CE106.pdf.
3. "Business Case for Safety and Health - Benefits." Occupational Safety and Health Administration. www.osha.gov/businesscase/benefits. Accessed 5 Apr. 2024.
4. "Personal Protective Equipment - Overview." Occupational Safety and Health Administration. www.osha.gov/personal-protective-equipment. Accessed 16 Apr. 2024.
5. Rey-Merchán, María Del Carmen, et al. "Virtual fence system based on IoT paradigm to prevent occupational accidents in the construction sector." International journal of environmental research and public health 18.13 (2021), vol. 18, no. 13, Jun 2021, <https://doi.org/10.3390/ijerph18136839>.

6. Nath, Nipun D., Amir H. Behzadan, and Stephanie G. Paal. "Deep learning for site safety: Real-time detection of personal protective equipment." *Automation in Construction* 112 (2020), vol. 112, no. 103085, Apr. 2020, <https://doi.org/10.1016/j.autcon.2020.103085>.
7. Vukicevic, Arso M., et al. "Assessment of the handcart pushing and pulling safety by using deep learning 3D pose estimation and IoT force sensors." *Expert Systems with Applications* 183 (2021), vol. 183, no. 115371, Nov. 2021, <https://doi.org/10.1016/j.eswa.2021.115371>.
8. Huang, Li, et al. "Detection algorithm of safety helmet wearing based on deep learning." *Concurrency and Computation: Practice and Experience* 33.13 (2021), vol. 33, no. e6234, Feb 2021, <https://doi.org/10.1002/cpe.6234>.
9. Li, Heng, et al. "Proactive behavior-based safety management for construction safety improvement." *Safety science* 75 (2015), vol. 75, no. 107-117, Jun 2015, <https://doi.org/10.1016/j.ssci.2015.01.013>.
10. "Detect - Ultralytics YOLOv8 Docs." Ultralytics. docs.ultralytics.com/tasks/detect/. Accessed 3 Feb. 2024.
11. "CCTV Object Detection Dataset and Pre-Trained Model by Dataset." Roboflow. universe.roboflow.com/dataset-uutxr/cctv-naxyo. Accessed 20 Dec. 2023.
12. "Introduction to OpenCV." OpenCV. https://docs.opencv.org/4.x/da/df6/tutorial_py_table_of_contents_setup.html. Accessed 2 Feb. 2024.
13. "Google Colaboratory." Google. colab.google. Accessed 20 Dec. 2023.

Figures and Figure Captions

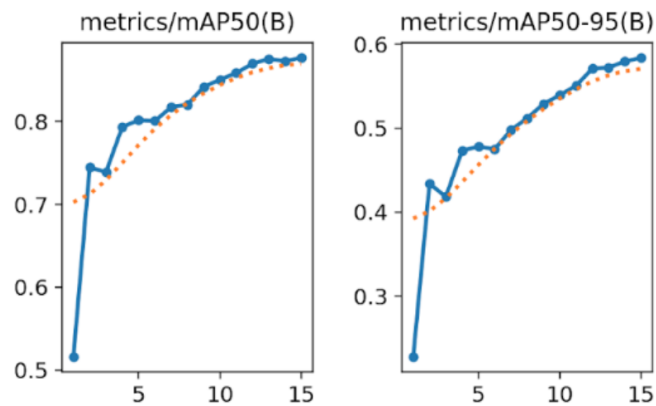


Figure 1. Variation of the performance metrics on validation dataset across various epochs. mAP50(B) graph showing variation of mAP50 across various epochs and mAP50-95(B) graph showing variation of mAP50-95 across various epochs.

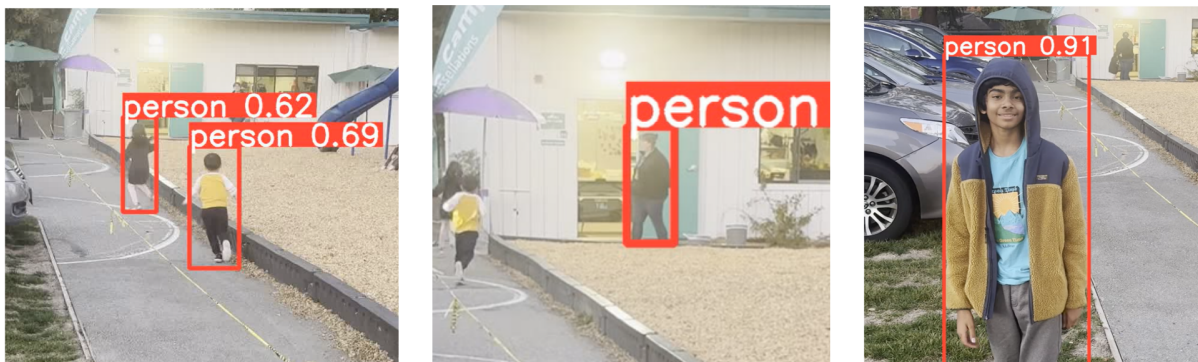


Figure 2. Sample output images from the final model. Images showing how well the final model did in real-world scenarios. An iPhone camera was used to record a video and then the recorded video was passed through the best YOLOv8 model.

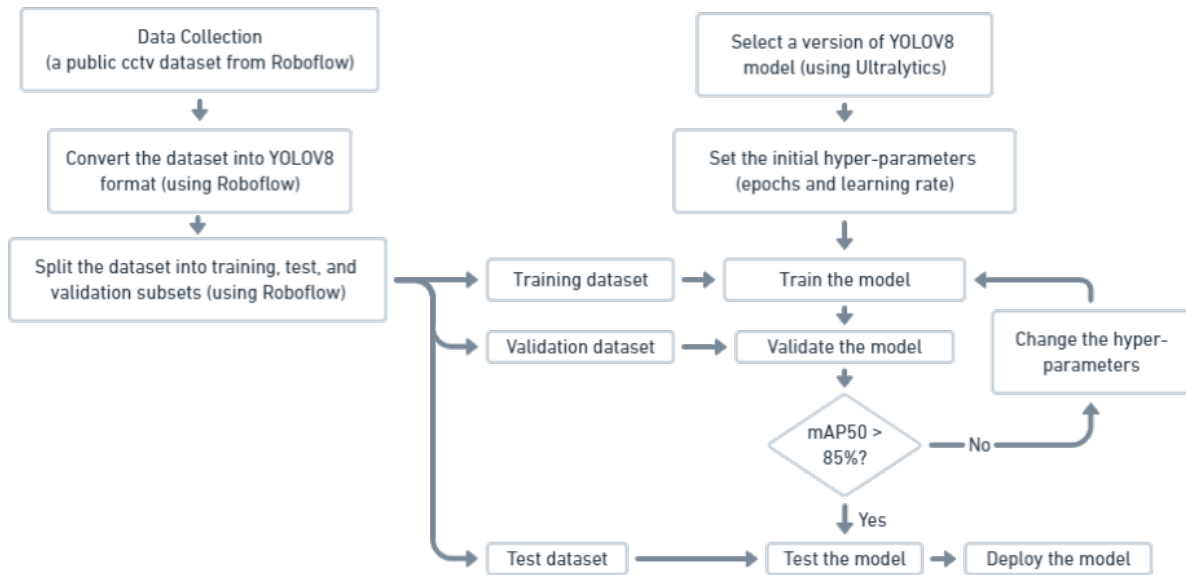


Figure 3. Flowchart showing the pipeline used to train the YOLOv8 model. We split the dataset into the train, validation, and test datasets and used ultralytics for training, validating, and testing the model.

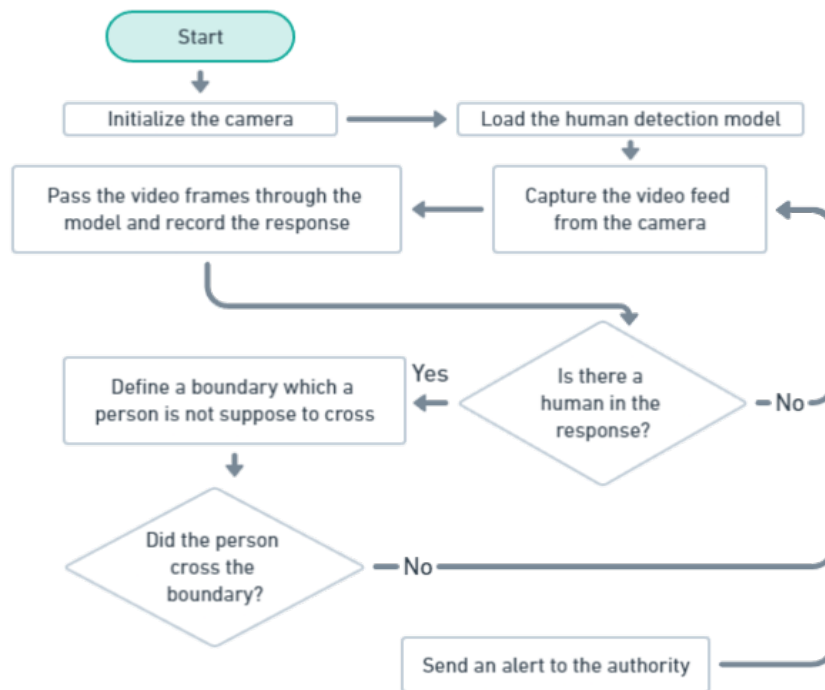


Figure 4. Flowchart showing the inference pipeline required for model usage. The flowchart details the steps required from capturing the images using a camera to detecting people and alerting authorities.

Tables with Captions

Epochs/ Learning Rates	5	10	15	20
0.01	0.593	0.702	0.853	0.798
0.005	0.668	0.785	0.801	0.818
0.001	0.832	0.785	0.861	0.874
0.0005	0.851	0.861	0.885	0.885

Table 1. A learning rate of 0.0005 and 15 epochs gave the best mAP50 score of 0.885 (88.5%).

Table showing mAP50 values for various epoch and learning rate combinations. We ran the training code multiple times with various hyperparameters to get the optimal mAP50.