

# Monitoring System using Artificial Vision and YoloV8 for Accident Prevention in Children

Johnny Angelino<sup>1</sup>, Andrés Quispe<sup>1</sup>, Ricardo Yauri<sup>2</sup>

<sup>1</sup>Facultad de Ingeniería Electrónica y Eléctrica, Universidad Nacional Mayor de San Marcos, Lima, Perú

<sup>2</sup>Facultad de Ingeniería, Universidad Tecnológica del Perú, Lima, Perú

## Article Info

### Article history:

Received Apr 13, 2026

Revised Sep 14, 2026

Accepted Sep 22, 2026

### Keywords:

Artificial Vision

Monitoring

Computer Vision

Raspberry Pi

ESP32

Object Detection

## ABSTRACT

The increase in student density in educational institutions has generated greater risks to child safety, especially in high-traffic scenarios where manual supervision is limited to prevent accidents. In this context, the need arises to implement non-invasive technologies and systems based on artificial vision, to improve monitoring and support traditional surveillance methods. Previous research describes the use of computer vision techniques and models such as YOLO for safety, monitoring, and their integration into low-cost embedded systems, with limitations in real-time response or accident prevention. This research shows the development of a monitoring system based on artificial vision with a YOLOv8 model trained and deployed on a Raspberry Pi with integration with an ESP32 to detect risk events aimed at preventing accidents in minors. The results indicate that risk classes are detected with confidence levels above 70%, with good performance in metrics such as precision (~0.9), recall (~0.8) and mAP50 (~0.9), without overfitting and with hypothesis validation. In conclusion, the system is a suitable solution for accident detection with metric validation, but it needs to be optimized to work in real-world heterogeneous scenarios.

Copyright © 2026 Institute of Advanced Engineering and Science.  
All rights reserved.

## Corresponding Author:

Ricardo Yauri

Facultad de Ingeniería

Universidad Tecnológica del Perú

Cal. Natalio Sanchez Nro. 125, Lima, Perú

Email: c24068@utp.edu.pe, boncer99@gmail.com

## 1. INTRODUCTION

In recent years, the increase in the number of students in educational institutions has generated problems related to child safety. In the case of high-traffic areas [1] such as recess, entrances and exits, these become critical scenarios that require continuous supervision [2]. In this context, the vulnerability of children in accidents or risky situations increases [3], making it necessary to use technologies that complement manual methods [4]. Therefore, computer vision-based systems [5] are an alternative to contribute to real-time monitoring [6]. In addition, manual supervision has limitations inherent to processes carried out in environments with many students and where the lack of personnel makes it difficult to monitor [7] and follow up on critical areas, which generates accidents or violence [8].

In the area of child monitoring, there are solutions that integrate technologies such as the Internet of Things [9], environmental sensors and location technologies [10] to monitor children's activities [11]. Some application aspects consider the use of wearable devices [12] to measure biological variables and detect accidents or falls, while others use systems with alerts and event analysis in safe zones [13]. Furthermore, it must be considered that these technologies must be used with children, which can be invasive. In contrast, computer vision-based systems are used non-intrusively, justifying the implementation of the system in this research.

In the field of child safety, research describes the use of facial recognition and trajectory tracking systems to identify and detect the behavior of minors [14]. Furthermore, it is noted that solutions using surveillance cameras do not allow for an immediate response [15]. Therefore, there is a trend toward using deep learning models to improve the accuracy of detecting and analyzing behavior and activities [16].

On the other hand, detection algorithms like YOLO (You Only Look Once) are used in security applications to detect suspicious people or behavior. Studies indicate that versions like YOLOv5 offer a balance between speed and accuracy [17], for video surveillance applications [18]. Furthermore, the importance of YOLOv8 is also highlighted due to its performance improvements [19] and its adaptability to specific security contexts [20].

Regarding other aspects of object detection in fall detection, pose estimation via skeletons has been employed instead of using bounding boxes [20], [21]. Furthermore, human activity recognition (HAR) techniques utilizing convolutional neural networks (CNNs) or temporal models (LSTM or Transformer) have been used to detect high-risk activities [22]. Despite the advancement of these technological tools, there is little integration into educational settings for accident prevention, and most applications focus on public or industrial safety.

In the case of hardware implementation, evaluations have been carried out on embedded systems such as Raspberry Pi [23] to integrate machine vision models, considering their low cost [24], and also the integration of cameras and sensors for monitoring [25]. However, there are processing limitations, which have led to the optimization of models for their execution. On the other hand, it is possible to combine embedded devices with microcontrollers [26] such as the ESP32 for alert management [27] and network communication, considering distributed systems [28].

This research proposes the development of a monitoring system using computer vision and the YOLOv8 model for accident prevention among children in educational institutions. Data collection and preprocessing are carried out, along with the training of a neural network in GPU-accelerated environments and its implementation on a Raspberry Pi. Furthermore, communication with an ESP32 microcontroller is integrated for triggering alerts and notifications. This involves activities that determine the system's characteristics, as well as the implementation and execution of computer vision models in an embedded system.

The motivation for this research is the need to protect children's safety in educational settings due to limited human supervision. The contribution lies in the design of a system using computer vision and embedded computing to detect risk events and integrate automatic alerts for critical problems. Its value lies in the application of real-time detection technologies.

## 2. RESEARCH METHOD

The proposed system utilizes YOLOv8 technology, which is used in other research due to its effectiveness in object detection [29] and classification tasks [30]. This technology allows for the creation of datasets and the training of models using an online environment such as Google Colab with GPU acceleration. The process begins with dataset preparation, followed by preprocessing and feature extraction. Finally, the model is trained using Python [31] and then imported onto a Raspberry Pi 5 [32]. Based on the alert configurations, signals are sent to an ESP32 microcontroller to activate alarm actuators [33], which is efficient for wireless connectivity tasks [34]. Additionally, a notification stage was implemented for mobile or computer devices within a local area network (LAN) (Figure 1).

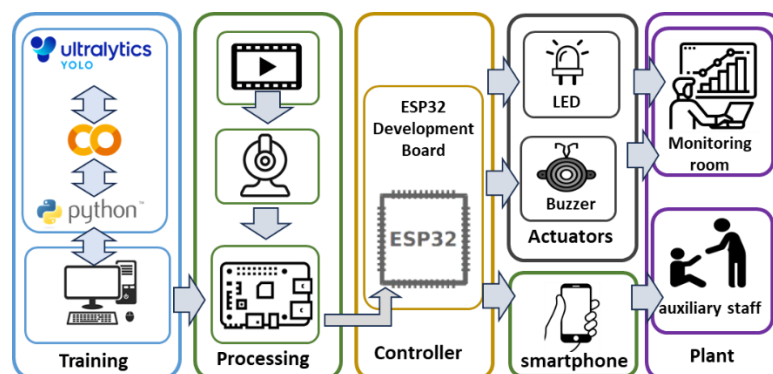


Figure 1. Definition of the project components

The development process is represented in **Error! Reference source not found.**, which divides it into three parts. First, the initial parameters are analyzed to establish the functional requirements, followed by

the conceptual design. Second, the datasets are created, and the neural network model is trained for integration with the hardware. Finally, the system is tested, and the model's performance is analyzed.

### 2.1. Machine vision model training

Figure 2 shows the process of creating the dataset using data science platforms (Kaggle, MendeleyData, ImageNet, etc.). Manually assigning labels to each image is necessary, for which the Roboflow platform is used. Subsequently, the dataset is imported from Google Colab, and training parameters such as epochs, learning rate, and image size are adjusted. Finally, the model file (.pt) is generated for integration into a Python file, considering variations of the YOLO architecture such as nano or small.

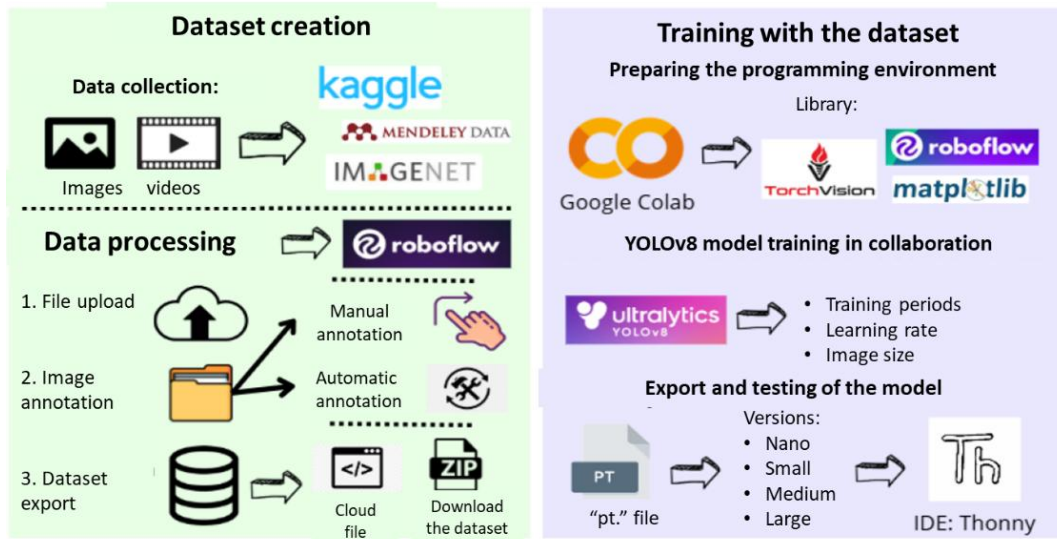


Figure 2. Dataset creation and training

For data processing in Roboflow, the datasets to be used are defined. Table 1, shows the information sources necessary for processing and classifying the data according to the project requirements and the classes to be detected.

Table 1. Classification of data source

| Name of the data source                    | Origin            | Number of images |
|--------------------------------------------|-------------------|------------------|
| Fall Detection Computer Vision Project     | Roboflow Universe | 410              |
| Dangerous Computer Vision Project          | Roboflow Universe | 7561             |
| Knife Dataset New Computer Vision Project  | Roboflow Universe | 4075             |
| Climbing Detection Computer Vision Project | Roboflow Universe | 512              |
| Shanghai Tech                              | Kaggle            | 1198             |
| CrowdHuman Crowd Detection CSV Labels      | Kaggle            | 12000            |
| Dangerous situation detection              | Mendeley Data     | 7000             |

On the Roboflow platform, each image is classified and labeled. The following classes are defined in this process: Fall (473 images), Crowd (4335), Weapon (662), Fire (3883), and Fall Risk (598). This class distribution exhibits an imbalance; consequently, data augmentation techniques (such as rotation and brightness adjustment) were applied to the 'Fall' and 'Fall Risk' classes to increase their sample size. The results for all classes are shown in Table 2, which presents the class distribution before and after augmentation.

Table 2. Class distribution

| Class     | Original images | Augmented images | Weight factor |
|-----------|-----------------|------------------|---------------|
| Fall      | 473             | 1,200            | 2.5           |
| Crowd     | 4,335           | 4,335            | 1.0           |
| Weapon    | 662             | 1,300            | 1.9           |
| Fire      | 3,883           | 3,883            | 1.0           |
| Fall Risk | 598             | 1,450            | 2.4           |

Google Colab with its integrated graphics processing engine based on the Tesla T4 GPU is used for training. The Python and Ultralytics libraries are then installed and their functionality tested using the official

Yolov8 functions (8.2.103). The Yolo command is executed for training, validation, and inference. The number of epochs chosen is 60 to ensure optimal learning. Furthermore, the image dimensions are 640x640.

## 2.2. Implementing the program on Raspberry Pi 5

Figure 3 defines the communication modules of the Raspberry Pi 5 module with a computer to transfer the detection program developed in Python. Images are acquired for processing via the camera using a Wi-Fi, UART, or USB 3.0 connection. Depending on the detected event, the visual and audible actuators (buzzer and LED) are activated by communicating with an ESP32 module using the UART or I2C serial communication protocol. The model generated based on the YoloV8 architecture allows inference to be performed through the capture and processing of images by OpenCV.

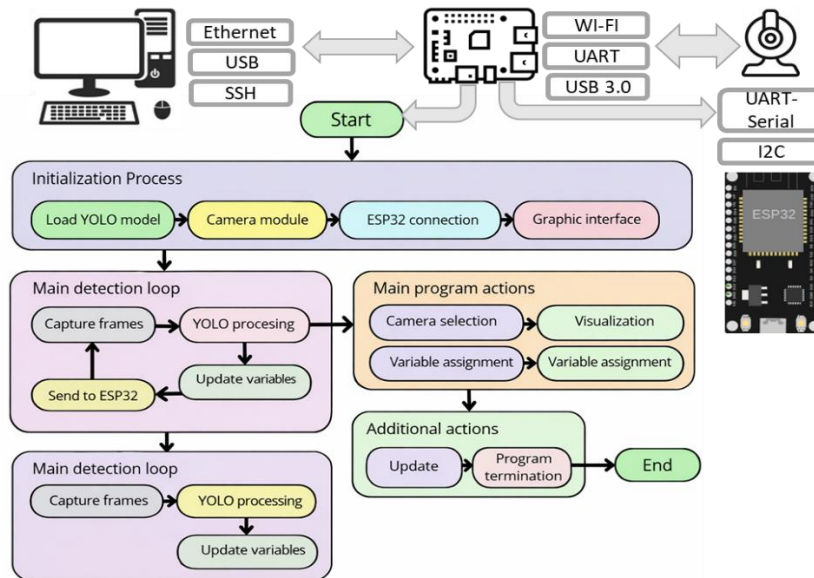


Figure 3. Flowchart of the program embedded in the Raspberry Pi.

The algorithm is implemented using three Python files (camera, gpio\_manager, and main) that utilize the trained model “best.pt”. This integration is achieved through the “ultralytics” library and OpenCV, which enables calculations in Python and the determination of the coordinates of the object identifier frames. Additionally, the “RPi.GPIO” library allows for the enabling and management of the Raspberry Pi 5's GPIO pins. In the case of the “camera.py” file, it serves to enable access to these pins. Finally, the “detect\_objects” function is used for object detection in each frame. This function uses the “model” parameter, the “frame” parameter, and the “results” variable, which stores information about the bounding box parameters and the detected class (Figure 4).

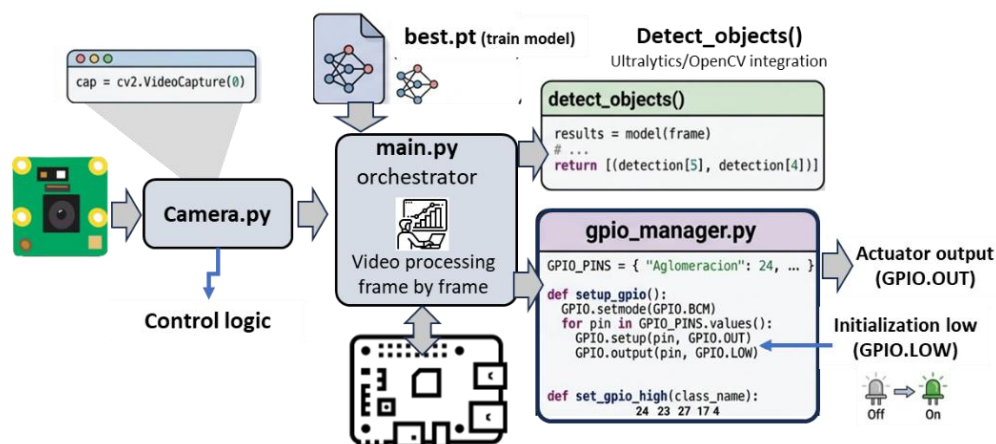


Figure 4. Functions generated for detection

According to the classification labels assigned on the Raspberry Pi 5, these are assigned to 5 GPIO pins (Crowd, GPIO 24; Weapon, GPIO 23; Drop, GPIO 27; Fire, GPIO 17; Drop\_Risk, GPIO 4). Additionally, pins 4 and 6 of the Raspberry Pi 5 are used as power supplies for the ESP32 module. The design for transmitting detections must address three requirements, including latency when the ESP32 detects a voltage level change. Additionally, it is necessary to avoid burdening the ESP32 with interpreting data packets; this reduces firmware complexity and eliminates the need for data processing.

### 2.3. Control algorithm in ESP32

The ESP32 module's algorithm implements data reception via GPIO and response transmission via Wi-Fi. The "ESPAsyncWebServer" library, capable of processing multiple web requests, is used to manage the connection between the ESP32 and the Wi-Fi network. The alerting process begins by reading the input pins, assigning values from 1 to 5 based on the detected pin activation level for each of the five classes. For each case, the following input pins are referenced: 13, 12, 14, 27, and 26 (Figure 5).

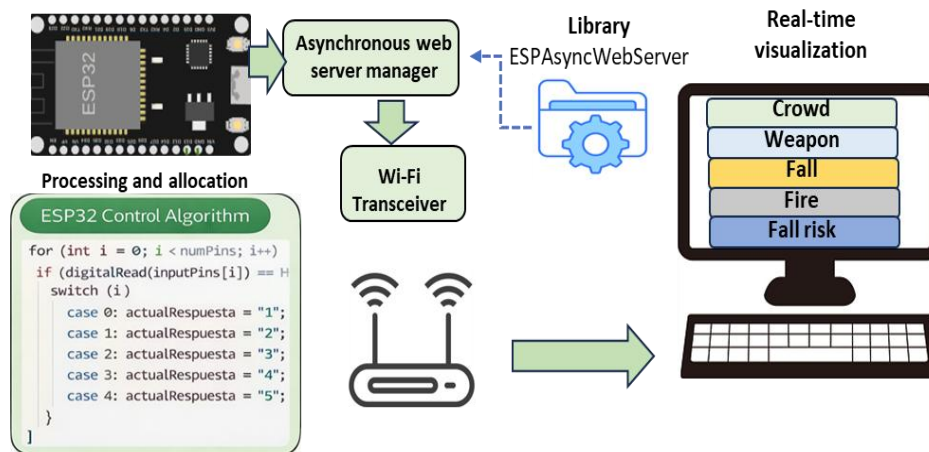


Figure 5. Pin assignment loop for indicator activation

### 2.4. User interface

The alert is sent via a wireless connection between the ESP32 within a LAN and a mobile application that receives the detection data. This application was created using the "App Inventor" tool. Its functionality begins with configuring ESP32's IP address, followed by using the "Activate Monitoring" button to receive the data. Depending on the type of event detected, a reference image is displayed. Additionally, an alarm sound and phone vibration were implemented. The interface components include the "Wi-Fi Connection" method, the URL retrieval process, and the "Activate" button for receiving alerts. The application then establishes the following conditions: "No Hazard," "Crowd," "Weapon," "Fall," "Fire," "Risk of Falling," and "No Connection" to configure the displayed image. In all cases, except for "No Hazard" and "No Connection," the phone's vibration and sound functionalities were activated (Figure 6).



Figure 6. Activation block for receiving and configuring vibration and visual alarm

### 3. RESULTS AND DISCUSSION

#### 3.1. Model testing

A wireless router was used to connect the Raspberry Pi and the ESP32 to verify the results. The mobile device connected to the router to access the server created on the ESP32 module via the mobile application. Fire and crowd detection tests were performed using a webcam connected to the Raspberry Pi 5 via USB. In this case, the confidence levels obtained did not fall below 70% ( Figure 7).

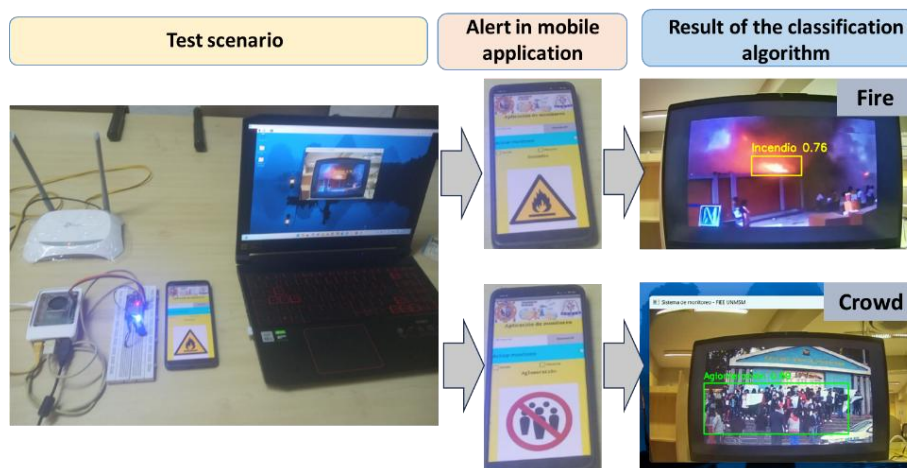


Figure 7. Classification of the “fire” and “crowd” classes.

For the tests conducted in an open environment, the following classes were used: Risk of Fall, Fall, and Weapon. In this case, a PTZ camera connected via Wi-Fi was used. The tests performed with the "weapon" class were conducted at distances between 3 and 9 meters. Beyond 6 meters, the confidence levels decreased. This demonstrates the effectiveness of the "weapon" class detection at a range of up to 6 meters. Furthermore, the confidence level does not fall below 70% even at the extreme distance of 9 meters (Figure 8).

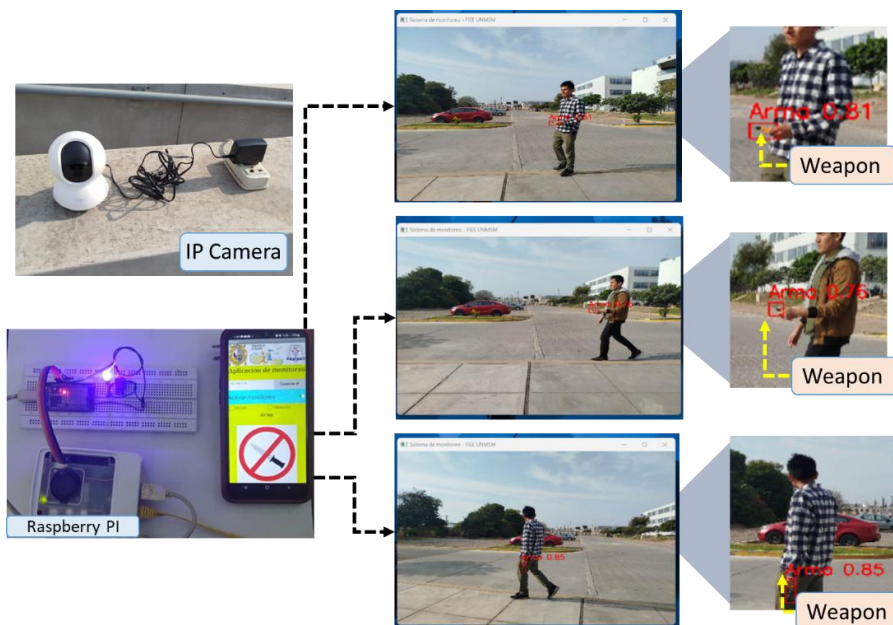


Figure 8. View of the local (LED), remote (cellular) alarm and Raspberry Pi interface for weapon detection

Subsequently, tests of the "fall" class are carried out, considering distances of up to at least 10 meters, obtaining with a confidence level greater than 80% (Figure 9).

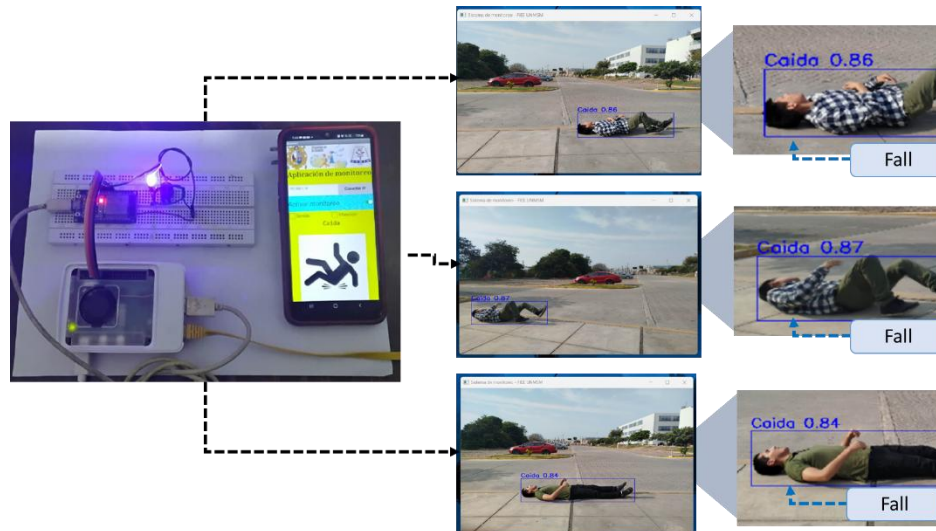


Figure 9. View of the local (LED), remote (cellular) alarm and interface for fall detection

In the case of the “Risk of falling” class, detection is observed in valid scenarios with a confidence level greater than 80% (Figure 10)

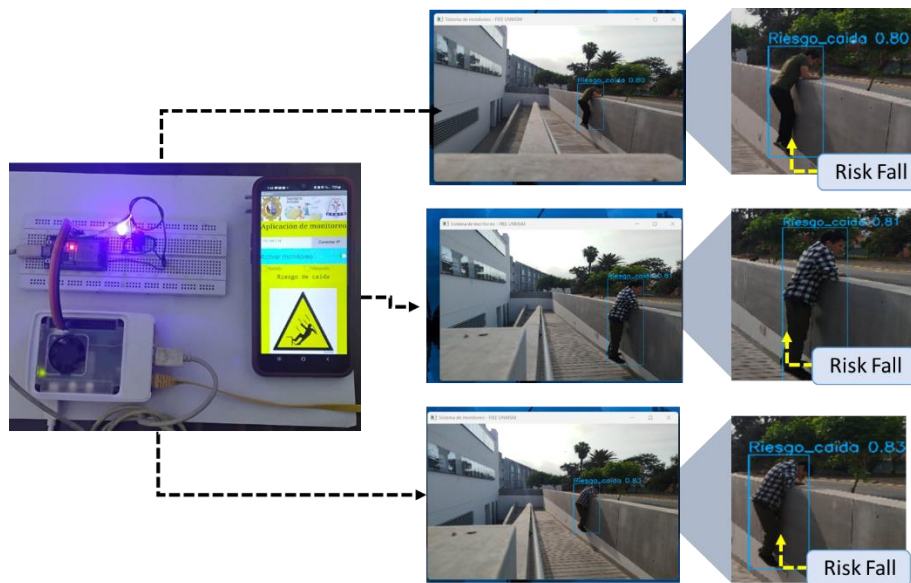


Figure 10. View of the local (LED), remote (cellular) alarm and Raspberry Pi interface for fall risk

### 3.2. Metrics analysis

Based on the confusion matrix obtained from the detection model's behavior, the "Weapon" class showed the best balance between accuracy (0.911) and recall (0.935), indicating that the model correctly identifies most real-world cases and makes few errors when predicting this class. Similarly, the "Fall Risk" class had high recall (0.967), although with reduced accuracy due to some false positives. On the other hand, the "Crowd" class performed well, with accuracy (0.901) and recall (0.872) values, with some losses associated with background confusion.

The 'Fall' class achieves adequate performance with an accuracy of 0.815 and a recall of 0.853, suggesting that the model can identify this type of event but still presents difficulties when postures are indistinguishable. Finally, the "Fire" class has lower accuracy values (0.791) and a recall of 0.836, showing a higher rate of confusion, related to the visual variability of the fire. Therefore, the results indicate that the model performs adequately with critical classes such as "weapon" and "risk of falling" (Figure 11).

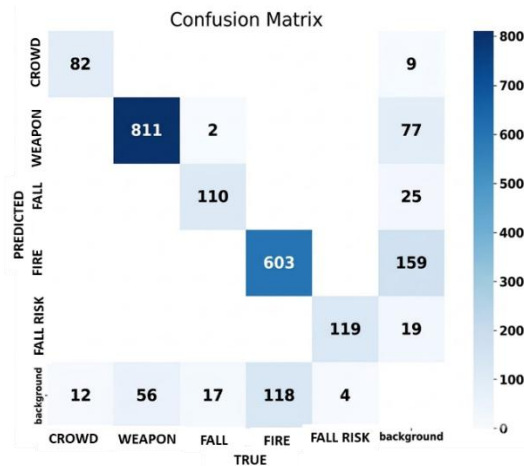


Figure 11. Training confusion matrix

Figure 12 shows the model's training evolution over 60 epochs, considering loss functions and performance metrics. The indicators train/box\_loss, train/cls\_loss, and train/df\_loss represent the error in bounding box localization, classification error, and edge regression loss using Distribution Focal Loss (DFL). Additionally, the metrics precision (B), recall (B), mAP50 (B), and mAP50-95 (B) consider accuracy, sensitivity, and precision. Losses during training decreased across all three metrics. Furthermore, box\_loss decreased from 1.7 to 1.0, indicating improved localization. On the other hand, cls\_loss decreased from 3.0 to 0.5, indicating that the model was learning to differentiate between the various classes. In the case of the validation metrics (val/box\_loss, val/cls\_loss and val/df\_loss), there is a positive generalization behavior of the model.

Performance metrics, such as accuracy, increased from 0.4 to 0.9, demonstrating a reduction in false positives during training. Recall stabilized at 0.8, indicating that the model detects most objects. The mAP50 value increased to 0.9, while the mAP50-95 value reached 0.6.

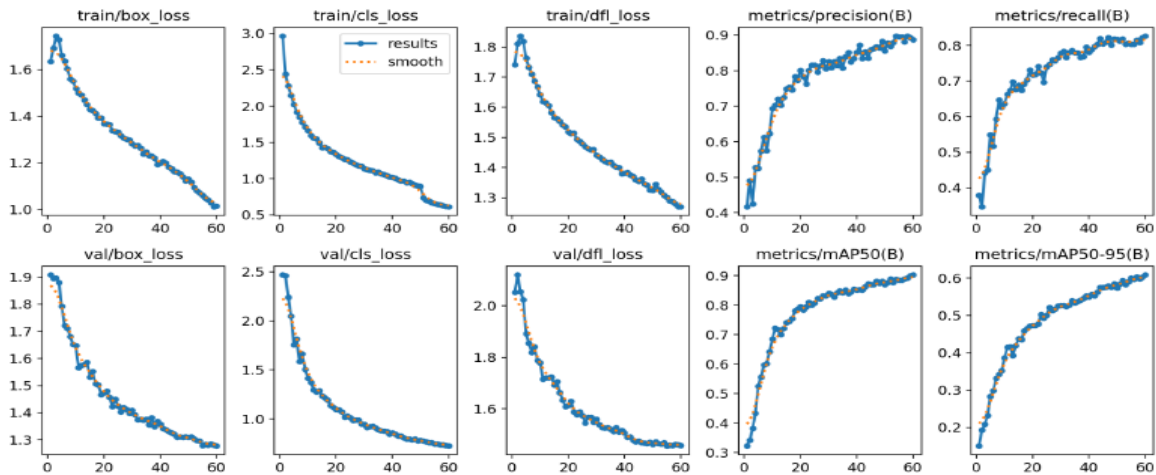


Figure 12. Metrics of the YoloV8 model training.

Regarding the mAP50-95 value of 0.6 it is determined that the model accurately detects object locations. To evaluate system performance, its integration into a Raspberry Pi 5 edge device is assessed by measuring latency and hardware resource consumption. Furthermore, the system operates continuously at 11.2 FPS, a rate that could be improved in future work to enhance the monitoring of high-risk events.

Table 3. deployment performance on Raspberry Pi

| Metric                                     | Value    |
|--------------------------------------------|----------|
| Inference latency (milliseconds per frame) | 89 ms    |
| Throughput                                 | 11.2 FPS |
| CPU utilization (avg.)                     | 78%      |
| RAM usage                                  | 1.451 GB |
| Peak SoC temperature (30 min load)         | 61 °C    |

### 3.3. Hypothesis testing

To perform the hypothesis test, the confidence level obtained in the classification processes was evaluated. For this test, images with a system confidence level greater than 70% were selected. Subsequently, the number of correctly and incorrectly classified images was tallied. The hypothesis test analysis determined whether the proportion of false positives was low enough to validate the accuracy. The calculations were performed using Minitab software, considering that the null hypothesis ( $H_0$ ) had a p-value of 0.7, indicating a false positive rate less than or equal to the critical threshold. Furthermore, for the alternative hypothesis ( $H_a$ ), a p-value greater than 0.7 was used for the false positive rate when it was greater than the critical threshold, with a significance level of  $\alpha = 0.05$  (Figure 13).

The tests performed treat detection as a binary result (classified above the 70% threshold) and assess whether the detections exceeding that threshold meet an acceptable false-positive rate. Consequently, this hypothesis test does not evaluate the statistical validity of individual confidence scores; rather, it uses the detection result itself as the basis for validation.

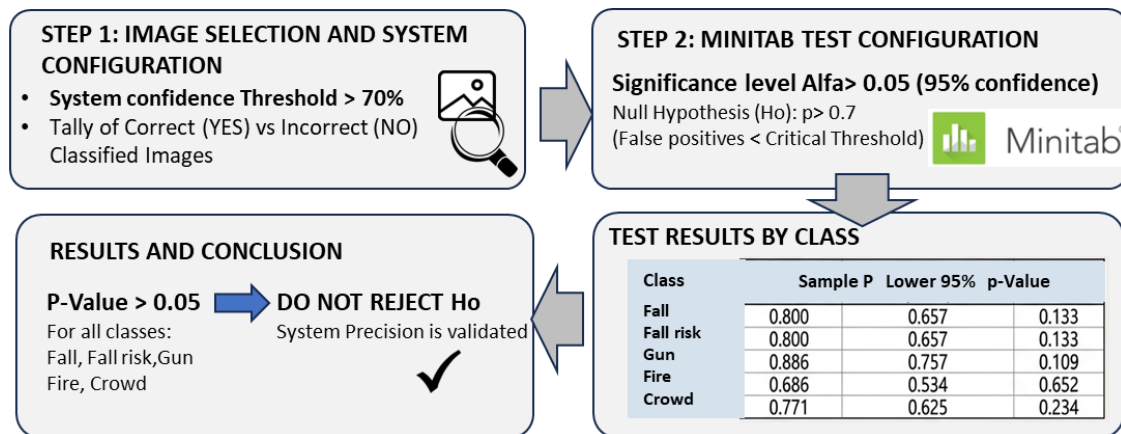


Figure 13. Hypothesis testing

The statistical software Minitab performs the statistical calculations for the hypothesis test, using the risk classes, observing that the p-value is always greater than  $\alpha$  (significance level = 0.05). This implies that the null hypothesis cannot be rejected, and that there is insufficient evidence to conclude that a confidence level lower than 70% is reliable. Therefore, it is concluded that a confidence level greater than 70% yields reliable results for the detection model.

### 3.4. Discussion of results

The results show that the monitoring system based on machine vision and the YoloV8 architecture successfully identified risk situations, such as falls, impacts, or dangerous behavior. However, it is important to note that the system's efficiency depends on environmental conditions, such as lighting, viewing angle, distance, and movement. In situations where these variables change, the system showed a decrease in the detection rate, indicating an area for future improvements.

Compared to other traditional monitoring systems (such as passive surveillance cameras or manual supervision), the system showed a greater ability to react because the use of artificial intelligence is a more effective tool for accident prevention. During the testing phase, some false positives were obtained, but with a confidence threshold above 70%. This result indicates the importance of establishing thresholds to issue alerts only in dangerous situations.

Since priority was given to integrating the system across remote components rather than exploring different architectures, a direct comparison of YOLO detection models was not required. Quantifying and comparing performance metrics at the edge remains an area for future research. The primary focus of this research is to validate the technological feasibility and integration of the detection system with the embedded hardware through pilot tests. The risk events the system aims to identify (such as falls or crowd clustering) are defined by generic visual patterns or changes in body posture common across different age groups, allowing for the extrapolation of the system's performance. Validation using other datasets (subject to ethical authorization protocols) is identified as future work.

Regarding the limitations of the research, the investigation was conducted in a simulated environment, which influences the effectiveness of incident detection. Furthermore, student behavior and movement patterns can be more controlled and predictable, facilitating detection. Additionally, the simulated environment configuration may not accurately reflect real-world conditions, student mobility, or the presence of obstacles.

#### 4. CONCLUSION

The research successfully implemented a monitoring system based on the YOLOv8 model for preventing accidents involving children in educational settings. This involved compiling and training the model and integrating it into an embedded system. System validation demonstrated its ability to detect risk situations, complementing manual supervision.

Regarding the results obtained, the detection of risk classes showed confidence levels exceeding 70%. Furthermore, there was a progressive decrease in losses and improvements in metrics such as accuracy (down to 0.9), recall (0.8), and mAP50 (0.9), indicating learning without overfitting. Additionally, field tests demonstrated detection at distances, proving the system's ability to identify falls, fall hazards, and dangerous objects. On the other hand, statistical analysis with hypothesis testing confirmed that the confidence threshold minimizes false positives. However, it was observed that factors such as lighting, viewing angle, and distance influence performance.

Finally, among the study's limitations are the use of simulated environments for testing and the hardware constraints that affect system performance. Future work could include expanding the dataset with specific images from real school environments and improving the model's performance under varying conditions by integrating specialized hardware to accelerate inference. Additionally, the system could be evaluated in real-world scenarios and integrated with IoT technologies.

#### REFERENCES

- [1] P. Liu and E. Dastbaravardeh, "Deep Learning-Driven Assessment of Student Movement and Performance Using Physiological Data in Physical Education Information Systems: An S-AIoT Solution," *Int. J. Intell. Syst.*, vol. 2025, no. 1, p. 10, Jan. 2025, doi: 10.1155/int/9479311.
- [2] M.-A. Chung, S.-Y. Chai, M.-C. Hsieh, C.-W. Lin, K.-X. Chen, S.-J. Huang, and J.-H. Zhang, "YOLO-LSD: A Lightweight Object Detection Model for Small Targets at Long Distances to Secure Pedestrian Safety," *IEEE Access*, vol. 13, no. 1, pp. 83061–83070, 2025, doi: 10.1109/ACCESS.2025.3567843.
- [3] A. H. Zinati, M. Sabeti, H. Kamyab, and E. Moradi, "Diagnosis of different types of single suture craniosynostosis in infants with deep learning techniques," *Med. Nov. Technol. Devices*, vol. 27, no. 1, p. 100370, Sep. 2025, doi: 10.1016/j.medntd.2025.100370.
- [4] K. Choopun, N. Pornsawan, K. Panyawong, and S. Wattanakul, "A Mobile ADHD Screening App for Primary School Children: A Pilot Study in Thai Primary Care," *Int. J. Interact. Mob. Technol.*, vol. 19, no. 23, pp. 50–67, Dec. 2025, doi: 10.3991/ijim.v19i23.56723.
- [5] Risfendra, Aripriharta, Suherman, G. F. Ananda, D. S. Putra, and Fahmi, "Contactless Infant Height Measurement for Enhanced Early Detection of Stunting Using Computer Vision Techniques," *IEEE Access*, vol. 13, no. 1, pp. 42364–42376, 2025, doi: 10.1109/ACCESS.2025.3548159.
- [6] S. G. Hong, S. M. Park, S. Kwon, H. Sakthivel, S. P. Nagappa, J. W. Leem, S. R. Steinhubl, P. Ngruwonsanga, J.-L. N. Mangara, C. Twizere, and Y. L. Kim, "Smartphone conjunctiva photography for malaria risk stratification in asymptomatic school age children," *npj Digit. Med.*, vol. 8, no. 1, p. 151, Mar. 2025, doi: 10.1038/s41746-025-01548-8.
- [7] Y. Shang, Y. Shang, and Y. Shang, "Enabled Intelligent Monitoring of Mental Health Indicators During Physical Activity Among Jiangsu Vocational College Students," *Eng. Reports*, vol. 8, no. 2, Feb. 2026, doi: 10.1002/eng2.70612.
- [8] Q. Lai, "Research on the refinement of college student education management based on artificial intelligence," *Discov. Artif. Intell.*, vol. 6, no. 1, p. 31, Dec. 2025, doi: 10.1007/s44163-025-00651-9.
- [9] K. Gangaraju, Y. H. Kadegowda, and S. Sangani, "Energy- and behavior-aware sensory data collection for ASD monitoring over IoT using unequal clustering and reinforcement learning," *Discov. Internet Things*, vol. 6, no. 1, p. 24, Feb. 2026, doi: 10.1007/s43926-026-00287-w.
- [10] L. Han, X. Ma, M. Dai, and L. Bai, "A WAD-YOLOv8-based method for classroom student behavior detection," *Sci. Rep.*, vol. 15, no. 1, p. 9655, Mar. 2025, doi: 10.1038/s41598-025-87661-w.
- [11] M. Alotaibi, A. Alruwaythi, S. Alenazi, and M. Alsaedi, "Integrating YOLOv8 and IoT in a Computer Vision System for Child Detection in Smart Cities," *Int. J. Adv. Comput. Sci. Appl.*, vol. 16, no. 9, pp. 636–644, Sep. 2025, doi: 10.14569/IJACSA.2025.0160960.
- [12] A. Guerrero-Blázquez, Á. C. Álvarez-Melcón, J. J. López-Marcos, P. Martín-Casas, A. Arranz-Escudero, and R. M. Ortiz-Gutiérrez, "Validity and Reliability of a Smart Band for Monitoring Cardiorespiratory Parameters in Children and Adolescents with Severe Cerebral Palsy," *Sensors*, vol. 26, no. 3, p. 828, Jan. 2026, doi: 10.3390/s26030828.
- [13] N. R. Wagh, S. R. Sutar, V. J. Kadam, S. M. Jadhav, A. S. Yadav, and V. S. Pawar, "Evaluation of IoT based smart safety systems for women and children using machine learning techniques," *Sci. Rep.*, vol. 16, no. 1, p. 87, Nov. 2025, doi: 10.1038/s41598-025-29146-4.
- [14] B. J. Tantra and M. Widjaja, "Automatic detection of dress-code surveillance in a university using YOLO algorithm," *IAES Int. J. Artif. Intell.*, vol. 14, no. 2, p. 1568, Apr. 2025, doi: 10.11591/ijai.v14.i2.pp1568-1575.
- [15] S. Peng, X. Zhang, L. Zhou, and P. Wang, "YOLO-CBD: Classroom Behavior Detection Method Based on Behavior Feature Extraction and Aggregation," *Sensors*, vol. 25, no. 10, p. 3073, May 2025, doi: 10.3390/s25103073.
- [16] Y. Cao, Q. Cao, C. Qian, and D. Chen, "YOLO-AMM: A Real-Time Classroom Behavior Detection Algorithm Based on Multi-Dimensional Feature Optimization," *Sensors*, vol. 25, no. 4, p. 1142, Feb. 2025, doi: 10.3390/s25041142.

- [17] Ü. Işıkdag, H. A. Çemrek, S. Sönmez, Y. Aydın, G. Bekdaş, and Z. W. Geem, "A Real-Time Advisory Tool for Supporting the Use of Helmets in Construction Sites," *Information*, vol. 16, no. 10, p. 824, Sep. 2025, doi: 10.3390/info16100824.
- [18] S. Diop, F. Jouen, J. Bergounioux, and I. Trabelsi, "Fine-Tuned YOLO Model for Monitoring Children Across Medical Scenes Based on a Large-Scale Real-World Dataset for Children Detection," *IEEE Access*, vol. 13, pp. 130953–130962, 2025, doi: 10.1109/ACCESS.2025.3588316.
- [19] M. Vijayalakshmi, A. J. R. Jose, G. K. Vishal, and N. Vishwanath, "Safety equipment detection using YOLOv8," *Signal, Image Video Process.*, vol. 19, no. 12, p. 994, Dec. 2025, doi: 10.1007/s11760-025-04583-w.
- [20] I. Vernikos and E. Spyrou, "Skeleton Reconstruction Using Generative Adversarial Networks for Human Activity Recognition Under Occlusion," *Sensors*, vol. 25, no. 5, p. 1567, Mar. 2025, doi: 10.3390/s25051567.
- [21] V. Sanikommu, S. Mummaneni, N. Jacob, E. K.C, B. Kumar, and R. Variam, "Deep Learning-Based Control System for Context-Aware Surveillance Using Skeleton Sequences from IP and Drone Camera video," *Int. Arab J. Inf. Technol.*, vol. 22, no. 5, pp. 919–929, Sep. 2025, doi: 10.34028/iajit/22/5/6.
- [22] E. Oliosi, S. Ferreira, A. P. Giordano, G. Viveiros, J. Parraca, P. Pereira, F. Guede-Fernández, and S. Azevedo, "Evaluation of Smartphone Camera Positioning on Artificial Intelligence Pose Estimation Accuracy for Exercise Detection: Observational Study," *JMIR mHealth uHealth*, vol. 14, no. 1, pp. e82412–e82412, Mar. 2026, doi: 10.2196/82412.
- [23] A. Alvarez Oviedo, J. F. Mamani Villanueva, G. A. Echaiz Espinoza, J. M. M. Villanueva, A. O. Salazar, and E. R. L. Villarreal, "Design of a System for Driver Drowsiness Detection and Seat Belt Monitoring Using Raspberry Pi 4 and Arduino Nano," *Designs*, vol. 9, no. 1, p. 11, Jan. 2025, doi: 10.3390/designs9010011.
- [24] S. Liawatimena and N. Isworo, "Annotated drowsiness detection dataset captured using Raspberry Pi 5," *Data Br.*, vol. 63, no. 1, p. 112211, Dec. 2025, doi: 10.1016/j.dib.2025.112211.
- [25] N. Tahiramani, P. Ahir, S. Saxena, V. P. Talreja, and P. Charanarur, "Edge-based AI solution for enhancing urban safety: helmet compliance monitoring with YOLOv9 on Raspberry Pi," *Discov. Internet Things*, vol. 5, no. 1, p. 25, Mar. 2025, doi: 10.1007/s43926-025-00113-9.
- [26] H. Bodkhe, C. Bilade, D. Naik, O. Deshmukh, A. Bulakh, P. Potdar, K. Shirbavikar, and S. Komble, "Track-Me-Down Emergency Location Service Provider," in *RAiSE-2023*, Basel Switzerland: MDPI, Feb. 2024, p. 235. doi: 10.3390/engproc2023059235.
- [27] R. Yauri, A. Castro, and R. Espino, "Automatic Leaf Health Monitoring with an IoT Camera System based on Computer Vision and Segmentation for Disease Detection," *WSEAS Trans. Electron.*, vol. 15, no. 1, pp. 148–156, Dec. 2024, doi: 10.37394/232017.2024.15.17.
- [28] D. C. Guevara Tocto, D. N. Ayerve Cari, and J. T. S., "Autonomous Visual Pest Detection System with ESP32-CAM, Zonal Classification, and Notification via Telegram for Agriculture," *Int. J. Electr. Electron. Eng.*, vol. 13, no. 1, pp. 60–68, Jan. 2026, doi: 10.14445/23488379/IJEEE-V13I1P107.
- [29] X. Cao, C. Li, and H. Zhai, "YOLO-AB: A Fusion Algorithm for the Elders' Falling and Smoking Behavior Detection Based on Improved YOLOv8," *Comput. Mater. Contin.*, vol. 83, no. 3, pp. 5487–5515, 2025, doi: 10.32604/cmc.2025.061823.
- [30] N. Khalil, M. H. Rashid, S. A. Rehman, M. Yasir, and M. Hussain, "Improved Traffic Safety with YOLO-v8 Driven Smart Helmet Detection," *VFAST Trans. Softw. Eng.*, vol. 13, no. 3, pp. 29–42, Aug. 2025, doi: 10.21015/vtse.v13i3.2184.
- [31] S. S. F. A. Elnozahy, S. C. Pari, and L. C. Liang, "Raspberry Pi-Based Face Recognition Door Lock System," *Internet of Things*, vol. 6, no. 2, Jun. 2025, doi: 10.3390/IOT6020031.
- [32] H. Ma, D. Guo, J. Wang, P. Li, C. Xu, and Y. Wang, "Real-time face recognition algorithm based on lightweight neural network in the field of computer vision," *Discov. Appl. Sci.*, vol. 8, no. 1, p. 33, Nov. 2025, doi: 10.1007/s42452-025-08004-y.
- [33] R. Gholami, H. D. Jahromi, and S. Sedaghat, "Design and Implementation of a Near Real-Time Human Detection Robot Using YOLO Framework and IoT Technologies," *IEEE Access*, vol. 13, no. 1, pp. 175960–175983, 2025, doi: 10.1109/ACCESS.2025.3618302.
- [34] D. Díaz-Delgado, S. M. Vigil-Ramírez, L. J. Acho-Cachay, B. R. Tuanama-Chávez, and L. A. Rojas-Puerta, "IoT-Based Smart Lock with Real-Time Person Detection Using YOLOv5 and Mobile App Integration," *Rev. Científica Sist. e Informática*, vol. 5, no. 2, p. e1005, Jul. 2025, doi: 10.51252/rcsi.v5i2.1005.

## BIOGRAPHY OF AUTHORS



**Ricardo Yauri**    is a PhD of Systems Engineering. He is associate professor at the Universidad Nacional Mayor de San Marcos (UNMSM). He is professor at Universidad Tecnológica del Perú and Universidad Peruana del Norte. He has participated as a teacher in courses oriented to the Internet of things and applications in home automation and the Cisco academy for IoT. He was a researcher at INICTEL-UNI in the Embedded Systems and Internet of Things research group. He developed research projects on the implementation of low consumption IoT devices that involve inference techniques, machine learning algorithms. He can be contacted at email: c24068@utp.edu.pe, ryaurir@unmsm.edu.pe.



Johnny Alberto Angelino Gutierrez    is a graduate of the Faculty of Electronic and Electrical Engineering at the National University of San Marcos. During the project, he specialized in the design of computer vision models, focusing on training the algorithm for risk event detection. He has developed skills in image processing and Python programming. His contribution relates to the construction of the dataset and the preprocessing of model data. His technological interests include applied artificial intelligence and computer vision. He can be contacted via email: [johnny.angelino@unmsm.edu.pe](mailto:johnny.angelino@unmsm.edu.pe) y [angelinosdf@gmail.com](mailto:angelinosdf@gmail.com)



Andrés Reynaldo Quispe Calla    is a graduate of the Faculty of Electronic and Electrical Engineering at the National University of San Marcos. He focused on the implementation of embedded systems and detection models on a Raspberry Pi and communication with the ESP32 microcontroller. He has developed embedded systems and communication networks. His work focused on hardware configuration and interface development. His interests include the Internet of Things (IoT) and systems. He can be contacted via email: [andres.quispe4@unmsm.edu.pe](mailto:andres.quispe4@unmsm.edu.pe) y [andres26rqc@gmail.com](mailto:andres26rqc@gmail.com)