Development of YOLO-based Model for Fall Detection in IoT Smart Home Applications

Pengcheng Gao*
School of Cyber Security, Gansu University of Political Science and Law, Lanzhou 730070, Gansu, China

***Abstract*—In smart home applications, effective fall detection is a critical concern to minimize the occurrence of falls leading to injuries, especially for the assistance of elderly individuals. Various methods have been proposed, including both vision-based and non-vision-based approaches. Among these, vision-based approaches have garnered significant attention from researchers due to their practicality and applicability. However, existing vision-based methods face challenges such as low accuracy rates and high computational costs, which still need further exploration to enhance fall detection effectiveness. This study aims to develop a vision-based fall detection system tailored for smart home care applications. The objective of this study is to develop an accurate and lightweight fall detection method that is applicable in IoT platforms. A You Only Look Once (YOLO) based network is trained and tested to identify human falls accurately. The experimental results demonstrate that the developed YOLO-based technique shows promising outcomes for human fall detection and holds potential for integration in the Internet of Things (IoT) enabled smart home applications.**

Keywords—Smart home; IoT; elderly care; computer vision; deep learning; YOLO

I. Introduction

In recent years, the rapid progress in Information and Communication Technology (ICT) has brought about significant changes in people's lives thanks to groundbreaking innovations. This has led to the rise of intelligent environments, cities, and societies [1]. By leveraging cutting-edge technologies like Artificial Intelligence (AI) and the Internet of Things (IoT), we have the potential to greatly improve our quality of life. These advanced solutions empower us to monitor our surroundings and make well-informed decisions to achieve desired outcomes. Among these advancements, smart homes play a crucial role as the cornerstone of smart living. They are poised to be instrumental in the development of smart cities and societies since homes serve as the fundamental building blocks for both urban areas and social structures.

Presently, several socioeconomic factors are contributing to a significant decline in fertility rates and an increase in life expectancy [2]. As a result, a growing number of elderly individuals are striving to maintain their independence and stay in their own homes. To address this, an automated home-based solution that reduces the burden on healthcare services and provides valuable insights into fall risk becomes an appealing alternative [3]. Furthermore, with the global increase in the elderly population, healthcare considerations for seniors are becoming increasingly critical. For this reason, human motion capture technologies are essential for elderly individuals living alone, as they can help tackle these challenges. By observing their posture, it becomes possible to monitor the health of elders, and if high-risk postures such as falling are detected, timely warnings can be sent [4], [5].

Compared to traditional Machine Learning (ML) algorithms, deep learning significantly simplifies the process of feature selection by automatically extracting abstract features through multiple hidden layers [6]. The effectiveness of deep learning in unsupervised learning and reinforcement learning has been proven, leading to a surge in the development of deep learning-based Human Activity Recognition (HAR) frameworks [5]. In particular, Convolutional Neural Networks (CNN), inspired by the hierarchical processing in the human visual cortex, have achieved remarkable success in image categorization in recent times [7]. CNN-based methods can automatically learn distinctive features from training data, making them highly efficient for feature extraction and classification tasks [8], [9].

Two general categories can be made for CNN-based identification tasks. The first category consists of two-stage detection algorithms that divide the phases of target detection into finding and identifying them. Conventional approaches, such as Region-Convolutional Neural Networks (R-CNN), have flaws and fall short of real-time performance requirements. Faster R-CNN and Faster R-CNN have been introduced, although they are still insufficient for real-time applications [10]. The second group uses a single-stage detection method that combines the positioning of the target with its identification.

This study addresses the research problem of devising a precise and resource-efficient fall detection method suitable for IoT platforms. The research questions include the design of such a method, the effectiveness of utilizing the YOLO5 network due to its memory-efficient and speedy detection characteristics, and the steps required for dataset curation to train and test the YOLO model in recognizing fallen postures for improved fall detection. The research objectives encompass creating an optimized fall detection solution for IoT environments, assessing the YOLO5 network's suitability for this purpose, and developing and preparing a dataset for robust fall posture recognition within the YOLO framework.

The main contributions of this study are as follows:

1) Developing a vision-based approach for fall detection with feasibility and applicability considerations.

2) A Yolo-based network implementation and model generation for human fall posture identification.

3) Fall dataset generation using image collection from various internet resources and annotation and augmentation process.

The rest of this paper is consisted as: Section II reviews the related works. Section III discuss about the research methodology. Section IV presents the results and discussion. Finally, this paper concludes in Section V.

II. Related Works

Ajerla et al.'s [11] developed a fall detection framework based on an LSTM network that took advantage of edge computing tools like laptops, reducing the requirement to upload raw data to the cloud for real-time fall event prediction. The system used the open-source Apache Flink streaming engine to process the three-axis accelerometer raw data. A part of the MobiAct dataset, which is openly accessible, was used for training and testing. To get the best results, the system advised putting sensors at the waist—the suggested framework successfully anticipated fall occurrences using real-time fall data with an amazing accuracy of 95.8%. The use of various sensors and data streams led to improved performance.

Queralta et al. [12] proposed a fall detection system for health monitoring facilities using low-power wide-area network (LPWAN) technologies with Edge computing and Fog computing, as well as a compression technique for data transfer, lowering system latency. To recognize falls from the received data, LSTM and RNN networks were developed on the edge computer. These edge gateways were used to transmit real-time alerts and notifications while unprocessed data was transferred to the cloud for online processing. With this strategy, the operation was possible even in places with poor network access and increased battery life.

By suggesting an approach based on video analysis, Wang et al. [13] attempted to increase fall detection accuracy and speed in complex contexts. The introduction of the YOLOv3 network model as the detection algorithm was the main contribution. In order to train and test the network model on a GPU server, they constructed their dataset for human fall detection using the Pascal VOC data set format. According to experimental data, the method is more reliable and efficient than traditional fall detection algorithms, achieving a mAP of 0.83 and an AP down at 0.97.

In study [14], an approach to detecting human falls based on the Fast Pose Estimation technique was presented. The method classified data from picture frames using TD-CNN-LSTM and 1D-CNN models, demonstrating excellent accuracy. The suggested technique proved to be a valuable addition to reliable human fall detection, suitable for implementation in edge devices due to its minimal computational and memory requirements. They enhanced the URFD dataset for training by applying rotation, brightness adjustments, horizontal flipping, and gamma correction.

III. Research Methodology

This section presents the research methodology. Firstly, the background of the YOLO method is discussed. Secondly, the preparation of the dataset is presented. Thirdly, the implementation environment using Google Colab is explained, and lastly, the training and testing procedure is presented for fall detection.

A. Background of YOLO

YOLO (You Only Look Once) was a pretrained object detector designed to recognize common objects like tables, chairs, cars, phones, and more [15]. A newer version of the YOLO algorithm, called YOLOv5, has been proposed with enhancements over YOLOv3. YOLOv5 achieves greater precision and smaller model size, leading to significantly faster detection speed compared to its predecessor. Despite its potential, the YOLOv5 technique has not yet been widely applied in fall detection [16]. Therefore, this paper aims to improve the YOLOv5 model for detecting senior fall behavior.

The YOLOv5 is one of the most prominent models in the one-stage detection series, avoiding the recomputation of candidate areas utilized in the two-stage series. It boasts excellent recognition accuracy and quick inference. The YOLOv5 architecture comprises four primary model structures: YOLOv5l, YOLOv5x, YOLOv5m, and YOLOv5s, each offering progressively fewer complex networks. Additionally, a YOLOv5n model was later developed with only 1.9 MB parameters, the same depth as YOLOv5s but with half the network width, making it suitable for deployment on mobile devices.

As depicted in Fig. 1 [16], the YOLOv5 baseline architecture consists of three main components: the backbone, neck, and head. Fig. 1(a) to Fig. 1(d) illustrates the composition of modules related to the baseline architecture. One of the Backbone structures is a Convolutional Neural Network (CNN), which combines various fine-grained images to form image features.

The architecture utilizes the conv module for 2D convolution, regularization, and activation. The c3 module aids in feature extraction, reducing the model size and enhancing inference speed. The up-sample and concat modules handle feature map sampling and combination. The spatial pyramid pooling (SPP) module expands the network's perceptual area. The Neck structure improves information flow with feature pyramid network (FPN) and path aggregation networks (PAN). Adaptive pooling connects features for optimal data utilization. Overall, these components optimize the model's accuracy and efficiency.

B. Dataset Preparation

The dataset for fall detection was compiled from diverse sources, including Google Images, to create a custom dataset. This dataset consists of images categorized into three labels: "Fall Detected," "Walking," and "Sitting “. The initial dataset involves 485 images. To prepare the dataset with mode diversity, image augmentation is performed. After augmentation, total dataset involves 1455 images. The labels directory also has two subdirectories, namely "train", "val." and “test”. Within these directories, text files are provided, containing labels corresponding to specific images. Fig. 2 displays some examples from our dataset.

The dataset for fall detection may not be large by industry standards, but it's essential to consider its specific context and the chosen model's complexity. As our experimental results indicated, it is sufficient for a well-designed model. Additionally, the dataset is diverse and representative of real-world scenarios, that it helps the model generalize effectively.

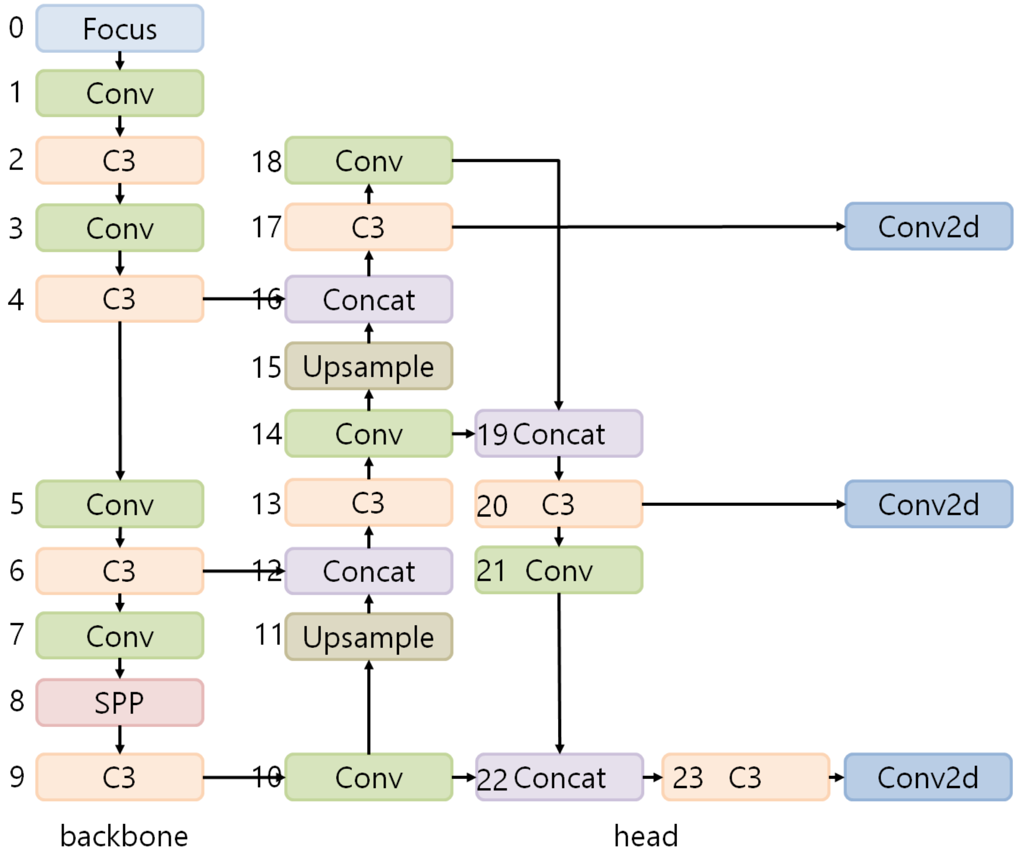

Fig. 1. The architecture of the YOLOv5 network.

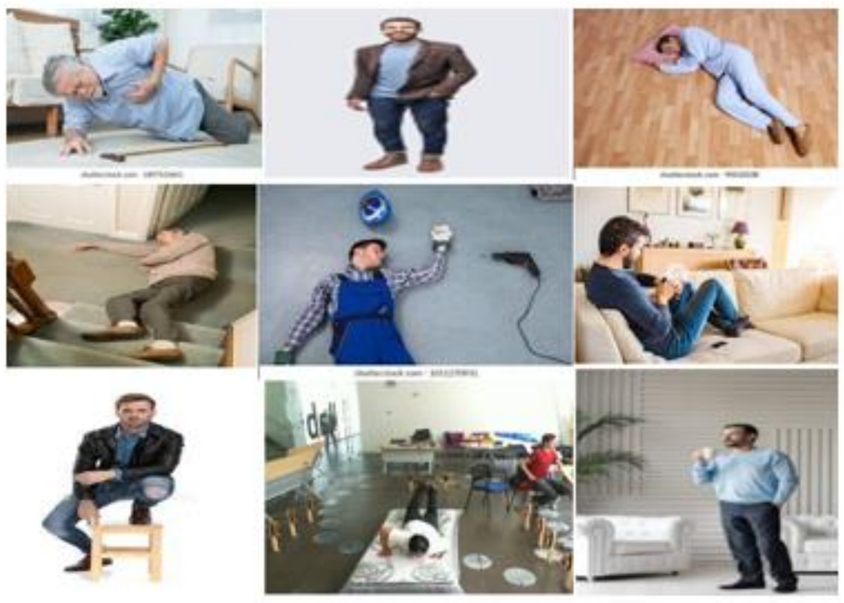

Fig. 2. Sample images from the dataset.

C. Training and Testing

When training an object detector, one common approach is to start with a preexisting model that has already been trained on large and diverse datasets. These pretrained models have learned to recognize various objects from the data they were initially trained on. Although the pretrained weights may not include specific objects relevant to the current experiment, they still capture valuable general features and patterns that can be beneficial for the new task.

This process of using a pretrained model as a starting point and fine-tuning its weights for a specific task is called transfer learning. By leveraging transfer learning, we can save time and computational resources, as the pretrained model has already learned to detect common objects effectively. The model acts as a feature extractor that can be fine-tuned to recognize the specific objects we need in our experiment.

In this case, a pretrained model containing weights trained on the COCO dataset is used as the starting point for the object detection task. COCO is a large and diverse dataset that includes a wide range of objects from various categories. Using a model pretrained on COCO, our network can benefit from the learned features, leading to faster convergence during training.

With transfer learning, we can achieve good results with fewer training data. In this experiment, the total dataset consists of 1455 images. To split the dataset for training, validation and testing, 70% of the images are used for training the model, and the remaining 20% are used for validating, and 10 % for testing for performance evaluation of the model.

In summary, transfer learning is a powerful technique that allows us to leverage existing knowledge from pretrained models to boost the performance of our object detection task. Utilizing a pretrained model and carefully selecting the appropriate amount of data for training and validation, we can efficiently train an accurate and effective object detector for our specific needs.

IV. Results and Analysis

In this section, we discuss the experiment's details, then show the training results using pretraining weights and compare the three models of YOLOv5.

A. Experimental Results

At this stage, we show a series of the model's stress measurement results and achieved high accuracy. Fig. 3 illustrates the results of the prediction fall in the dataset.

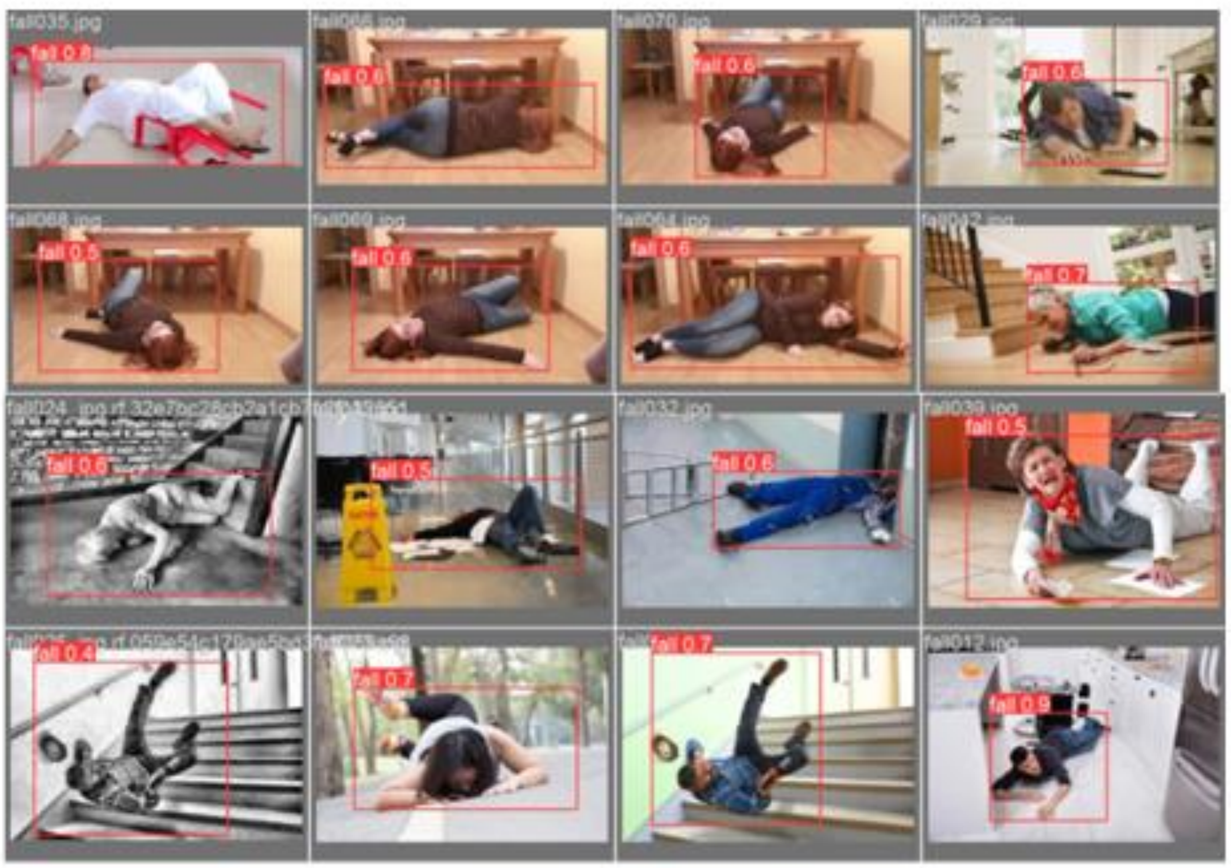

Fig. 3. Prediction results of fall detection.

B. Model Evaluation

The first version of our model was trained for the training set. The results of it are shown in Fig. 4. Label 0 is for falling, label 1 for walking and label 2 for sitting. As experimental results are shown in Fig. 4, this model achieved relatively accurate results.

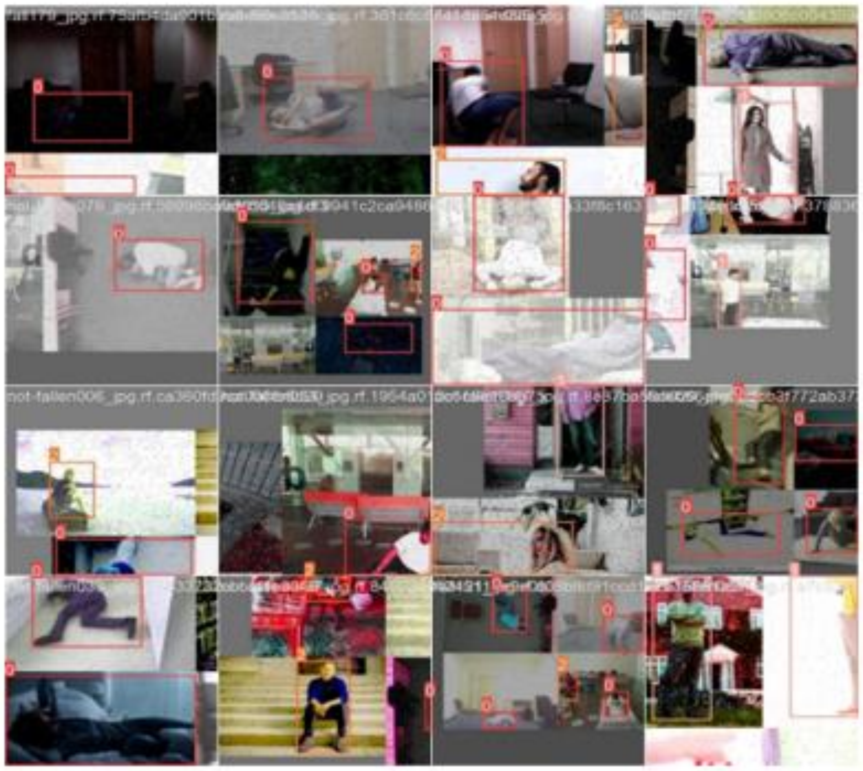

Fig. 4. Labels in the training process.

Various performance metrics are computed to evaluate the fall detection model. These metrics typically include precision, recall, and F1 score. We calculated the precision rate and recall rate. The results of the evaluation metrics of the trained model are shown in Fig. 5. Precision measures the accuracy of fall predictions, recall measures the model's ability to detect all falls, and the F1 score combines precision and recall into a single value. An analysis of the model's errors is performed to gain insights into its performance. This involves examining false positive and false negative predictions [18]. False positives are instances where the model incorrectly identifies a non-fall instance as a fall, while false negatives are cases where the model fails to detect an actual fall. Analyzing these errors helps identify areas for improvement in the model and dataset. Fig. 5 to Fig. 7 illustrates the performance metrics.

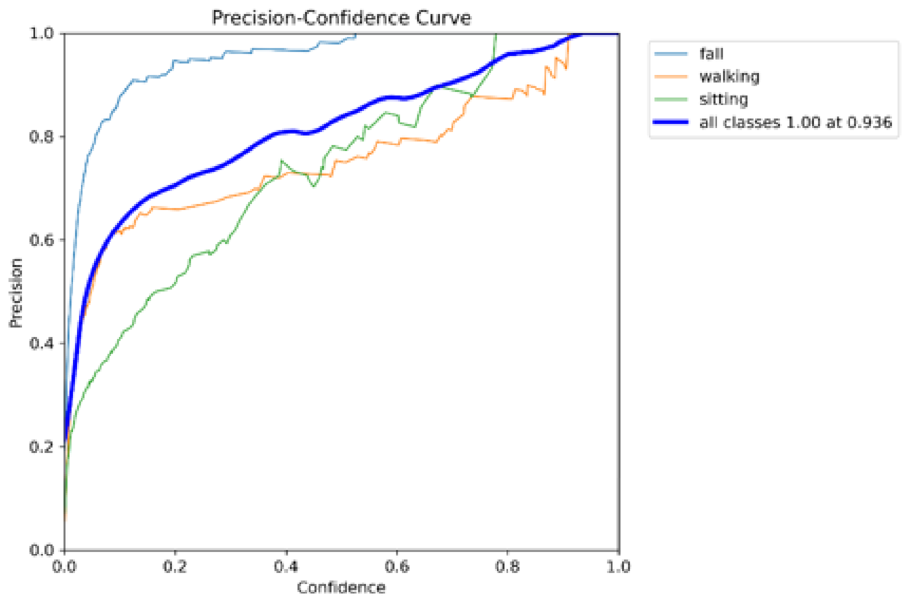

Fig. 5. Result of precision metric.

As shown in Fig. 5, a precision curve graph for a generated YOLOv5 model fall detection system with fall, walking, and sitting classes shows the relationship between the confidence rate and the precision rate. The X-axis represents the confidence rate, which indicates the level of confidence the model has in its predictions. In contrast, the Y-axis represents the precision rate, which measures the accuracy of the fall detection system. To obtain accurate results based on the obtained precision confidence from the generated YOLOv5 model, the following steps are typically followed:

Prediction and Confidence Threshold: The YOLOv5 model is applied to the test dataset, and for each detected object, the model assigns a confidence score or probability indicating its confidence in the prediction. The confidence score represents the model's belief that the object belongs to a particular class, such as falling, walking, or sitting. To generate the precision curve graph, different confidence thresholds are set to analyze the trade-off between precision and recall.

Precision Calculation: For each confidence threshold, the model's predictions are compared against the ground truth labels. True positive (TP) refers to the correct detection of a fall instance, false positive (FP) represents a non-fall instance being incorrectly identified as a fall, and false negative (FN) indicates a missed detection of a fall. The precision is then calculated using the formula: Precision = TP / (TP + FP) [17,18]. By examining the precision curve graph, one can identify the confidence threshold that provides the desired precision rate for fall detection. It allows for fine-tuning the system based on the specific requirements, striking a balance between accurate fall detection and minimizing false positives.

As depicted in Figure 5, the achieved precision rate of 0.93 for the YOLOv5 model in detecting fall, walking, and sitting classes is highly indicative of its effectiveness. A precision rate of 0.93 implies that 93% of the predicted positive cases were indeed true positives, minimizing false positives. This high precision indicates the model's ability to accurately classify these activities, reducing the likelihood of misclassification.

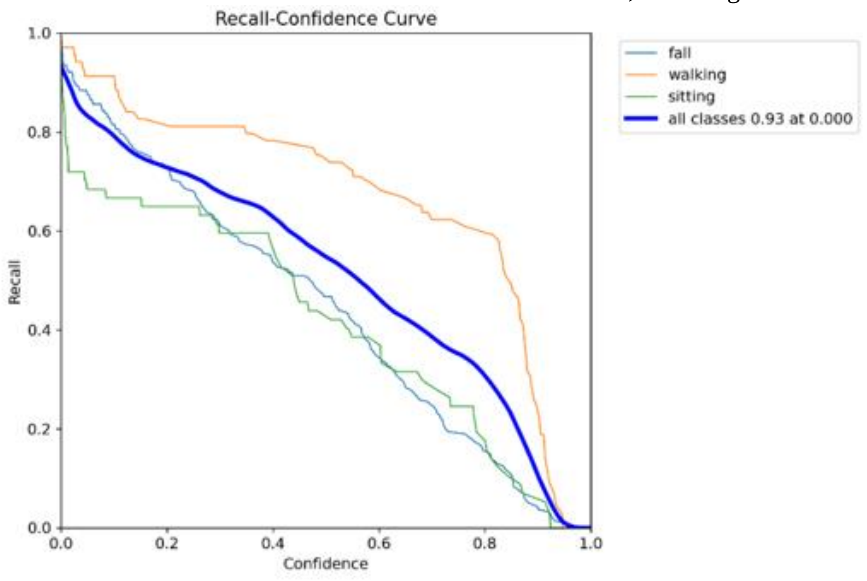

Fig. 6. Result of recall metric.

Fig. 6 illustrates the recall curve. A recall curve graph for a generated YOLOv5 model fall detection system with fall, walking, and sitting classes shows the relationship between the confidence rate and the recall rate. The X-axis represents the confidence rate, which indicates the level of confidence the model has in its predictions, while the Y-axis represents the recall rate, which measures the ability of the fall detection system to correctly identify all instances of falls.

For recall calculation, the model's predictions are compared against the ground truth labels for each confidence threshold. True positive (TP) refers to the correct detection of a fall instance, false positive (FP) represents a non-fall instance being incorrectly identified as a fall, and false negative (FN) indicates a missed detection of a fall. The recall is then calculated using the formula: Recall = TP / (TP + FN) [19,20].

Moreover, for recall curve plotting, as the confidence threshold is varied, the recall rate is calculated at each point. These recall values are plotted against the corresponding confidence thresholds on the graph. The resulting recall curve shows how the recall rate changes as the confidence rate increases. A higher recall rate indicates that the fall detection system is more effective in correctly identifying all fall instances. The recall curve graph allows us to analyze the relationship between recall and confidence thresholds. Based on the desired trade-off between recall and precision, an optimal confidence threshold can be selected. If maximizing the number of detected falls is the priority, a lower confidence threshold can be chosen, which may result in higher recall but potentially more false positive predictions. By examining the recall curve graph, one can identify the confidence threshold that provides the desired recall rate for fall detection.

As depicted in Fig. 6, the overall recall rate is 0.93 for the YOLOv5 model in detecting fall, walking, and sitting classes. I show that the model successfully captured 93% of all actual positive cases, demonstrating its ability to detect these classes with a high level of sensitivity.

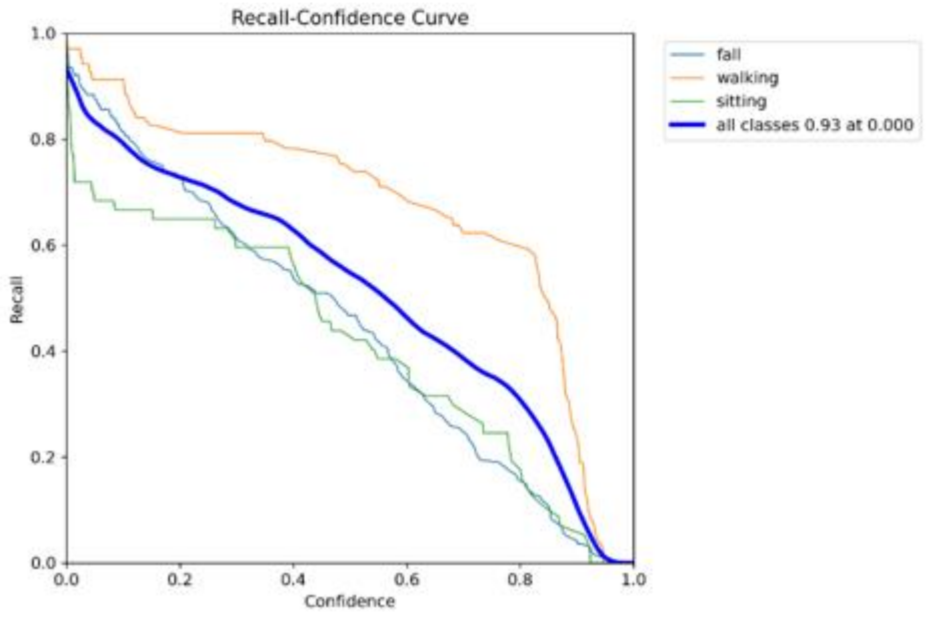

Fig. 7. Result of precision-recall curve.

As shown in Fig. 7, the precision-recall curve graph for a generated YOLOv5 model fall detection system with fall, walking, and sitting classes shows the relationship between the confidence rate and the precision-recall rate. The graph provides a visual representation of how precision and recall change as the confidence threshold varies. Precision measures the accuracy of fall predictions, while recall measures the ability to detect all fall instances. By examining the curve, the optimal confidence threshold can be determined based on the desired balance between precision and recall.

The precision-recall curve graph helps in evaluating the performance of the fall detection system and selecting the appropriate confidence threshold. A higher precision indicates more accurate fall predictions, while a higher recall indicates a greater ability to detect all fall instances. The graph allows for the analysis of the trade-off between precision and recall, enabling the system to be fine-tuned to meet specific requirements. By selecting the optimal confidence threshold based on the precision-recall curve, the generated YOLOv5 model can achieve accurate results in detecting falls while minimizing false positives and false negatives.

Finally, as illustrated in Fig. 7, the overall precision-recall rate of 0.93 for the YOLOv5 model in detecting fall, walking, and sitting classes. This metric signifies a balanced performance in terms of precision (the ability to correctly classify positive cases) and recall (the ability to capture all actual positive cases). This obtained score demonstrates the model strikes a favorable balance between minimizing false positives and successfully identifying true positives.

V. Conclusion

In IoT smart home applications, detecting human fall detection is a difficult problem. The high complexity, poor accuracy, and time constraints of human fall detection in smart home applications is the focus of this work. The aim of this study is to develop an accurate and lightweight fall detection method that is applicable in IoT platforms. It developed a vision-based fall detection system that can recognize human fallen posture for use in smart home applications. The developed method involves training and testing a YOLO network to identify the postures in the prepared dataset. Based on the YOLO5 algorithm, which offers a high accuracy rate and satisfactory speed in posture identification, this Yolo-based technique was developed. One limitation of this study is the reliance on a relatively small dataset, which may limit the diversity and representation of fall-related scenarios. A larger and more diverse dataset could provide a more comprehensive understanding of fall detection in various real-world situations, potentially enhancing the model's generalizability and robustness to different environmental and contextual factors. Future work in this area could involve the expansion of the dataset to include a wider variety of fall-related scenarios, encompassing different environments, age groups, and diverse physical conditions. This would help improve the model's ability to handle a more extensive range of fall detection challenges. Other potential directions for future study include improving the accuracy and performance of the system by exploring alternative deep learning models or refining the existing technique. Another direction is to focus on the real-time implementation and deployment of the system in real-world smart home environments, considering factors such as scalability, reliability, and integration with IoT technologies. These advancements would contribute to the effective utilization of the system in IoT-based smart home applications.

References

[1] E. Alqahtani, N. Janbi, S. Sharaf, and R. Mehmood, “Smart homes and families to enable sustainable societies: A data-driven approach for multi-perspective parameter discovery using bert modelling,” Sustainability, vol. 14, no. 20, p. 13534, 2022.

[2] S. Zolfaghari, E. Khodabandehloo, and D. Riboni, “TraMiner: Vision-based analysis of locomotion traces for cognitive assessment in smart-homes,” Cognit Comput, vol. 14, no. 5, pp. 1549–1570, 2022.

[3] G. Forbes, S. Massie, and S. Craw, "Fall prediction using behavioural modelling from sensor data in smart homes," Artif Intell Rev, vol. 53, no. 2, pp. 1071–1091, 2020.

[4] W. Quan, J. Woo, Y. Toda, and N. Kubota, "Human posture recognition for estimation of human body condition," Journal of Advanced Computational Intelligence and Intelligent Informatics, vol. 23, no. 3, pp. 519–527, 2019.

[5] L. M. Dang, K. Min, H. Wang, M. J. Piran, C. H. Lee, and H. Moon, "Sensor-based and vision-based human activity recognition: A comprehensive survey," Pattern Recognit, vol. 108, p. 107561, 2020.

[6] T. Plötz and Y. Guan, "Deep learning for human activity recognition in mobile computing," Computer (Long Beach Calif), vol. 51, no. 5, pp. 50–59, 2018.

[7] S. Ramasamy Ramamurthy and N. Roy, "Recent trends in machine learning for human activity recognition—A survey," Wiley Interdiscip Rev Data Min Knowl Discov, vol. 8, no. 4, p. e1254, 2018.

[8] A. Kamel, B. Sheng, P. Yang, P. Li, R. Shen, and D. D. Feng, "Deep convolutional neural networks for human action recognition using depth maps and postures," IEEE Trans Syst Man Cybern Syst, vol. 49, no. 9, pp. 1806–1819, 2018.

[9] A. Khan, A. Sohail, U. Zahoora, and A. S. Qureshi, "A survey of the recent architectures of deep convolutional neural networks," Artif Intell Rev, vol. 53, pp. 5455–5516, 2020.

[10] T. Chen, Z. Ding, and B. Li, "Elderly Fall Detection Based on Improved YOLOv5s Network," IEEE Access, vol. 10, pp. 91273–91282, 2022.

[11] D. Ajerla, S. Mahfuz, and F. Zulkernine, "A real-time patient monitoring framework for fall detection," Wirel Commun Mob Comput, vol. 2019, pp. 1–13, 2019.

[12] J. P. Queralta, T. N. Gia, H. Tenhunen, and T. Westerlund, "Edge-AI in LoRa-based health monitoring: Fall detection system with fog computing and LSTM recurrent neural networks," in 2019 42nd international conference on telecommunications and signal processing (TSP), IEEE, 2019, pp. 601–604.

[13] X. Wang and K. Jia, "Human fall detection algorithm based on YOLOv3," in 2020 IEEE 5th International Conference on Image, Vision and Computing (ICIVC), IEEE, 2020, pp. 50–54.

[14] M. Salimi, J. J. M. Machado, and J. M. R. S. Tavares, "Using deep neural networks for human fall detection based on pose estimation," Sensors, vol. 22, no. 12, p. 4544, 2022.

[15] M. Hatab, H. Malekmohamadi, and A. Amira, "Surface defect detection using YOLO network," in Intelligent Systems and Applications: Proceedings of the 2020 Intelligent Systems Conference (IntelliSys) Volume 1, Springer, 2021, pp. 505–515.

[16] G. Dai, L. Hu, and J. Fan, "DA-ActNN-YOLOV5: hybrid YOLO v5 model with data augmentation and activation of compression mechanism for potato disease identification," Comput Intell Neurosci, vol. 2022, 2022.

[17] Aghamohammadi, A., Ang, M.C., A. Sundararajan, E., Weng, N.K., Mogharrebi, M. and Banihashem, S.Y. A parallel spatiotemporal saliency and discriminative online learning method for visual target tracking in aerial videos. Plos one, 13(2), p.e0192246, 2018.

[18] Lin, Bor-Shing, Tiku Yu, Chih-Wei Peng, Chueh-Ho Lin, Hung-Kai Hsu, I-Jung Lee, and Zhao Zhang. "Fall detection system with artificial intelligence-based edge computing." IEEE Access 10 (2022): 4328-4339.

[19] Gomes, Mouglas Eugenio Nasario, David Macedo, Cleber Zanchettin, Paulo Salgado Gomes de-Mattos-Neto, and Adriano Oliveira. "Multi-human fall detection and localization in videos." Computer Vision and Image Understanding 220, 2022.

[20] Yacchirema, Diana, Jara Suárez de Puga, Carlos Palau, and Manuel Esteve. "Fall detection system for elderly people using IoT and ensemble machine learning algorithm." Personal and Ubiquitous Computing ,2019, pp. 801-817.