A Deep Learning Approach for Basic Human Activity Recognition with YOLOv4-tiny

Jason Halim
Electrical Engineering Department
Petra Christian University
Surabaya, Indonesia
jhelectrical101@gmail.com

Handy Wicaksono
Electrical Engineering Department
Petra Christian University
Surabaya, Indonesia
handy@petra.ac.id

***Abstract*—The regression of the elderly body condition over time causes the elderly to be more susceptible to illness and other accidents. When the elderly experience illness or an accident such as a fall, it is necessary for the family to realize this and take prompt treatment immediately. However, with relatively many elderly people in Indonesia choosing to live independently, it will be difficult for families to find out and provide help instantly. Therefore, in this study, the authors form a model for recognizing basic human activities with deep learning-based computer vision that can be implemented in a supervisory system in a room. A deep-learning approach is needed because of the complexity and variance of body postures and forms of human activities. However, the deep learning approach requires extensive resources and computational capabilities. Therefore, the model is formed by the YOLOv4-tiny method, one of the tiny versions of YOLO. Model training using the author's laptop and model inference testing was carried out on the Jetson Nano and the author's laptop to compare the inference time between the two devices. We investigate the performance of the YOLOv4-tiny model application on the Jetson Nano and a laptop, as well as the accuracy of recognizing human activities. This study shows that this particular vision-based human activity recognition model formed using YOLOv4-tiny as a deep learning method can be applied using Jetson Nano as an embedded device in real-time, with a speed of about 20 frames per second, mAP@0.50 of 99.04%, and an average F1-Score of 94.18%.**

***Keywords*—Human Activity Recognition, Deep Learning, Computer Vision, YOLOv4-tiny, Jetson Nano**

I. Introduction

When reaching old age, some choose to stay with their extended family, and the others decided to live. Not only are the elderly who live alone, but elderly who are left alone at home are a group at risk and require special attention. As people age, the muscles will weaken and affect the strength and balance of the body, causing the elderly to fall easily. Approximately 28-35% of people aged of 65 and over fall each year [1, 2, 3] increasing to 32-42% for those over 70 years of age [4, 5, 6]. In addition, the potential for loss of consciousness due to heart problems, problems with feet, vision, and hearing can make the elderly easily slip, hit furniture, and fall. A sudden fall can be fatal for the elderly, especially if they don't get immediate help [7]. Therefore, it would be better if there is a supervisory system that could detect the basic activities of the elderly, especially for activities that are not commonly done, such as sitting on the floor and lying on the floor.

Several methods that can be used to detect the posture of the elderly are by using wearable devices [8, 9] and nonwearable devices. Nonwearable devices can be divided into non vision-based devices [10] and vision-based. The development of computer vision and artificial intelligence technology has prompted various research to be carried out in the vision-based recognition of human activities, known as HAR or Human Activities Recognition [11]. The method that can be used to recognize humans is the hand-crafted features and feature learning approach. Feature learning methods can be categorized into non-deep-learning and deep learning. The feature learning approach with deep learning methods is a topic that is widely explored because of its excellent performance in extracting features from data and its ability to generalize various kinds of data.

One algorithm that is often used for object recognition and classification is YOLO [12]. YOLO generally has a fast performance when compared to other algorithms. The YOLO algorithm has also been used several times in research for the YOLO human form and activity recognition system [13, 14, 15]. Researchers found that from previous studies, no studies specifically identified and classified humans on mattresses and on the floor with human objects that might be blocked from view of the camera or occluded. This research was conducted to fill the gap from previous studies using the YOLOv4-tiny method to be tested on a laptop and the Jetson Nano as an embedded device to recognize basic human activities.

II. Literature Review

In recent years, many researchers have adopted deep learning approaches such as the CNN approach to overcome problems in object classification in images and videos [16]. The algorithms that are the mainstay in detecting and classifying objects are faster R-CNN [17] and YOLO [12]. Faster R-CNN has a higher mean average precision value than YOLO, but it is different because YOLO detects and classifies objects simultaneously so that the process runs much faster and can be implemented in real-time. YOLO has begun to be widely used in research on making posture and activity detection systems.

Jun Wu et al. [13] studied human pose recognition by utilizing Kinect V2 to capture temporal and spatial data of human objects in images. Then the data will be converted to RGB form and fed to the YOLO algorithm. There are 2000 RGB images used for training with a size of 25x200. Activities that try to be recognized are standing, sitting, lying down, getting up (from lying down), eating, reading, and drinking. The results of the training system were designed to run well in real-time with the mean average precision of

81.88%. However, the system's performance is not good at detecting humans whose body parts are occluded by other objects.

Shinde et al. [14] conducted a study on the introduction of human interaction both with fellow humans and other objects using YOLO. Ten activities are trying to be identified from 367 actions contained in 167 videos. The 167 videos were divided into 109 videos for training and 58 videos for testing. YOLO training is carried out for 40000 iterations. The model results have an accuracy rate of 88.378%. The system created can be said to run very well. However, for the system to run at real-time speed, it requires a GPU with medium to high quality. The GPU used by the researchers is a Gigabyte GeForce GTX 1050Ti, which in terms of performance can be considered as mid-tier processing unit.

Gul et al. [15] conducted a study to identify patients' activities that were considered abnormal, especially those related to a sick body condition using YOLO. There are 23040 frames that are used as a set of data. The training was carried out twice. In the first experiment, the dataset was divided into a ratio of 60:40, while in the second experiment, the dataset was divided into a ratio of 70:30. The test results are very good, with an accuracy rate of 95.7% and 96.8%. The system can be run in real-time conditions with excellent performance. The training and testing were carried out using the Titan XP GPU, which in terms of performance can be considered as high-tier processing unit.

Heriyanto [27] conducted a thesis in the form of research on a defect detection system of beverage packaging using YOLOv3-tiny on Jetson Nano. YOLOv3-tiny [23] is one of the lightweight versions of the YOLO method based on YOLOv3 [24], which can be applied using embedded devices such as the Jetson Nano, which in terms of their performance can be considered as low-tier processing unit. The dataset used is 200 images. The system made can only display at a speed of 2-3 FPS.

III. Proposed Methodology

This study uses an experimental method that aims to test the performance of the model that will be formed when run both on the researcher's laptop and on the Jetson Nano as an embedded device to introduce basic human activities.

The modeling process includes system design, software and hardware used, datasets, training and testing preparations, and metric equations.

A. System Design

The workflow that the author implemented is explained in Fig. 1. The author chose a one-stage regression approach with the YOLOv4-tiny method [20] for model training and testing. The training and model testing were carried out on the author's laptop using a personal dataset in images containing predetermined human activities.

Next, the model is loaded into OpenCV to run the object detection and classification process simultaneously using the help of CUDA and cuDNN. Boundary boxes and labels will appear on the object of human activity that corresponds to the class being trained. To reduce redundant boundary boxes and ensure that the detection obtained is good, the authors set a threshold score and a non-maximum suppression threshold, respectively, with a value of 0.5. The test is carried out to see the speed of the system in making inferences if the system is run on a laptop or on an embedded device with either a CPU or a GPU. The embedded device used in this study is the Nvidia Jetson Nano.

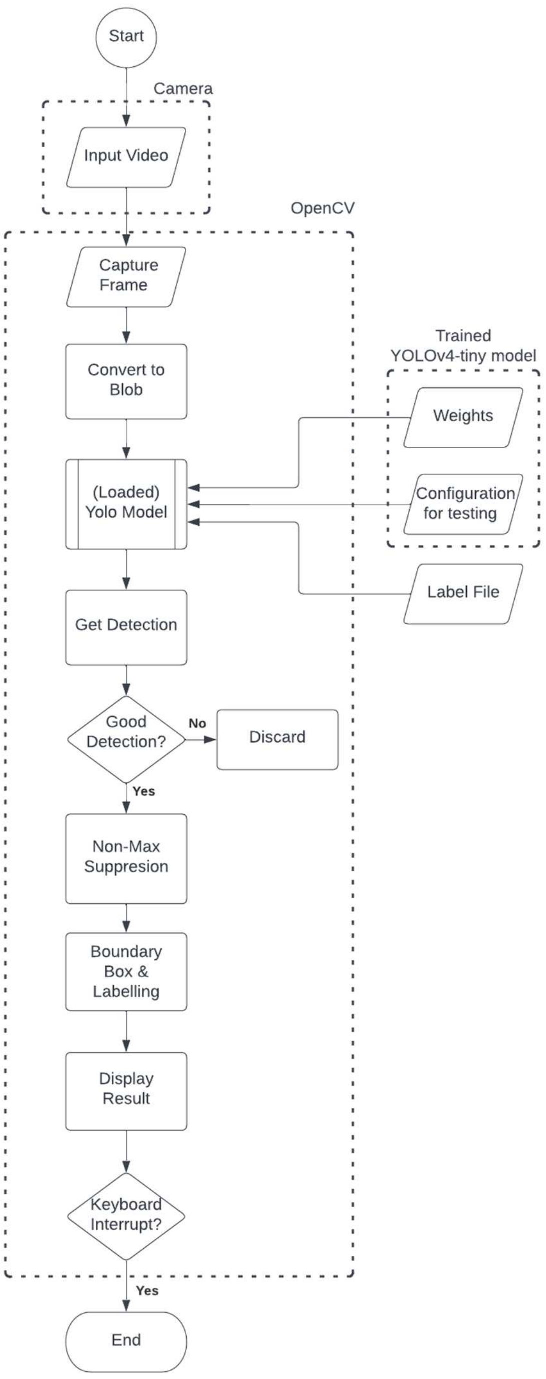

Fig. 1. The workflow of the proposed system

B. YOLOv4-tiny

YOLOv4-tiny [20] is one of the lighter versions of the YOLO (You Only Look Once) model series. YOLOv4-tiny was designed based on the YOLOv4 method [19]. The backbone that YOLOv4-tiny uses is CSPDarknet53-tiny, a lightweight version of CSPDarknet53 used in YOLOv4. To reduce detection duration, YOLOv4-tiny uses FPN (Feature Pyramid Network), and 2 YOLO heads compared to

YOLOv4, which uses SPP (Spatial Pyramid Pooling) and PANet (Path Aggregation Network) as well as 3 YOLO heads. YOLOv4-tiny was trained with 29 convoluted pre-trained layers, while YOLOv4 was trained with 137 convolutional pre-trained layers. This allows YOLOv4-tiny to detect faster than YOLOv4 while taking a slight dip in accuracy.

C. CUDA and cuDNN toolkit

The CUDA toolkit is provided by Nvidia for developing applications requiring GPU acceleration. In the CUDA toolkit there are libraries, compilers, CUDA runtime and several other development tools. cuDNN is a GPU-accelerated library for deep learning applications.

D. OpenCV

OpenCV (Open Source Computer Vision Library) [18] is a computer vision and machine learning software library. OpenCV provides a common infrastructure for computer vision applications. OpenCV is equipped with more than 2500 computer vision and machine learning algorithms. OpenCV also provides a module for deep learning inference since version 3.3 is called OpenCV DNN. OpenCV DNN is optimized specifically for Intel processors but can also be used on other processors.

E. Hardware

We used a Jetson Nano and a laptop for our experiments. Jetson Nano is an embedded device developed by Nvidia. The specification of Jetson Nano used is *128-Core NVidia Maxwell GPU, Quad-core ARM A57 @ 1.43GHz CPU, 2.00 GB RAM* which in terms of performance can be considered as low-tier GPU. The laptop used in this study is a laptop with an *Intel(R) Core i7-7500U CPU @ 2.70GHz, GPU NVidia 940MX 2GB RAM 20.00 GB* which in terms of performance can be considered mid-tier processing unit.

F. Dataset

The author created and used his own dataset for this study which consist of 1619 images with a size of 1280 x 720, of which the image contains a person who is doing or posing one of 5 simple activities as listed in Table 1. namely: lying on the mattress, lying on the floor, standing, sitting on the mattress, and sat on the floor. Classes are not evenly distributed. The dataset consists of images with a bedroom background containing a mattress and one person. The picture is taken in a bird's-eye view from 4 different positions which is shown in Fig. 2. In addition, the person uses three different sets of clothes to add variety to the dataset. The author also incorporated occluded human pose to improve the model's recognition capability which previously the model could not do [13]. The lying on the mattress and sitting on the pose are mostly not occluded. On the contrary, the standing pose, sitting on the floor pose and lying on the floor pose used for the dataset are mostly occluded. The image is obtained by taking frames from the recorded video. Images are captured every 1 second, which equals to every 60 frames.

Furthermore, the dataset will be tidied up by deleting images that do not match the desired criteria. After the dataset has been tidied up, each image will be labeled and drawn using the image annotation tool [21] to get a .txt file containing 5 values: class, x-axis center value, y-ordinate center value, width, and height. The dataset was randomly divided into 1267 images for the training dataset and 352 images for testing, as shown in Table 2.

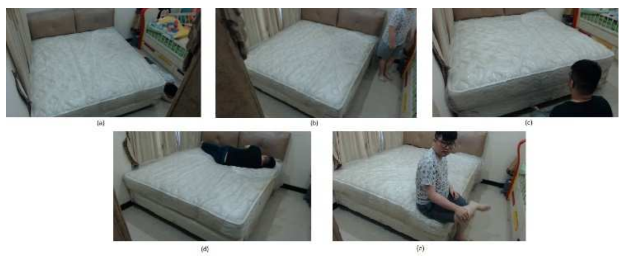

Fig. 2. Some samples of images in the dataset. *(a) point of view 1, pose: lying on the floor; (b) point of view 2, pose: standing; (c) point of view 3 pose: sitting on the floor; (d) point of view 4 pose: lying on the mattress; (e) point of view 4, pose: sitting on the mattress.*

TABLE I. CATEGORIES OF BASIC HUMAN ACTIVITIES IN THE DATASET.

No.	Activity	Abbreviation
1	Lying on The Mattress	LB
2	Lying on The Floor	LF
3	Standing	S
4	Sitting on The Mattress	SB
5	Sitting on The Floor	SF

TABLE II. DISTRIBUTION OF BASIC HUMAN ACTIVITY RECOGNITION DATASET.

No.	Abbreviated Label	Training	Testing	
1	LM	361	99	460
2	LF	199	54	253
3	S	231	65	296
4	SM	259	73	332
5	SF	217	61	278
		1267	352	1619

G. Training and Testing Preparation

The block diagram for training the YOLOv4-tiny model is explained in Fig. 3. which will be described in the next chapter. The YOLOv4-tiny method requires several files to be prepared and placed in the darknet folder [20], which will then be accessed to carry out the training process. The files that need to be prepared include:

- The .txt file which contains the path to all the images that are going to be used for training
- The .txt file contains the path to all the pictures that are going to be used for testing
- The file which contains the name of the entire class (obj.names)
- obj.data file
- Pre-trained convolutional weights (.weights) files
- YOLOv4-tiny configuration file (.cfg) for training
- YOLOv4-tiny configuration file (.cfg) for testing

The author set the values for several parameters in the training configuration file, such as the following:

- Batch = 64
- Subdivision = 32
- Max batch = 10000
- Steps = 8000, 9000 (80% and 90% of max batch)
- Filter (linear convolution layer) = 30

The author set the values for several parameters in the testing configuration file, such as the following:

- Batch = 1
- Subdivision = 1
- Max batch = 10000
- Steps = 8000, 9000 (80% and 90% of max batch)
- Filter (linear convolution layer) = 30

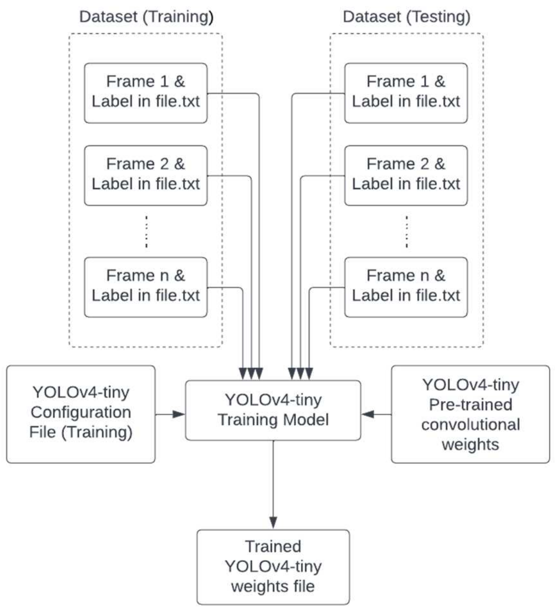
Dataset (Training)
Dataset (Testing)
Frame 1 & Label in file.txt
Frame 2 & Label in file.txt
Frame n & Label in file.txt
YOLOv4-tiny Configuration File (Training)
YOLOv4-tiny Training Model
YOLOv4-tiny Pre-trained convolutional weights
Trained YOLOv4-tiny weights file

Fig. 3. The block diagram of YOLOv4-tiny training process.

H. *Evaluation Metrics*

The metrics that are sought from the test results apart from the average accuracy are precision, recall, and F1-Score [16]. These metrics can show the fine-grain insight of how well the detection and recognition model performs, compared to just knowing the accuracy value. The parameters required are true positive (TP), false positive (FP), and false negative (FN).

Precision is a metric that shows how many positive predictions are true. To calculate the precision value [22]:

$$Precision = \frac{TP}{TP+FP} \tag{1}$$

Recall, or sensitivity, is a metric that shows how many true cases were correctly predicted. To calculate the recall value [22]:

$$Recall = \frac{TP}{TP+FN} \tag{2}$$

F1-Score, or harmonic mean, is a metric that combines the results of the precision and recall metrics harmonically. The F1-Score can provide a more precise indication of the model's performance, especially if the classes in the dataset have an uneven distribution. To calculate the F1-Score [22]:

$$F1 - Score = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{3}$$

There are two other parameters which can help give contexts to the training result and discussion, which is mAP and IoU. The mean Average Precision or mAP score is calculated by taking the mean AP over all classes and/or overall IoU thresholds, depending on different detection challenges that exist [25].

IoU (Intersection Over Union) is a metric that measures the overlap between 2 boundaries. IoU is used to measure how much our predicted boundary overlaps with the ground truth (the real object boundary). In this dataset, we predefine an IoU threshold at 0.50 in classifying whether the prediction is a true positive or a false positive [26].

$$IoU = \frac{Area\ of\ Overlap}{Area\ of\ Union} \tag{4}$$

IV. Result and Discussions

The training was run on the author's laptop and took approximately 80 hours. Each iteration takes 8.69 seconds in the training process with an average loss of 0.079. The training process speed depends on the GPU's capabilities. The training is carried out in up to 10000 iterations. The results of the training are in the form of several .weights files. The author uses the results of the best .weights files for testing. As for the test, it was carried out using a Jetson Nano and the author's laptop as a comparison. The selected .weights files, .cfg files for training, and class label files will be loaded by the program that has been built using the OpenCV DNN library when the program is run. Programs run on the GPU with CUDA and cuDNN. The mean average precision with intersection over union threshold at 0.50 or mAP@0.50 obtained from training YOLOv4-tiny model is 99.04%. This parameter shows that the predictions given by the model is mostly consistent and has high accuracy and sensitivity.

TABLE III. Comparison of the Mean Inference Time and Frames Per Second of the Recognition Model.

No.	Device/Hardware	Average Inference time (ms)	Frames Per Second (FPS)
1	NVIDIA GeForce 940MX (Laptop)	3.50	285.71
2	NVIDIA Jetson Nano	51.00	19.61

As shown in Table 3. the inference time consumed to perform activity recognition when the model is run on a laptop has an average speed of about 3.5 milliseconds or about 285 - 286 FPS (frames per second). Meanwhile, the average time required to perform activity recognition inference when the model is deployed on the Jetson Nano is about 51 milliseconds or about 19 - 20 FPS. Although the speed of the Jetson Nano is not as fast as the laptop, it is relatively close to real-time speed with a delay of less than 1 second.

TABLE IV. COMPARISON OF THE DEVICE USED FOR YOLO INFERENCE AND INFERENCE TIME NEEDED.

No.	YOLO Version	Device	Performance Tier	Frames Per Second (FPS)
1	Not Specified [14]	GeForce GTX 1050 Ti	Mid-tier	15 - 16
2	Not Specified [15]	Titan XP	High-tier	45
3	YOLOv3-tiny [27]	NVIDIA Jetson Nano	Low-tier	2 - 3
4	YOLOv4-tiny	NVIDIA Jetson Nano	Low-tier	19.6
5	YOLOv4-tiny	GeForce 940MX	Mid-tier	285 - 286

We also tried to compare the performance of YOLO method used on previous researches with disregard to their model purpose. As shown in Table 4. YOLOv4-tiny deployed on mid-tier processing unit has a speed of 285 - 286 FPS which far outperforms unspecified YOLO version model which is deployed on high-tier processing unit [15]. YOLOv4-tiny deployed on low-tier processing unit has a speed of 19.6 FPS, which also outperforms unspecified YOLO version model which is deployed on mid-tier processing unit and YOLOv3-tiny on low-tier processing unit [14, 27].

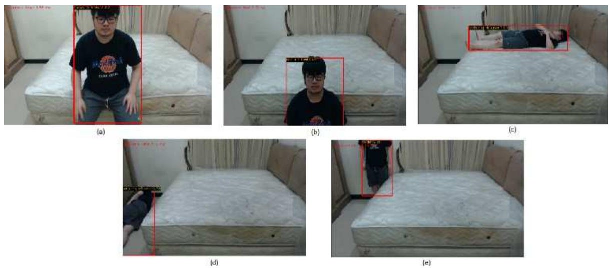

Fig. 4. New point of view in the same room. (a) pose: sitting on the mattress, (b) pose: sitting on the floor, (c) pose: lying on the mattress, (d) pose: lying on the floor, (e) pose: standing

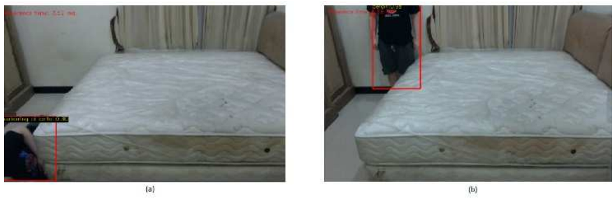

Fig. 5. Occluded human poses. *(a) pose: lying on the floor with the upper body colluded (b) pose: standing with the head and legs occluded.*

The model can recognize basic human activities and recognize them well when tested from the point of view of a different room while still in the same room. Fig. 4. shows that the model can detect and classify each class with a confidence value successively (a) sitting on the mattress with the confidence of 1.00, (b) sitting on the floor with the confidence of 0.99, (c) lying on the mattress with the confidence of 1.00, (d) lying on the floor with the confidence of 0.89, and (e) standing with the confidence of 0.98. The model can also recognize humans as objects that are occluded or part of the body that is occluded by other objects (see Fig. 5). Where the model can detect and classify each class with successive confidence values (a) sitting on the mattress with the confidence of 1.00, and (b) sitting on the floor with the confidence of 0.99.

The values derived from the test dataset are the primary metric for evaluating the model's performance. Table 5. shows the true positive, false positive, and false negative values for each class from the test results. Table 6 shows each class's calculation results of precision, recall, and F1-Score.

TABLE V. *PRECISION, RECALL, F1-SCORE* FOR EACH CATEGORY.

No.	Abbreviated Label	Precision (%)	Recall (%)	*F1-Score (%)*
1	LM	98.00	98.99	98.49
2	LF	94.64	98.15	96.36
3	S	77.78	98.44	86.90
4	SM	86.75	98.63	92.31
5	SF	95.31	98.39	96.83
	Mean	90.50	98.52	94.18

Table 5. shows that the model can recognize activities lying on the mattress, lying on the floor, standing, sitting on the mattress, and sitting on the floor with very high precision, recall, and F1-Score values with mean values of 90.50%, 98.52% and 94.18. %. This shows that in general the model can predict each activity extremely well. However, based on its precision value in, it can be seen that sometimes the model incorrectly predicts supposed-to-be other activities as standing or sitting on the mattress.

TABLE VI. *TRUE POSITIVE*, *FALSE POSITIVE*, AND *FALSE NEGATIVE* FOR EACH CATEGORY.

No.	Abbreviated Label	TP	FP	FN
1	LM	98	2	1
2	LF	53	3	1
3	S	63	18	2
4	SM	72	11	1
5	SF	61	3	0

As shown in Table 5. and 6. the standing class has a precision value of 77.78% with a positive predictive error (FP) of 18 from the validation dataset other than standing images. The class sitting on the mattress has a precision value of 86.75% with a positive predictive error (FP) of 11 from the image validation dataset sitting on the mattress.

The author conjecture that these false positive predictions arise due to author's attempt in making a model which can predict occluded objects. The author incorporated image with occluded objects for all but sitting on the mattress activity which can be seen in Fig. 4. and Fig. 5 to make a YOLO model which can comfortably recognize human activities even when the object is partially obscured. YOLO annotation captures the class value, center of axis, center of ordinate, width, and height. Some poses such as lying on the mattress, lying on the floor, and sitting on the floor can be easily distinguished by the model due to their unique combination of annotation value. On the other hand, some standing pose and sitting on the mattress pose can actually have identical value combination especially on parameter center of axis, center of ordinate, width, and height.

V. Conclusions

The conclusions obtained from our research are as follows:

1. YOLOv4-tiny can be deployed on low-tier processing unit such as NVIDIA Jetson Nano with a speed of 19 – 20 FPS which is close to real-time speed, making it a suitable deep learning model for smart surveillance system.
2. The high mean average precision (mAP) value (99.04%) and F1-Score (94.18%) in the model indicates that YOLOv4-tiny as a deep learning model can be used to comfortably recognize different static human activities.
3. Including images with occluded objects as part of model training results in better occluded activity detection, with a slight decrease of recognition precision, which results in the increase of false positive predictions, as its drawback.

In the future, we will focus on testing deep learning models on other embedded devices and various libraries to compare. In addition to the devices and libraries used, similar research, such as optimizing the YOLOv4-tiny method, can also be carried out to improve the performance of the tiny series YOLO model when the model is implemented using embedded devices.

VI. References

[1] Blake A et al. “Falls by elderly people at home: prevalence and associated factors.” *Age Ageing*, 1988, 17:365-372.

[2] Prudham D, Evans J. “Factors associated with falls in the elderly: a community study.” *Age Ageing*, 1981, 10:141-146.

[3] Campbell AJ et al. “Falls in old age: a study of frequency and related clinical factors.” *Age Ageing*, 1981, 10:264-270.

[4] Tinetti ME, Speechley M, Ginter SF. “Risk factors for falls among elderly persons living in the community.” *New England Journal of Medicine*, 1988, 319:1701-1707.

[5] Downton JH, and Andrews K. “Prevalence, characteristics and factors associated with falls among the elderly living at home.” *Aging* (Milano), 1991, 3(3):219-28.

[6] Stalenhoef PA et al. “A risk model for the prediction of recurrent falls in communitydwelling elderly: A prospective cohort study.” *Journal of Clinical Epidemiology*, 2002, 55(11):1088- 1094.

[7] “Magnitude of Falls - A Worldwide Overview.” Who Global Report on Falls Prevention in Older Age, World Health Organization, Geneva, 2008, pp. 2–4.

[8] Liu, Junxiu, et al. "Human Body Posture Recognition Using Wearable Devices." *SpringerLink*, Springer International Publishing, 9 Sept. 2019, https://link.springer.com/chapter/10.1007/978-3-030-30493-5_33.

[9] Özgül, Gizem, and Fatma Patlar Akbulut. "Wearable Sensor Device for Posture Monitoring and Analysis during Daily Activities: A Preliminary Study." International Advanced Researches and Engineering Journal, 15 Apr. 2022, https://doi.org/10.35860/iarej.1018977.

[10] Huang, Xiaoping, et al. "A Posture Recognition Method Based on Indoor Positioning Technology." *MDPI*, Multidisciplinary Digital Publishing Institute, 26 Mar. 2019, https://doi.org/10.3390/s19061464.

[11] Beddiar, Djamila Romaissa, et al. "Vision-Based Human Activity Recognition: A Survey - Multimedia Tools and Applications." *SpringerLink*, Springer US, 15 Aug. 2020, https://link.springer.com/article/10.1007/s11042-020-09004-3.

[12] Redmon, Joseph et al. "You Only Look Once: Unified, Real-Time Object Detection". *Arxiv.Org*, 2022, https://doi.org/10.48550/arXiv.1506.02640.

[13] Jun Wu et al. "Skeleton Based Temporal Action Detection with YOLO." *Journal Of Physics: Conference Series*, *1237*(2), 022087. https://doi.org/10.1088/1742-6596/1237/2/022087.

[14] Shinde, Shubham, et al. "YOLO based Human Action Recognition and Localization." *Procedia Computer Science*, *133*, 831-838. https://doi.org/10.1016/j.procs.2018.07.112.

[15] Gul, Malik Ali et al. "Patient Monitoring By Abnormal Human Activity Recognition Based On CNN Architecture". *Electronics*, vol 9, no. 12, 2020, p. 1993. *MDPI AG*, https://doi.org/10.3390/electronics9121993.

[16] O'Shea, Keiron, and Ryan Nash. "An Introduction to Convolutional Neural Networks." *ArXiv.org*, 2 Dec. 2015, https://arxiv.org/abs/1511.08458.

[17] Ren, Shaoqing, et al. "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks." *ArXiv.org*, 6 Jan. 2016, https://arxiv.org/abs/1506.01497.

[18] Culjak, Ivan, et al, "A brief introduction to OpenCV," 2012 Proceedings of the 35th International Convention MIPRO, 2012, pp. 1725-1730.

[19] Bochkovskiy, Alexey, et al. “Yolov4: Optimal Speed and Accuracy of Object Detection.” *ArXiv.org*, 23 Apr. 2020, https://arxiv.org/abs/2004.10934.

[20] Bochkovskiy, Alexey, Darknet: Open Source Neural Networks in Python. 2020. https://github.com/AlexeyAB/darknet.

[21] Yonghye, Kwon. "Yolo Label." 2021 https://github.com/developer0hye/Yolo_Label.

[22] Kanstrén, Teemu. "A Look at Precision, Recall, and F1-Score." Medium, Towards Data Science, 19 May 2021, https://towardsdatascience.com/a-look-at-precision-recall-and-f1-score-36b5fd0dd3ec.

[23] Adarsh, Pranav, et al. "Yolo v3-Tiny: Object Detection and Recognition Using One Stage Improved Model." 2020 6th International Conference on Advanced Computing and Communication Systems (ICACCS), 2020, https://doi.org/10.1109/icaccs48705.2020.9074315.

[24] Redmon, Joseph, and Ali Farhadi. "Yolov3: An Incremental Improvement." ArXiv.org, 8 Apr. 2018, https://doi.org/10.48550/arXiv.1804.02767.

[25] Yohanandan, Shivy. “Map (Mean Average Precision) Might Confuse You!” Medium, Towards Data Science, 9 June 2020, https://towardsdatascience.com/map-mean-average-precision-might-confuse-you-5956f1bfa9e2.

[26] Hui, Jonathan. “Map (Mean Average Precision) for Object Detection.” Medium, Medium, 3 Apr. 2019, https://jonathan-hui.medium.com/map-mean-average-precision-for-object-detection-45c121a31173.

[27] Heriyanto, Joshua Alexander. “Pengenalan Objek Untuk Mendeteksi Cacat Kemasan Pada Sistem Otomatis Berbasis PLC.” Online Catalog | Library@Petra, 2021, https://dewey.petra.ac.id/catalog/digital/detail?id=49872.