Jurnal Elektronika dan Telekomunikasi (JET), Vol. 24, No. 1, August 2024, pp.52-61
Accredited by KEMDIKBUDRISTEK, Decree No: **158/E/KPT/2021**
doi: 10.55981/jet.602

Comparison of YOLOv3-tiny and YOLOv4-tiny in the Implementation Handgun, Shotgun, and Rifle Detection Using Raspberry Pi 4B

Faris Zulkarnain S. Hi. Rauf[a], Djati Handoko[a,*], Ilham S Pradana[a], Dimas Alifta[b]

[a]*Department of Physics, Faculty of Mathematics and Natural Science*
Universitas Indonesia
Pondok Cina, Kecamatan Beji, Kampus UI Depok, 16424
Jawa Barat, Indonesia
[b]*Department of R&D Robotics*
Teknologi Handal Lancar
Gading Serpong Boulevard No.M5/18, Kabupaten Tangerang
Banten, Indonesia

Abstract

Criminal activities frequently involve carriable weapons such as handguns, shotguns, and rifle classes. Frequently, the targets of these weapons that are captured are concealed from plain sight by the people of the crowd. The detection process for these weapons can be assisted by using deep learning. In this case, we intend to identify the model of the firearm that was detected. This research aims to apply one of the deep learning concepts, namely, You Only Look Once (YOLO). The authors use versions of YOLOv3-tiny and Yolov4-tiny for the detection and classification of types of weapons, which are one of the fastest and most accurate methods of object detection, outperforming other detection algorithms. However, both require heavy computer architecture. Therefore, YOLOv3-tiny and YOLOv4-tiny, lighter versions of YOLOv3, can be solutions for smaller architectures. YOLOv3-tiny and YOLOv4-tiny have higher FPS, which is supposed to yield faster performance. Since YOLOv3-tiny and YOLOv4-tiny are modified versions of YOLOv3, the accuracy is improved, and YOLOv3 is already outperforming Faster Single Shot Detector (SSD) and Faster Region with Convolutional Neural Network (R-CNN). The authors employ YOLOv3-tiny and YOLOv4-tiny due to the fact that the Frame Per Second (FPS) and Mean Average Precision (mAP) performance of both approaches are superior in object detection. The study found that YOLOv3-tiny had a high FPS and low mAP performance: an average Intersection over Union (IoU) score of 71.54%, an accuracy of 90%, a recall score of 78%, an F1 score of 84%, and a mAP of 86.7%. While YOLOv4-tiny has low FPS and high mAP: an average IoU score of 73.19%, an accuracy of 90%, a recall score of 84%, an F1 score of 87%, and a mAP of 90.7%.

Keywords: Gun detection, deep learning, YOLO, YOLOv3-tiny, YOLOv4-tiny.

I. INTRODUCTION

Crime using fire weapons is a controversial topic. A state with strict weapon laws does not necessarily have fewer crime incidents related to fire weapons [1]. In Indonesia, the spread of fireweapons in society has become a global phenomenon, which is one of the causes of crimes emerging with fire weapon misuse. Due to the lack of organized supervision of legal or illegal fire weapon civil society ownership, illegal fire weapon ownership is difficult to track, causing the authorized officers not to know the exact numbers of fire weapons spreading in the public. This is what caused crime using fire weapons to happen many times and threaten others' safety. Different kinds of fire weapon misuse happen in the middle of public lives, causing fear and inconvenience in daily activities. According to the statistics, around 500 people pass away every day from gun violence. More than 44% of murder crimes in the world are related to weapon violence. In 2019 alone, more than 250,000 people died as a result of firearms worldwide. Nearly 71% of gun deaths were homicides, about 21% were suicides, and 8% were unintentional firearms-related accidents. A smaller subset of gun deaths occur as the result of mass shootings and school shootings, which are often highly publicized [2]. To overcome crime that uses fire weapons is not easy and takes much time, besides the consciousness of the society about the authority of fire weapon ownership. A part of society has considered that fire weapons are their right of ownership to protect themselves. However, weapons are used for violence more often than for self-defense [3]. Therefore, controlling weaponry crime from illegal weapon ownership requires modern techniques to handle it before a fireweapon crime occurs. Most countries have already used video surveillance systems to monitor people around the crowds to identify terrorism and fire weapon crimes. Although this method is often found inefficient because under human supervision, whose thoroughness is different for every person, even sometimes not always under supervision, detecting fire weapons in surveillance recordings is one of the reasons that affect fire weapon crime. Deep learning is a solution to detect fire weapons in surveillance recording that could be an option for both automated fire weapon

* Corresponding Author.
Email: Djati.handoko@ui.ac.id
Received: November 23, 2023 ; Revised: January 16, 2024
Accepted: February 01, 2024 ; Published: August 31, 2024

detection development and determining detected weapon types.

The identification and detection object field are more advanced from time to time, along with knowledge and technology. Deep learning, or a specific branch of machine learning, is a new way to represent data that pushes into layered learning. As time passed, deep learning developed in detecting objects and classifying object types, a problem involving skills for humans that, long ago, could not be understood by machines. By using deep learning, it can estimate and evaluate objects in an image through classifications and localizations called object detection [4]. The chosen detector for object and face detection included You Only Look Once (YOLO), Fast Region with Convolutional Neural Network (RCNN), and Faster-RCNN. This detector has the next level of precision but light detection in various fields. Each method has its own advantages and shortcomings. Nevertheless, YOLO is one of the fastest and most accurate object detection methods, outperforming other detection algorithms [5].There are a lot of architectures and algorithms available in object detection, such as YOLO and its version [6]–[9]. YOLO is a method developed by Joseph Redmond around 2015. This algorithm was developed to get an automatic object identification process faster and more precisely than the Convolutional Neural Network (CNN), so YOLO has been developed many times to detect objects in real-time. This method is often used in detecting vehicles, people, fruits, and other objects. YOLO is considered to have faster and more accurate architecture. Even though the speed of its detection is fast, YOLO does not have a detection phase in prior, so the mistake of object placement is also big. Besides that, YOLO also has difficulties in detecting small objects that are close to each other. YOLO also has a heavy computer architecture that makes the training process take a long time. YOLOv3-tiny and YOLOv4-tiny version was created or modified from YOLOv3 for lighter, faster, and more efficient architecture than the YOLOv3 version itself. YOLOv3-tiny and YOLOv4-tiny networks have fewer convolutional layers compared to YOLOv3, so the training process is faster and can be used to classify and identify objects. YOLOv3-tiny and YOLOv4-tiny have a Frame Per Second (FPS) and mean Average Precision (mAP) that outperforms each other.

Studies about the usage of the YOLO method have been done many times. The study conducted by Arif Warsi et al. [10] discussed weapon detection using YOLOv3 method. Besides that, to minimize false positive using YOLOv3 method. Estimating the classification model YOLOv3 is based on considering one class to determine handgun, shotgun, and rifle location. In the study, the results of YOLOv3 with VGG16 or RCNN algorithm were compared. The objective of the study is to evaluate the YOLOv3-based detector performance on four different videos and to minimize false positive using YOLOv3 algorithm. The study parameter results are precision, recall and F1 score. The results show the precision ratio on video 1, 2, 3, and 6 YOLO: VGG16 respectively are (98.64%: 88.24%); (87.06%: 98.7%); (41.77%: 62.5%); (96.51%: 82.46%); The recall ratio on videos 1, 2, 3, and 6 respectively are (33.18%: 37.04%); (28.77%: 60.03%); (50.76%: 43.1%); (61.94%: 48.62%). F1 score ratio (YOLO: VGG16) on videos 1, 2, 3, and 6, respectively, are (66.18%: 52.17%); (43.26%: 74.36%); (45.83%: 51.02%); (75%: 61.17%). On that account, YOLOv3 can be said to have better detection performance even on low-quality video compared to RCNN, which is faster. Rana M. Alaqil et al. [11] discussed an automated weapon detection system using Faster R-CNN from an image. They conducted a test of different CNN architectures, Faster R-CNN, YOLOv2, and four feature extractors (ResNet50, Inception-ResNetV2, VGG16, and MobileNetV2). Each Faster R-CNN and YOLOv2 model was implemented using MATLAB and trained using weapon datasets that were shown previously. Two cloud service platforms, Microsoft Azure and Amazon AWS, were used during the model training. The results show mAP validation accuracy for Faster R-CNN (ResNet50) as 73%, Faster R-CNN (Inception ResNetV2) as 81%, Faster R-CNN (VGG16) as 72%, Faster R-CNN (MobileNetV2) as 70%, and YOLOv2 (ResNet50) as 76%. The best mAP was obtained from Faster R-CNN, which used Inception-ResNetV2. For testing time total variations from the entire series of testing also the average time testing per image for the same model resulting a test time and an average test time: MobileNetV2 = 449,10s and 0,74s; ResNet50 = 662,83s and 1,10s; Inception ResNetV2 = 1061,96s and 1,76s; YOLOv2 = 5,5s and 0,0264s; VGG16 = 66,04s and 0,11s. As a result, in terms of training and testing time, YOLOv2 has the shortest time, followed by VGG16, MobileNetV2, ResNet-50, and Inception-ResNetV2 lasts. Yunbin Deng et al. [12] developed a Semantic Embedding (SE) based method for zero-shot gun and fire detection. By using the Contrastive Language-Image Pre-Training (CLIP) model pre-training, the input image and arbitrary text can be mapped into Semantic vectors, and the similarity can be calculated. By defining object classes using a Semantic vector from each class description, very accurate object detection accuracy can be achieved without training any new mode. The detection results obtained accuracy value, False Positive (FP), False Negative (FN), Recall, Precision, and F1 score respectively as YOLO (%): 86.5; 3.3; 10.3; 79.5; 92.4; 85.5, and SE (%): 99.8; 0.2; 0; 99.6; 99.8; 99.7. For weapon detection accuracy value, FP, FN, Recall, Precision, and FI were obtained respectively in YOLO (%): 96.3; 0.9; 2.8; 95.7; 97.3; 96.5, CNN (%): 82.6; 11.2; 6.2; 88.4; 80.9; 84.5, and SE (%): 97.3; 1.7; 1.0; 98; 96.6; 97.3. Hence, even though YOLO already obtained a good result outperforming the CNN method, SE can outperform the YOLO method. However, in this study, the comparison between the SE method with the YOLO-tiny version, which is a modification resulting from YOLO, has not yet been conducted. Marks Dextre et al. [13] discussed weapon detection in real-time using YOLOv5 on Jetson AGX Xavier. The purpose is to train weapon detection systems based on the YOLOv5 series for different datasets. The model was trained by comparing two YOLO series which are YOLOv5 and YOLOv3. They used Jetson AGX Xavier architecture that obtained good precision besides concluding in real-time. The test image was divided into 3 sources: images taken from YouTube, cellphone video camera, and film

images, so a total of 366 images were gathered. Images from the YouTube database was added into JoinDatabase, and 100 epochs were trained in S dan M model. The results show precision percentage from YOLOv5-S, YOLOv5-M, and YOLOv3 respectively as: 99.56%, 99.68%, and 97.30%. YOLOv5-S has a precision value of 98.56%, mAP (0.5) of 99.32%, and mAP (5:95) of 79.65%. YOLOv5-M has a precision value of 98.84%, mAP (0.5) of 97.87%, and mAP (5:95) of 81.89%. Thus, YOLO can successfully detect weapon existence, and YOLOv5 is superior to YOLOv3. Yutra A. Z et al. [14], in the year 2022, discussed automated 2D material detection using YOLOv7. The study focused on the nanotechnology field, which is a two-dimensional material (2D). Because of its unique physical and chemical properties, this material can be applied in various industries, such as sensors, batteries, and display screens. The study suggested an object detection system based on DL using YOLOv7, to automatically search 2D materials with few atomic layers (width between 1 to 13 layers). Furthermore, they measure training model accuracy using Precision, Recall, F1 score, and Average Precision. The test results showed P= 84.3%; R= 92.1%; F1 Score= 88%; and AP= 91.3%. The trained model achieved high accuracy in detecting several layers of MoS_2. Accordingly, YOLO can be said to be successful in detecting several layers of MoS_2 on SiO_2/Si substrate using the YOLOv7 model from a microscopical optic image. Hangyue Zhao et al. [15] in 2023 discussed UAV Maritime image object detection based on YOLOv7 improvement. The purpose of this study is to improve YOLOv7 to detect people, ships, and other objects on open waters in analyzing scenarios captured by maritime drones and in search and rescue missions using YOLOv7. The datasets used in this study are SeaDronesee datasets. However, SeaDronesee datasets have small target characteristics and big ocean surface interference that present a big challenge for general object detectors. Therefore, to overcome this problem, the study suggested a YOLOv7-sea detector that has been improved. Moreover, they integrated a Simple, Parameter-Free Attention Module (SimAM) to find the attention area in the scene. The AP results from YOLOv7-sea are 59.00%, around 7% higher than the baseline model (YOLOv7). Thus, when YOLO is modified, it will result in a better value, the same goes for other modifications of YOLO versions.

In this study, the authors implemented two improvement methods or modifications from YOLO, which are YOLOv3-tiny and YOLOv4-tiny, in detecting fire weapons and determining weapon types. The authors used a Single Board Computer (SBC), a Raspberry Pi model 4B, and a Kiyo camera to obtain data in real-time. The operating system used on Raspberry Pi is Raspbian. Raspbian operation system was made based on Debian, which is one of the distributions from Linux OS. It was expected that YOLOv3-tiny and YOLOv4-tiny could detect and determine weapon types on camera recordings in real-time. It is also expected that it could determine the best mAP and FPS value based on needs because, between YOLOv3-tiny and YOLOv4-tiny, there will be better mAP and loss in terms of FPS, and vice versa.

II. METHOD

The research uses digital image processing techniques to detect weapons and determine their types on camera recordings. The research process of determining the type of weapon using YOLO for real-time detection. The initial phase of this research began with the preparation of images for a dataset of various types of weapons to be labeled, including handguns, shotguns, and rifles. Following this, the image data is compiled in a folder referred to as a dataset, which is subsequently processed. Besides that, the authors also used datasets that are already available on *www.kaggle.com*. The dataset comprises images of weapons, which are subsequently filtered to include only three types: handguns, shotguns, and rifles. The purpose is to create more datasets. In this preprocessing, the image data size modification and data labeling were conducted. Following this, the data will be trained with YOLOv3-tiny and YOLOv4-tiny. The objective of utilizing these two models is to identify the most effective version of the YOLO method for weapon detection and classification. Following this, the accuracy and quality of the model were assessed by comparing the results of the two training models prior to their execution in the detection system. If the training model is already desired, then the model is ready to be implemented for real-time object detection. The research scheme of the weapons detection imaging process can be seen in Figure 1.

A. You Only Look Once (YOLO)

You Only Look Once (YOLO) is an object detection system that is targeted to process real-time and change object detection into single regression matter, processing image pixels directly to be bounding box coordinates and class probability. The YOLO system examined an image only once (you only look once) in order to make a prediction regarding the object's location and identification [6]. Currently, there are different kinds of

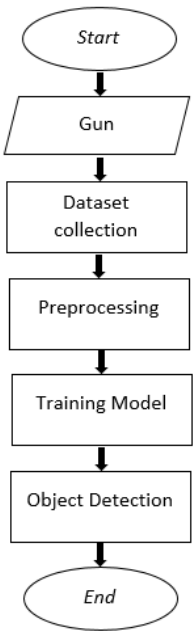


Figure 1. Schematic of Research.

object detection methods which are extensions of the Convolutional Neural Network (CNN) that are commonly used, such as LeNet, AlexNet, ZFNet, GoogLeNet, VGGNet, ResNet, and YOLO. Each method fulfills the fundamental function of CNN in the same way [16]. In object detection, the YOLO method provides the quickest and most precise. YOLO unites separate components of object detection into a single neural network. YOLO utilizes the features of the overall image to predict each bounding box, which can be seen in Figure 2.

YOLO predicts all bounding boxes on all object classes for an image simultaneously. This indicates that YOLO considers the entire portion of the image and every object within it on a global scale. YOLO divides the given image into $S \times S$ grid. Grid cells can correlate with one object and predict a fixed number of boxes. Each box is also given a confidence value. For object classification estimates, grid cells relate to the probability number of classes of the model class represented with C. Whereas $S \times S$ shows the number of grid cells contained by the given image, and B shows the bounding box contained in each grid cell (1). The main theme behind the first version of YOLO or YOLOv1 is to create a single CNN network for prediction [17]. This confidence value indicates both the certainty that the box contains the object and the precision of the predicted box (1).

$$C = Pr(Object) \text{ x } \text{IoU}_{\text{pred}}^{\text{truth}} \quad (1)$$

where, $\text{IoU}_{\text{pred}}^{\text{truth}}$ is the Intersection over Union between the predicted box and the ground truth.

If there are no objects in the cell, the confidence value should be zero. Otherwise, the belief value will be equal to the Intersection of Union (*IoU*) between the predicted box and the background truth box (ground truth). Each bounding box has five predictions: x, y, w, h, and confidence values. Coordinates (x, y) represent the center of the box relative to the bounding of the space cell. Width (w) and height (h) are predicted relative to the overall picture. Finally, confidence prediction will represent the *IOU* between the predicted box and any ground truth box. Each space cell also predicts the probability of conditional class *C, Pr* $(Class_i \mid Object)$. This probability is dependent on the space cell containing the object. YOLO predicts a set of probability classes per space cell, regardless of the number of boxes B. At the testing time, we multiply the probability of the conditional class and the prediction of the confidence of individual boxes, which gives the class-specific confidence value for each box (2). This value encodes the probability of the class that appears in the box and how well the predicted box corresponds to the object.

$$Pr(C_i|Obj) * Pr(Obj) * \text{IoU}_{\text{pred}}^{\text{truth}} = Pr(C_i)\text{IoU}_{\text{pred}}^{\text{truth}} \quad (2)$$

The final predictions are encoded as an $S \times S$ $(B \times 5 + C)$ tensor.

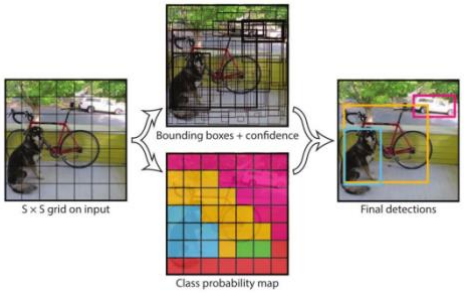


Figure 2. YOLO Model [6].

YOLOv1 has been shown to be able to process images quickly and with high accuracy. However, YOLOv1 still has many shortcomings, one of which has difficulty recognizing small nearby objects caused by the presence of spatial constraints on its bounding box prediction [6]. Then, YOLOv2 and YOLO9000 were able to solve the problems of YOLOv1. YOLOv2 has superior speeds but lower accuracy, whereas YOLO9000 has high precision by being able to recognize 9000 classes of objects. Although it is able to distinguish new types of objects, it is still difficult to learn new categories of objects [6], [7]. YOLOv3 is larger in size than previous versions and is capable of detecting objects with a higher frame rate. YOLOv3 applies logistical regression to its bounding box to better detect objects. In addition, the use of Softmax has been replaced by the Independent Logistic Classifier because Softmax is stated to have no direct influence on performance. In addition, binary cross-entropy loss is also used during training to predict class objects. YOLOv3 uses the Darknet-53 architecture that has the highest size of floating-point operations per second, which means it can make better use of the GPU, making it more efficient and faster. However, behind the higher performance of this previous version, YOLOv3 is more recommended to run on an old detection matrix with 0.5 *IoU* [8]. The square accuracy is usually measured using the *IoU*. The *IoU* calculates the meeting area of the target prediction box and the groundwork box and divides it into their connection area. When evaluating the object detection algorithm, an *IoU* threshold of 0.5 is usually used to determine whether the detection is correct [18]. However, the *IoU* value = 0.5 has a fairly loose area, so it is generally desirable to have an *IoU* greater than 0.5 [19]. Thus, the YOLOv3-tiny and YOLOv4-tiny, a lighter version of YOLOv3, are the solution. The convolution layers in the YOLO-tiny architecture are reduced so the training process can be faster and can be applied to computers that have adequate specifications [8]. Table 1 displays the YOLOv3-tiny feature extractor, while Table 2 exhibits the YOLOv4-tiny feature extractors.

B. Dataset

The image dataset for this study is a handgun, rifle, and shotgun images. Image data is taken manually using webcam. The total number of images in the dataset is 49,230 images in jpg format. In addition to the manually collected data, the study also takes the dataset from Kaggle. The dataset of the Kaggle is a collection of images of weapons, including handguns, rifles, and shotguns, which will then be trained and tested. The number of weapons datasets trained and tested can be seen in Table 3. A total of 44,306 weapon images will be trained, and a total of 4,924 weapon pictures will be

TABLE. 1
YOLOV3-TINY FEATURE EXTRACTOR

Layer	Type	Filters	Size/ Stride	Input	Output
0	Convolutional	64	3×3/1	416×416×3	416×416×16
1	Maxpool		2×2/2	416×416×16	208×208×16
2	Convolutional	32	3×3/1	208×208×16	208×208×32
3	Maxpool		2×2/2	208×208×32	104×104×32
4	Convolutional	64	3×3/1	104×104×32	104×104×64
5	Maxpool		2×2/2	104×104×64	52×52×64
6	Convolutional	128	3×3/1	52×52×64	52×52×128
7	Maxpool		2×2/2	52×52×128	26×26×128
8	Convolutional	256	3×3/1	26×26×128	26×26×256
9	Maxpool		2×2/2	26×26×256	13×13×256
10	Convolutional	512	3×3/1	13×13×256	13×13×512
11	Maxpool		2×2/2	13×13×512	13×13×512
12	Convolutional	1024	3×3/1	13×13×512	13×13×1024
13	Convolutional	256	1×1/1	13×13×1024	13×13×256
14	Convolutional	512	3×3/1	13×13×256	13×13×512
15	Convolutional	255	1×1/1	13×13×512	13×13×255
16	Yolo	-	-	-	-
17	Route 1 3	-	-	-	-
18	Convolutional	128	1×1/1	13×13×256	13×13×128
19	Upsample	-	2×2/2	13×13×128	26×26×128
20	Route 1 9, 8	-	-	-	-
21	Convolutional	256	3×3/1	26×26×384	26×26×256
22	Convolutional	255	1×1/1	26×26×255	26×26×255
23	YOLO	-	-	-	-

tested. The datasets from Kaggle contain a total of 70,800 weapons images.

The image data is then preprocessed, consisting of image size changes and labeling. Image labeling is an early stage in which each image in the dataset is labeled to convey image information. In order to accomplish the labeling procedure, an image of the bounding box and the class name of each object are provided. The file.txt format utilizes the following three class designations for the labeling: "handgun" for the first class, "rifle" for the second class, and "shotgun" for the third class.

C. YOLOv3-tiny and YOLOv4-tiny Training Model

After successfully assembling the images into a single dataset, labeling is done manually with the help of the LabelImg model tool with YOLO darknet format to ".weight". Training design can be seen in Figure 3. If the model is using YOLO ultralitic then the file to be produced is ".pt". The configuration of hyperparameters is shown in Table 4.

At the training stage, the dataset will be included in the train set in the "obj.data" file and use the YOLOv3-tiny and YOLOv4-tiny configurations specified in the yolov3_training.cfg and yolov4_trainning.cfg files that serve as load weights and require trained YOLOv3-tiny and YO LOv4-tiny weights downloaded.

TABLE. 2
YOLOV4-TINY FEATURE EXTRACTOR

Layer	Type	Filters	Size/ Stride	Input	Output
0	Convolutional	32	3×3/2	416×416×3	208×208×32
1	Convolutional	64	3×3/2	208×208×32	104×104×64
2	Convolutional	64	3×3/1	104×104×64	104×104×64
3	Route 2	-	-	-	-
4	Convolutional	32	3×3/1	104×104×32	104×104×32
5	Convolutional	32	3×3/1	104×104×32	104×104×32
6	Route 5 4	-	-	-	-
7	Convolutional	64	1×1/1	104×104×64	104×104×64
8	Route 2 7	-	-	-	-
9	Maxpool	-	2×2/2	104×104×128	52×52×128
10	Convolutional	128	3×3/1	52×52×128	52×52×128
11	Route 10	-	-	-	-
12	Convolutional	64	3×3/1	52×52×64	52×52×64
13	Convolutional	64	3×3/3	52×52×64	52×52×64
14	Route 13 12	-	-	-	-
15	Convolutional	128	1×1/1	52×52×128	52×52×128
16	Route 10 15	-	-	-	-
17	Maxpool	-	2×2/2	52×52×256	26×26×256
18	Convolutional	256	3×3/1	26×26×256	26×26×256
19	Route 18	-	-	-	-
20	Convolutional	128	3×3/1	26×26×128	26×26×128
21	Convolutional	128	3×3/1	26×26×128	26×26×128
22	Route 21 20	-	-	-	-
23	Convolutional	256	1×1/1	26×26×256	26×26×256
24	Route 18 23	-	-	-	-
25	Maxpool	-	2×2/2	26×26×512	13×13×512
26	Convolutional	512	3×3/1	13×13×512	13×13×512
27	Convolutional	256	1×1/1	13×13×512	13×13×256
28	Convolutional	512	3×3/1	13×13×256	13×13×512
29	Convolutional	21	1×1/1	13×13×512	13×13×21
30	YOLO	-	-	-	-
31	Route 27	-	-	-	-
32	Convolutional	128	1×1/1	13×13×256	13×13×128
33	Upsample	-	2×2/2	13×13×128	26×26×128
34	Route 33 23	-	-	-	-
35	Convolutional	256	3×3/1	26×26×384	26×26×256
36	Convolutional	21	1×1/1	26×26×256	26×26×21
37	YOLO	-	-	-	-

Once the training process is successful then the training results will be saved in the form of a weight file. The weight storage process will start from 0, then on the first 10,000 iterations to the final weight file named "GUN_cnfg_v3tiny-416x416-2506final.weights". This best weight file with the file name "GUN_cnfg_v3tiny-

TABLE. 3
GUN DATASET

Gun Dataset	Total Gun Images
Training	44,306
Testing	4,924

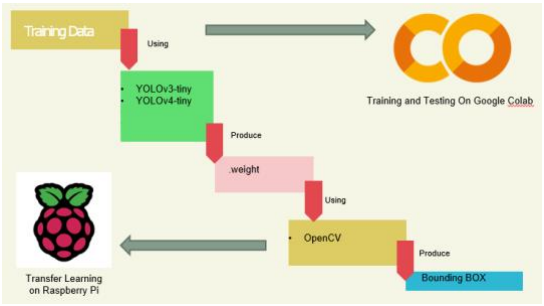


Figure 3. Training and detection design.

416x416-2506_best.weights" will be used for detection on static images and real-time detection. Figure 4 shows examples of weight files stored on Google Drive. The last step is using a transfer learning program like OpenCV DNN. OpenCV DNN for YOLO model reading.

D. YOLOv3-tiny and YOLOv4-tiny Object Detection

After completing the training process on the Google Colaboratory platform and obtaining the ".weight" file, the next step is a real-time weapon detection validation program. Programming results for the YOLOv3-tiny and Yolov4-tiny training frameworks are inserted via the ".weight" file. Python is utilized as the programming language. The validation process is executed on the Raspberry Pi model 4B Single Board Computer (SBC) via a VNC that acts as an interface between monitors and the 4B. The Raspberry Pi is connected to a Kiyo camera for the purpose of capturing data in real time. In order to initiate the validation procedure, it is necessary to download the YOLOv3-tiny and Yolov4-tiny model endweight files, the.cfg file comprising the YOLOV3-tny and YOLOV4-tny network hyperparameter configuration, and the dataset files that have been previously processed. This validation process is conducted on Anaconda3 using Jupyter Notebook.

To see the validation results of both frameworks, rely on the confusion matrix values (true positive (TP), true negative (TN), false positive (FP), and false negative (FN), as well as the accuracy, precision, recall, and mean average precision (mAP) that have been obtained. An improved outcome is indicated by a higher mAP presentation and F1-Score score. The results of the detection and prediction tests are a real-time bounding box with the label of the name of the class of the weapon, the percentage or confidence value, and the FPS value. The accuracy of weapon detection test results and weapon classification is enhanced as the percentage of detection and FPS increase.

III. Result and Discussion

Once the training process is fully concluded, the YOLOv3-tiny network performance test is executed to assess the model's real-time detection capabilities. The performance of the YOLOv3-tiny and YOLOV4-tiny networks in data testing will be evaluated by calculating the confusion matrix values (TP, FP, TN, and FN), accuracy, precision, recall, F1 score, and mAP. The performance test will be performed on some frames of the video of the test data taken in real-time using the Kiyo camera. We conducted the detection ourselves by

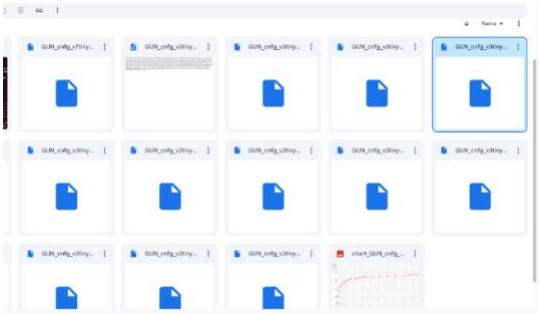


Figure 4. Saved Weight Files

TABLE. 4
CONFIGURATION OF YOLOV3-TINY AND YOLOV4-TINY

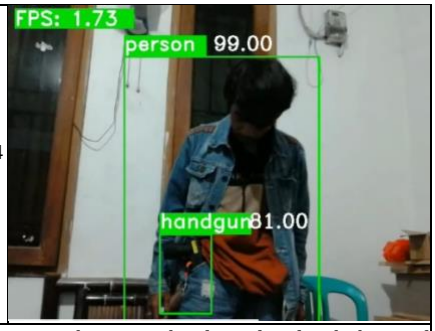
Type of Configuration	Value
Class	3
Max_Batch	100,000
Filter YOLO	30

carrying replicas of handguns, shotguns, and rifles, each of which contained one to two varieties of weapons, and by placing samples of weapons and humans inside. In this test, a specified distance of 2 meters from the camera was used. The lighting at the time of the test was not very careful and came only from the terrace house lights at night. The test results can be seen in Table 5. The output is obtained on the monitor screen: if a green bounding box appears, it detects the person, weapon, and type of weapon. The white description indicates the percentage of the confidence value.

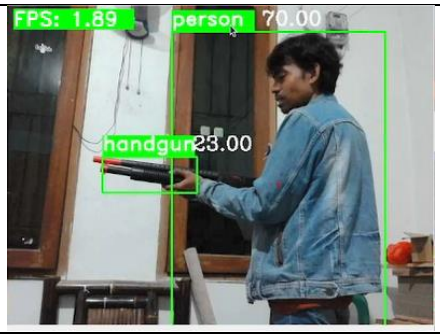
The Raspberry Pi terminal also exhibits the TP, FP, and FN values. These values are subsequently converted into a confusion matrix for YOLOv3-tiny and Yolov4-tiny, as depicted in Figure 5 and 6, respectively. Predict 1 (true) and Actual 1 (true) indicate the value of TP, Predict 1 (true) and Actual 0 (false) indicate the FP value, and Predict 0 (false) and Actual 1 (true) indicate the FN value. For YOLOv3-tiny, the value of TP is obtained a total of 1,779, FP = 194, and FN = 500. As for YOLOv4-tiny, the value of TP is 1,903, FP = 206, and FN = 376.

TABLE. 5
REAL-TIME TESTING RESULT

Actual	Result	
Handgun	YOLOv3-tiny	

	YOLOv4 -tiny	
	At this detection, there was a handgun placed in the bag, and it was detected and identified with confidence values on YOLOv3-tiny of 72% and FPS 1.94. Whereas on Yolov4-tiny, it was 81% and FPS 1.73.	
Shotgun	YOLOv3 -tiny	
	YOLOv4 -tiny	
	This detection involved a shotgun, which was identified with a 97% confidence value and FPS of 1.92 on YOLOv3-tiny, and with a 99% confidence value and FPS of 1.73 on YOLOv4-tiny.	
Rifle	YOLOv3 -tiny	
	YOLOv4 -tiny	
	At this detection, a rifle was detected and identified with 90% assurance on YOLOv3-tiny and 1.94 FPS, compared to 99% assurance and 1.73 FPS on YOLOv4-tiny.	

Handgun #2	YOLOv3 -tiny	
	YOLOv4 -tiny	
	During this detection, a handgun was identified and detected with 79% confidence on YOLOv3-tiny and 1.93 FPS, compared to 93% confidence and 1.73 FPS on YOLOv4-tiny.	
Shotgun, Handgun, and rifle	YOLOv3 -tiny	
	YOLOv4 -tiny	
	At this detection there were shotguns and handguns that were deployed very quickly and were successfully detected and identified with the confidence of the shotgun on YOLOv3-tiny of 48%, handgun of 37% and FPS of 1.93. whereas on Yolov4-tiny it was 74% for shotgun, 95% for handgun, and FPS of 1.73.	
Rifle and Handgun (YOLOv3 -tiny)		
	Pocketed rifles and handguns were discovered during this detection, with a handgun scoring 56% and the rifle receiving a 56% confidence rating.	

Shotgun (YOLOv3 -tiny)		
	A shotgun was detected successfully but failed to be classified as a "handgun" at this time due to the fact that the camera captured only a portion of the shotgun's form or there was an insufficiency of training datasets.	
Shotgun, Handgun, and Rifle	YOLOv3 -tiny	
	YOLOv4 -tiny	
	At this detection, there were shotguns, handguns, and rifles that were successfully detected and classified.	

In addition, the Raspberry Pi Terminal also displays the Precision, Recall, F1 score, average IoU, and mAP values. The parameter values for YOLOv3-tiny and Yolov4-tiny are presented in Table 6.

A comparison graph depicting the mAP and loss values of the YOLOv3-tiny and Yolov4-tiny models utilized in this investigation is illustrated in Figures 7 and 8. The horizontal axis of the graph indicates the number of times a YOLO model has been trained, while the vertical axis represents the loss value. The blue line represents the loss value, which decreases as the amount of training increases (the lower the value, the better). The

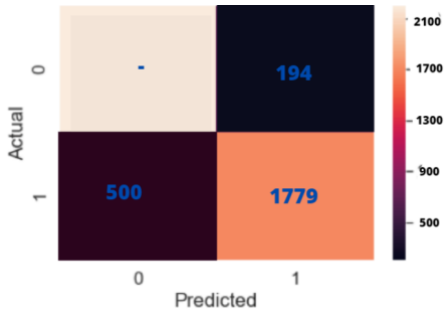


Figure 5. Confusion Matrix of YOLOv3-tiny.

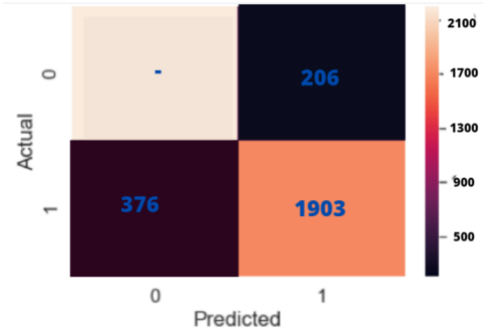


Figure 6. Confusion Matrix of YOLOv4-tiny.

TABLE. 6
THE PERFORMANCE OF YOLOV3-TINY AND YOLOV4-TINY

Model	Avg. IoU (%)	Precision (%)	Recall (%)	F1 Score (%)	mAP 0.5 (%)	FPS
YOLOv3-tiny	71,54	90	78	84	86.7	0.62 s
YOLOv4-tiny	73.19	90	84	87	90.7	0.95 s

red line indicates the mAP value, which increases with the increasing number of training until it merges to a constant value and becomes flat (the higher the value, the better). The optimal results obtained when the YOLOv3-tiny and Yolov4-tiny model mAP values were applied to the Raspberry Pi in this study indicate their implementation success.

For comparison, to evaluate the performance of the full-size YOLOv3-tiny model applied to the Raspberry Pi, after 100,000 iterations, the mAP value was 86.7%, while the performance for the full-size yOLOv4-tiny models applied on the RasPberry Pi, after 100,000 iterations, the mAP value was 90.7%, which represents an increase of 4% of the mAP value. Loss comparison for YOLOv3-tiny has an average loss value of 0.7107, and for YOLOv4-tiny has an average loss value of 0.4375. Thus, the graph shown shows that the number of iterations is closely related to the high loss of the object.

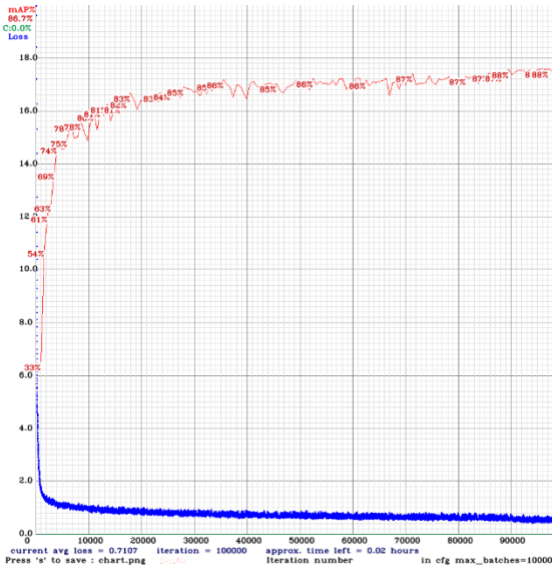


Figure 7. Loss Graph and mAP of YOLOv3-tiny.

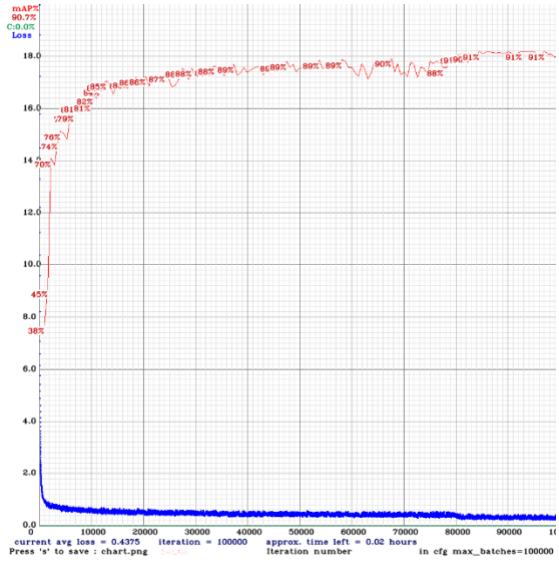


Figure 8. Loss Graph and mAP of YOLOv4-tiny.

Based on the graph, the higher the number of epochs, the lower the trend of loss of objects detected during training.

IV. CONCLUSION

In this research, the efficacy of YOLO in object detection and classification for weapon categories was demonstrated. The confusion value of the matrix and mAP of the YOLOv4-tiny framework is better than that of the YOLOv3-tiny framework. YOLOv3-tiny frameworks have an accuracy of 90% or 0.9 which means they have almost perfect values (equal to 1), recall or overall detection rate of 78%, F1 score of 84%, mAP of 86.7%, and average loss of 0.7107. While YOLOv4-tiny Frameworks has a precision of 90% or 0.9 which means it has almost perfect values (equal to 1), recall or overall detection rate of 84%, F1 score of 87%, mAP of 90.7%, and average loss of 0.4375. As a result, the value of real-time network performance tests is remarkably high.

This study demonstrates that the system exhibited the capability to identify multiple weapons and classify them accordingly, even in situations where the weapons were partially obscured by other objects or only partially visible. Model B Raspberry Pi 4 is already capable of resolving the YOLOv3s issue with its higher computer specifications. However, an emerging issue arises in the form of the system's continued inability to detect weapons when objects are in motion at remarkable speeds. This can be enhanced by using a Raspberry Pi model or a more sophisticated microcontroller. Furthermore, the system's performance will be enhanced by incorporating additional datasets or enhancing the image quality of datasets related to the shotgun class.

This study also concluded that although YOLOv3-tiny is not superior to YOLOv4-tiny in the detection of mAP values, it is superior in the speed of FPS values compared to YOLOv4-tiny. The FPS of YOLOv3-tiny is documented to be 1.9, while that of YOLOv4-tiny is 1.7. This gap can be attributed to the lighter computer architecture of YOLOv4-tiny, which consequently enables for a quicker detection time. Therefore, the system performance of both of these frameworks is already quite excellent, surpassing their respective mAP and FPS values, depending on whether the user is more concerned with speed or precision.

DECLARATIONS

Conflict of Interest

The authors have declared that no competing interests exist.

CRediT Authorship Contribution

Faris Zulkarnain S. Hi. Rauf: Conceptualization, Methodology, Writing - Original Draft, Writing - Reviewing & Editing, Hardware and Software Data Curation; Djati Handoko: Conceptualization, Methodology, Supervision, Validation, Resources; Ilham S Pradana: Formal Analysis; Dimas Alifta: Hardware and Software Data Curation.

Funding

Research reported in this publication was supported by Dr. Djati Handoko as head of Physics Department University of Indonesia.

Acknowledgement

The authors would like to thank University of Indonesia that support the completion of this research.

REFERENCES

[1] J. Celentano and E. Abdelfattah, "Analyzing Gun Violence in the United States," in *2020 11th IEEE Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON)*, 2020, pp. 258–261. doi: 10.1109/UEMCON51285.2020.9298154.

[2] "2023 World Population Review. Gun Violence," in *Amnesty International*, 2023. [Online]. Available: https://www.amnesty.org/en/what-we-do/arms-control/gun-violence/

[3] D. Hemenway and M. Miller, "Gun Threats Against and Self-defense Gun Use by California Adolescents," *Arch. Pediatr. Adolesc. Med.*, vol. 158, no. 4, pp. 395–400, Apr. 2004, doi: 10.1001/archpedi.158.4.395.

[4] Z. Q. Zhao, P. Zheng, S. T. Xu, and X. Wu, "Object Detection with Deep Learning: A Review," *IEEE Trans. Neural Networks Learn. Syst.*, vol. 30, no. 11, pp. 3212–3232, 2019, doi: 10.1109/TNNLS.2018.2876865.

[5] W. Fang, L. Wang, and P. Ren, "Tinier-YOLO: A Real-Time Object Detection Method for Constrained Environments," *IEEE Access*, vol. 8, pp. 1935–1944, 2020, doi: 10.1109/ACCESS.2019.2961959.

[6] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 2016, pp. 779–788, Dec, 2016, doi: 10.1109/CVPR.2016.91.

[7] J. Redmon and A. Farhadi, "YOLO9000: Better, faster, stronger," in *Proc. - 30th IEEE Conf. Comput. Vis. Pattern Recognition, CVPR*, Jan, 2017, vol. 2017, pp. 6517–6525, doi: 10.1109/CVPR.2017.690.

[8] J. Redmon and A. Farhadi, "YOLOv3: An Incremental Improvement," 2018, [Online]. Available: http://arxiv.org/abs/1804.02767

[9] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "YOLOv4: Optimal Speed and Accuracy of Object Detection," 2020, [Online]. Available: http://arxiv.org/abs/2004.10934

[10] A. Warsi, M. Abdullah, M. N. Husen, M. Yahya, S. Khan, and N. Jawaid, "Gun Detection System Using Yolov3," in *2019 IEEE International Conference on Smart Instrumentation, Measurement and Application (ICSIMA)*, 2019, pp. 1–4. doi: 10.1109/ICSIMA47653.2019.9057329.

[11] R. M. Alaqil, J. A. Alsuhaibani, B. A. Alhumaidi, R. A. Alnasser, R. D. Alotaibi, and H. Benhidour, "Automatic Gun Detection From Images Using Faster R-CNN," in *2020 First International Conference of Smart Systems and Emerging Technologies (SMARTTECH)*, 2020, pp. 149–154. doi: 10.1109/SMART-TECH49988.2020.00045.

[12] Y. Deng, R. Campbell, and P. Kumar, "Fire and Gun Detection

Based on Sematic Embeddings," in *2022 IEEE International Conference on Multimedia and Expo Workshops (ICMEW)*, 2022, pp. 1–4. doi: 10.1109/ICMEW56448.2022.9859303.

[13] M. Dextre, O. Rosas, J. Lazo, and J. C. Gutiérrez, "Gun Detection in Real-Time, using YOLOv5 on Jetson AGX Xavier," in *2021 Wavelet Active Media Technology and Information Processing (ICCWAMTIP)*, 2022, pp. 1–5. doi: 10.1109/ICCWAMTIP56608.2022.10016569.

[15] H. Zhao, H. Zhang, and Y. Zhao, "YOLOv7-sea: Object Detection of Maritime UAV Images based on Improved YOLOv7," in *2023 IEEE/CVF Winter Conference on Applications of Computer Vision Workshops (WACVW)*, 2023, pp. 233–238. doi: 10.1109/WACVW58289.2023.00029.

[16] A. Saxena, "An Introduction to Convolutional Neural Networks," *Int. J. Res. Appl. Sci. Eng. Technol.*, vol. 10, no. 12, pp. 943–947, 2022, doi: 10.22214/ijraset.2022.47789.

XLVII Latin American Computing Conference (CLEI), 2021, pp. 1–7. doi: 10.1109/CLEI53233.2021.9640100.

[14] Y. A. Zenebe, L. Xiaoyu, W. Chao, W. Yi, H. A. Endris, and M. N. Fanose, "Towards Automatic 2D Materials Detection Using YOLOv7," in *2022 19th International Computer Conference on*

[17] I. Khurram, M. M. Fraz, M. Shahzad, and N. M. Rajpoot, "Dense-CaptionNet: a Sentence Generation Architecture for Fine-grained Description of Image Semantics," *Cognit. Comput.*, vol. 13, no. 3, pp. 595–611, 2021, doi: 10.1007/s12559-019-09697-1.

[18] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The Pascal Visual Object Classes (VOC) Challenge," *Int. J. Comput. Vis.*, vol. 88, no. 2, pp. 303–338, 2010, doi: 10.1007/s11263-009-0275-4.

[19] C. L. Zitnick and P. Dollár, "Edge Boxes: Locating Object Proposals from Edges BT - Computer Vision – ECCV 2014," 2014, pp. 391–405.