

YOLOv8 Based Early Land Fire Detection System Integrated with Telegram

Muhammad Rif'ad Affandi¹, Rieke Adriati Wijayanti^{2*}, Putri Elfa Mas'udia³

^{1,2} Digital Telecommunication Network Study Program, Department of Electrical Engineering, State Polytechnic of Malang, 65141, Indonesia.

³ Telecommunication Engineering Study Program, Department of Electrical Engineering, State Polytechnic of Malang, 65141, Indonesia.

¹ipankhatame@gmail.com ²riekeaw@polinema.ac.id ³putri.elfa@polinema.ac.id

Abstract— This early land fire detection system uses a Raspberry Pi 4B device integrated with RGB- and HSV-based color filtering methods, as well as a target mask module. The Raspberry Pi system connects to Google Drive via an OAuth client to store detection results. Additionally, a device running the YOLOv8 algorithm is used for validation to determine whether detected objects are fires or not. The validation results from YOLOv8 are then sent to Telegram via a chat bot in daytime testing, the system used a confidence threshold of 0.25 and achieved a total confidence of 43.6%, an average transmission time of 5730.30 ms (approximately 5.73 seconds), and an accuracy rate of 70% for the captured images. Meanwhile, in nighttime testing, the accuracy increased to 86%, with an average confidence of 49.07% and an average detection time of 115.20 ms. However, the YOLOv8 algorithm can still produce false negatives if the test data or model training data lacks sufficient variation.

Keywords— *Raspberry Pi, Flame, Image Processing, YOLOv8, Telegram.*

I. INTRODUCTION

A bushfire is a fire that takes place in the wild, but can also destroy nearby homes and farmland. Some of the common triggers for fires include lightning, dry land, human carelessness, and setting fires by accident[1]. Between 2002 and 2016, an estimated 4,225,000 km² of land was burned due to uncontrollable fires. These events significantly affect wildlife and human life [2].

Early detection of forest fire is beneficial to reduce fire losses and protect the ecological environment. Various methods such as manual inspection, ground sensor monitoring, and satellite remote sensing have traditionally been employed for fire detection[3].

IoT is very broad and can be integrated with various applications and devices, such as the telegram application. The bot account functions as an interface for code that is run on the chatbot application developer's server. Telegram supports secure text, audio, video, and image messaging[4]. In Additional IoT adoption also faces significant cost challenges in the construction and operation of communication infrastructure, especially when seeking to develop networks on a smaller scale. Nevertheless, this has prompted markets in developing countries to offer affordable IoT connectivity services. This challenge is a major focus for many organizations and companies that are trying to implement IoT technology, especially in terms of infrastructure development cost efficiency and increased accessibility to IoT connectivity across a wide geographical area[5].

Digital Image Processing is a field of science that studies how an image is formed, processed, and analyzed so as to produce information that can be understood by humans. Digital Image Processing is a series of attempts to transform an image or image into another image using certain techniques. An

image is a two-dimensional image that can be displayed on a computer screen as a set or discrete digital value called a pixel or image element. Mathematically, image is a continuous function of the intensity of light on a two-dimensional plane[6].

Using google drive as an intermediary the use of cloud storage services for electronic data exchange offers the potential for greater efficiency in data management and can reduce data storage costs[7].

Image processing-based using method RGB image data is typically converted into HSV color space to isolate fire-like characteristics[8]. YOLOv8 model shows significant improvements over previous models. A significant improvement is the shift from an anchor-based strategy to an anchor-free strategy for bounding box regression

. The anchor-free strategy abandons the use of anchor boxes and instead uses a probability regression method based on the midpoint for bounding box regression[9].

Based on test results conducted at two different times—daytime and nighttime—the system uses a confidence threshold of 0.25. In the daytime test, an average confidence of 43.6% was achieved, with an average data transmission time of 5730.30 ms, or approximately 5.7 seconds. From the 15 test images, an accuracy of 70% was achieved, with 14 true positives and 6 false positives. These results indicate that the system is capable of detecting signs of fire during the day, although there are still some false detections of non-fire objects.

During night-time testing, the accuracy value increased to 86.67%, with an average detection time of 115.20 ms. From the 15 test images taken at night, there were some instances when the YOLO algorithm did not detect objects that should have been classified as fire. These conditions include false negatives, where the fire object is not detected by the model. In

fact, at night, the fire and the background have a very stark color contrast. This error is likely due to a lack of variation in the training data for the fire and non-fire classes, meaning that fires with colours similar to white are not recognized as fire objects by the YOLO algorithm.

Other journals report that detection accuracy decreases with distance, and camera height can reduce nearby object accuracy. FFYOLO highlights fire detection as a multi-source RGB data fusion process.

Overall, the device has been able to fulfil its main purpose as an early fire detection system. The system can detect indications of fire, validate them using YOLO, and send the detection results as visual information. With this system in place, fires are expected to be detected earlier, helping to prevent larger fires.

II. METHOD

Fig.1 shown, this monitoring system is designed to monitor the land area which is divided into 2 different areas, namely area 1 and area 2, with an image processing system. The main purpose of this system is to detect environmental conditions in real-time and provide early warning of potential land fires. The system utilizes a Raspberry Pi, which runs a deep learning model using the YOLOv8 algorithm.

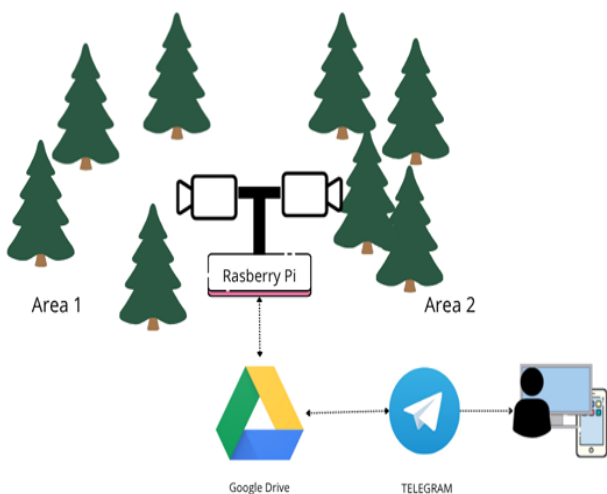


Figure 1 Research Scheme

The camera captures image frames, and the Raspberry Pi processes these using RGB threshold filtering to assist in detecting fire-like features. Once detected, the image data is stored in a local database. From there, the processed image that detected fires are sent to the Telegram application, allowing users to monitor both areas remotely in real time.

1) Block Diagram

Fig.2 illustrates the system architecture for real-time land fire detection. The system uses two cameras, Camera 1 and Camera 2, to capture image frames from two different

monitored areas. These image frames are then sent to the Raspberry Pi, which functions as the central processing unit.

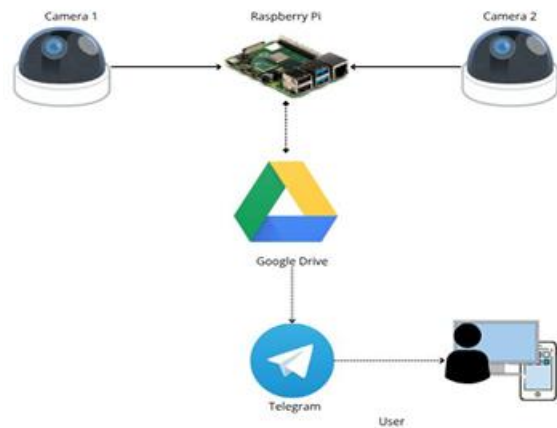


Figure 2 Block Diagram

On the Raspberry Pi, the captured frames are processed using a YOLOv8 deep learning algorithm along with RGB threshold filtering to identify potential fire. If fire is detected, the processed data is stored temporarily in Firebase, which acts as the cloud-based database and intermediary for data synchronization. From Firebase, the system integrates with the Telegram application to send real-time notifications—including sensor status or fire detection alerts—to users. The captured images from the USB camera detection will be stored on the SD Card as a backup and uploaded to a Google Drive folder using an ID that is utilized to identify and access specific items via the Google Drive API[10]. Application Programming Interface (API) is a set of rules within software that enables access to or use of services and resources owned by other software that have been integrated into that API. The benefit of an API is that it serves as a bridge between two different software systems[11].

2) Flowchart System

The flowchart can be seen in Fig. 3 which can be explained. When the frame captures fire by color filtering RGB and HSV. Masking takes the frame that not indicates fire by colour filtering then will be immediately the captured image will be sent to Google Drive and YoloV8 will predict that is true fire or not if detected fire by yolo then user will get notification from telegram. Fig.3 shows the overall flow of how the system works, from the initial input to the generation of the output in the telegram (user).

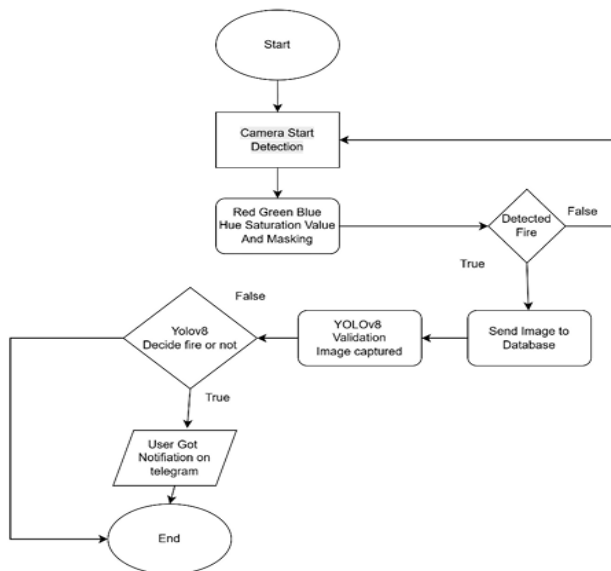


Figure 3 Flowchart System

By completing the processing of the frame, the captured image will be sent to and Telegram will take the image in Google drive so that it can appear on the user. If the yolov8 algorithm doesn't consider object it will pass and detect next object on database.

3) Image Processing Workflow

The image processing begins when Camera 1 and Camera 2 capture real-time frames in RGB format, which is the standard colour space produced by most digital image sensors. These RGB images serve as the direct input to the YOLOv8 deep learning model for object detection, where potential fire regions are identified. In parallel, the same RGB image is also converted into the HSV (Hue, Saturation, Value) colour space to isolate fire-like colour characteristics using threshold filtering. The result from both the YOLOv8 detection and HSV filtering is then processed by the target mask module, which combines and highlights the most probable fire areas. The system generates four output visuals—original image, RGB input, HSV conversion, 39 and target mask—which are combined and sent via Telegram bot as a real-time alert to the user.

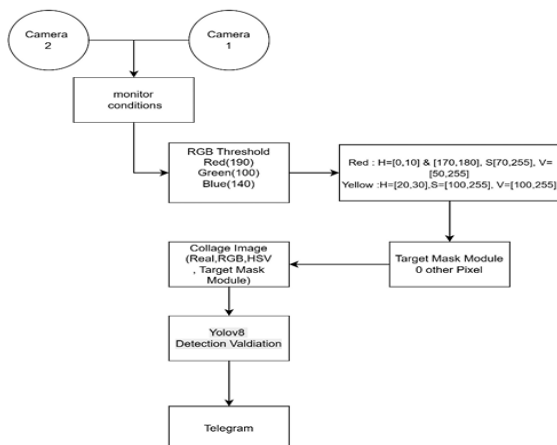


Figure 4 Flowchart Fire detection

On Fig.4 is workflow of the program. First is The input being an image captured by RGB and HSV. The process involves The remaining pixels are assigned a value of 0 (black). This masking makes it easier to isolate objects from the background, while also serving as the basis for the next segmentation stage. With the mask in place, quantitative analysis can be performed more accurately, such as calculating the percentage of a color area or the average HSV value[28]. Red Green Blue (RGB) is a color space used in digital image processing to represent colors based on the red, green, and blue (RGB) components.

4) Colour Filtering

The image classification problem is one of the challenging tasks in the image processing domain. In the past, traditional image processing techniques utilized RGB channel comparison to detect different objects, such as fire in frames. These traditional methods are not free of errors and are not fully reliable. For instance,

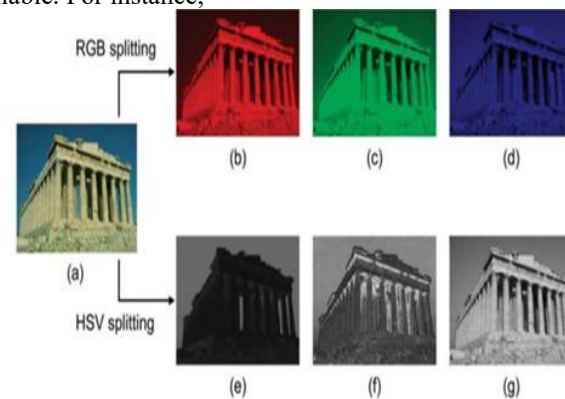


Figure 5 Red Green Blue & Hue, Sue, Value

RGB value comparison methods that usually consider a threshold value to detect fire may detect sunset and sunrise as a false positive outcome [12]. Hue, Saturation, Value). First, Hue (H) is the type of colour (e.g. red, green, or yellow). Hue represented as angular degrees whose values range from 0° until 360° (although for some applications it is normalized from 0% to 100 to 100%). Second, Saturation (S) is represented as the distance from the bright black-white axis. Its value ranges from 13 0% to 100%. And the last one is Value (V), which is at the height of the axis represented as black-white. Possible distance values range from 0% to 100%. A value of 0 is always black. Based on saturation, 100 can be white or more or less saturation[13] .

5) You Only Look Once Algorithm

The YOLOv8 model manifests significant enhancements over its predecessors. A significant advancement is the shift from the anchor-based strategy to the anchor-free strategy for bounding box regression. Anchor-free strategy abandons the use of anchor boxes, instead employing a method of probability regression based on the centre point for bounding box regression.

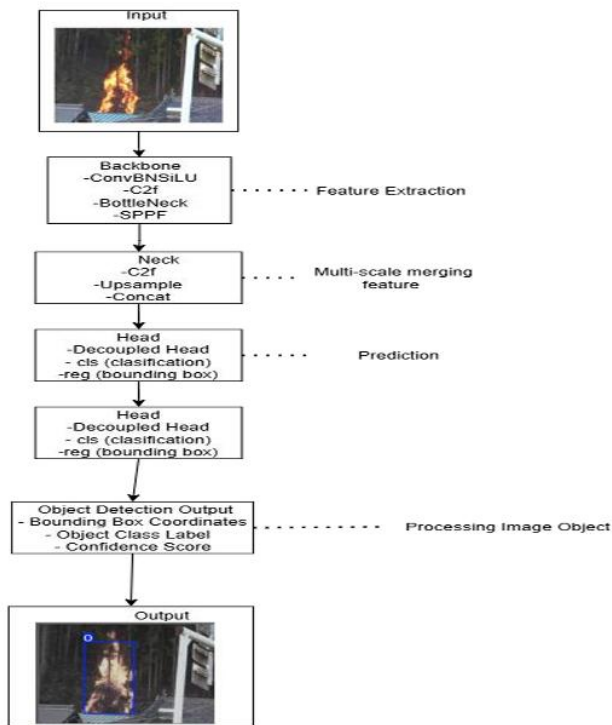


Figure 6 YOLOv8 Structure workflow

The anchor-based strategy performs well in fixed datasets or scenes with consistent object distributions. However, its performance may falter in complex forest fire detection environments. In contrast, the anchor-free strategy is more suitable for forest fire detection parameters[14].

6) Raspberry Pi

Using The Raspberry Pi is used to run office programs, computer games, and as a media player for high-resolution videos. Its performance is comparable to that of a desktop system. Memory: It offers RAM options of 1GB, 2GB, or 4GB LPDDR4-2400 SDRAM. Connectivity: Dual-band 802.11ac wireless networking, 2.4 GHz and 5.0 GHz IEEE 802.11ac wireless[15].

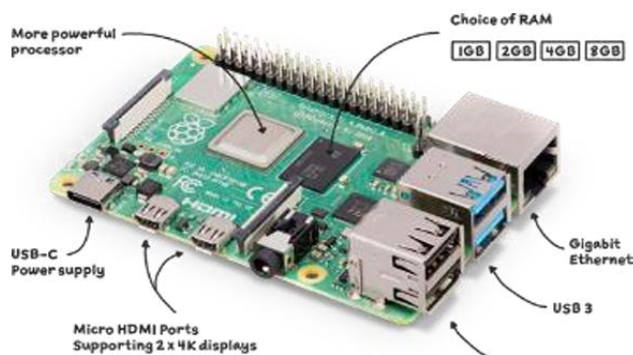


Figure 7 Raspberry Pi 4 Model B

The Raspberry Pi 4 Model B display shown in Fig.7 is adopted in this work as an intelligent device with the Raspbian open-source operating system. Program writing is done using Python

programming language, with other python libraries such as Natural Language tool kit, OpenCV and Teleport. The Natural Language toll kit executes commands sent through the Telegram bot, while OpenCV serves as a framework for synthesizing machine learning-based algorithms and computer vision applications[16].

III. RESULTS AND DISCUSSION

This research was conducted on “Jalan Warinoid Dalam, RT 04/RW 14”, on a 7 x 11 meter² plot of land. It features a variety of trees, including mango, guava, and durian trees, as well as cassava and various non-tree plants. The plants in this area range in size from small to large, making it ideal for conducting experiments.



Figure 8 Place Condition Land for research

This location is considered strategically advantageous, as it is supported by a reliable internet connection that is accessible in the area. Fig.8 shows the actual condition of the orchard, including the arrangement of the plants and the environmental characteristics. The research was conducted from February to May, covering testing, simulation, and data collection.

1) Time Detection to notify

The required transmission time is shown in Fig.9. The graph indicates that most of the data transmission times are relatively stable, while a significant spike occurs at the second data point. This spike was caused by an internet connection outage during the transmission process, which increased the upload time.

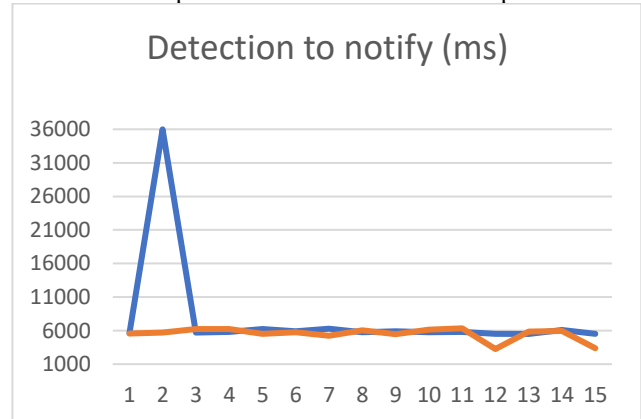


Figure 9 Detection to ms graph

Therefore, the stability of the internet connection plays an important role in the performance of the early fire detection system. To calculate the average, we use the following formula:

$$\text{Mean} = \frac{\text{Total data day} + \text{Total data night}}{\text{Total data}}$$

And the data will be obtained by dividing the total amount of data collected during the day and night into:

$$\text{Total} = \frac{117325,64 + 82586,99}{30} = 6663,75 \text{ ms}$$

And the average of the data obtained from the daytime and nighttime studies is 6,663.75 ms, or seconds.

2) Processing Time Colour Filtering and yolov8

The detection process tends to be very fast, both during the Raspberry Pi detection stage and during the validation process using the YOLO algorithm. This indicates that the system is capable of performing the detection process in a relatively short amount of time. By calculating the average as in the previous step using the average formula:

$$\text{Mean} = \frac{\text{Total data day} + \text{Total data night}}{4}$$

$$\text{Mean} = \frac{34,75 + 91,45 + 31,16 + 96,98}{4}$$

$$\text{Mean} = 65,5822 \text{ ms}$$

So, the result is 65.5822 milliseconds, or 0.06 seconds. Detection based on color filtering and target masks works quite well in capturing indications of fire color, while YOLO is used to validate the detected objects. However, based on the test results, there are still some conditions where the YOLO model is not yet able to recognize fire objects optimally.

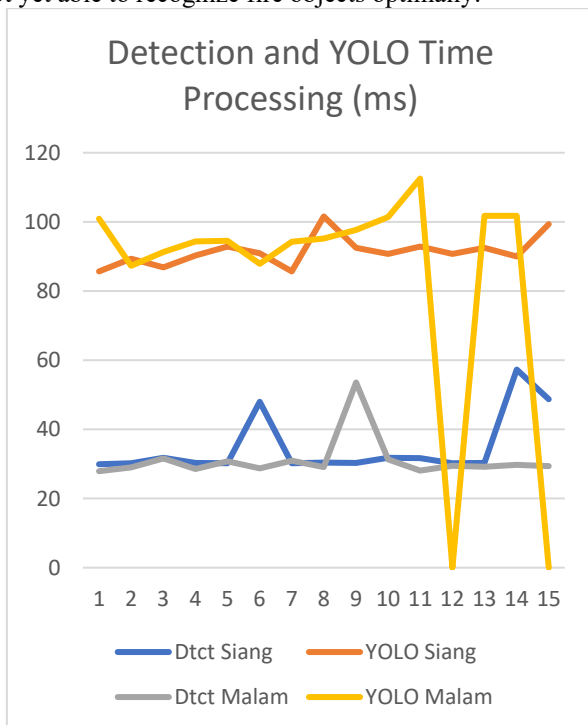


Figure 10 Detection and YOLO Graph

This indicates that the model still has limitations, primarily due to a lack of training data variation. Therefore, adding a dataset with variations in fire shape, fire color, object position, camera distance, and lighting conditions is necessary for the YOLO model to improve its detection capabilities

3) Confidence Rate Algorithm YOLOv8

The average confidence score shown on fig.11 falls into the moderate category because the system uses a confidence threshold of 0.25. This threshold allows the model to continue detecting objects with low confidence scores as long as they exceed this minimum threshold

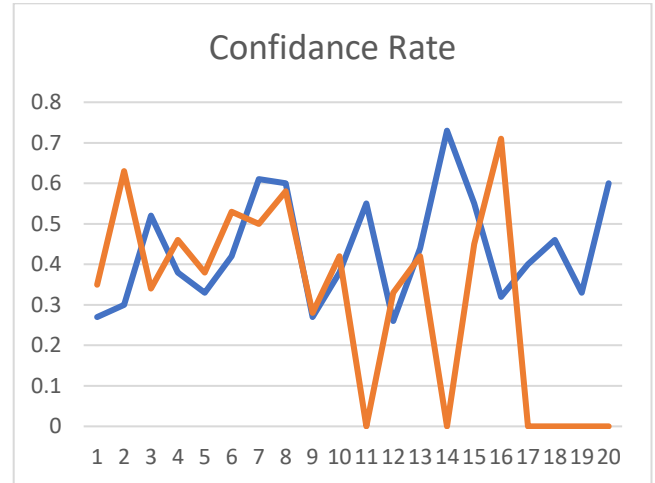


Figure 11 Confidence Rate

The threshold of 0.25 was chosen to make the model more sensitive to signs of fire. And the average data that have gotten by research by :

$$\text{Mean}(\%) = \frac{\text{Total data day} + \text{Total data night}}{\text{Data Obtained}}$$

$$\text{Mean}(\%) = \frac{9,156 + 6,38}{33} \times 100$$

$$\text{Mean}(\%) = 47,08\%$$

The average confidence value from the combination of daytime and nighttime testing reached 47%. This value indicates that the model's confidence level is in the moderate category using a confidence threshold of 0.25. The use of a low threshold aims to allow the system to detect small fire objects, which are critical to identify in the early stages of a fire. However, using a low threshold can also increase the likelihood of false positives, so the detection results still require validation from the YOLO model.

The results of the research should be written clearly and concisely. Discussions consider outlines the importance of research, not repeat it. Avoid excessive uses quotations and discussions about literature published.

4) Google Drive

The detected images are first saved to Google Drive after the Raspberry Pi detects an object using the color filtering method.

At this stage, the Raspberry Pi detects signs of fire based on color characteristics that match a predetermined threshold.

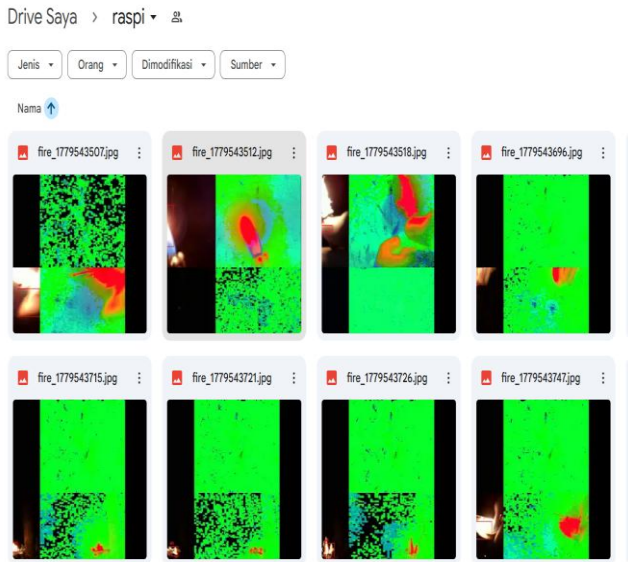


Figure 12 Google drive

If the YOLOv8 validation results indicate that the object belongs to the fire class, the image will be sent to Telegram as a notification to the user. With this workflow, the system has two processing stages: initial detection using color filtering on the Raspberry Pi and subsequent validation using YOLOv8 before the final result is sent to Telegram.

Thus, the output sent via Telegram consists of images of detection results from the Raspberry Pi that have undergone color filtering. In the initial stage, the Raspberry Pi detects indications of fire based on color characteristics that match a predetermined threshold. Once a fire indication is detected, the image is then processed and validated using the YOLOv8 algorithm to confirm whether the detected object truly belongs to the fire class or not.

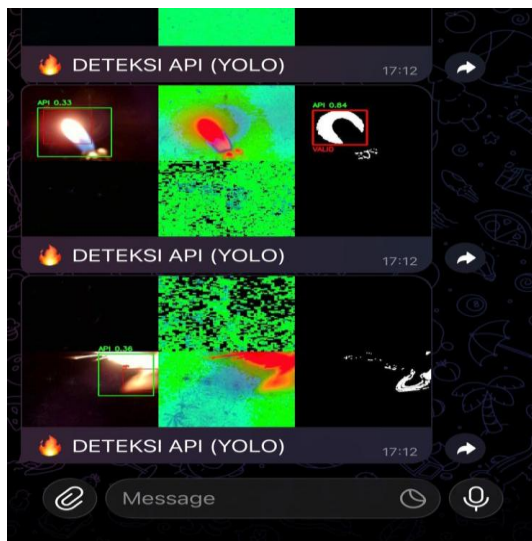


Figure 13 Telegram Output

This process gives the system two validation stages: initial validation using the color filtering method on the Raspberry Pi and advanced validation using YOLOv8 as the object detection algorithm. With these two validation stages, users can receive detection results via Telegram in the form of images, making the information clearer and usable as visual evidence of fire indications. Users benefit from two validation stages: an initial color-based validation by the Raspberry Pi and a subsequent object-detection-based validation by YOLOv8.

IV. CONCLUSION

Based on the test results, the early fire detection system performs quite well in terms of detection time, data transmission, and confidence scores. The detection process on the Raspberry Pi and validation using YOLO showed relatively fast response times, enabling the system to support early fire detection. Based on the overall experimental results, the proposed fire detection system was successfully implemented using Raspberry Pi 4 Model B, color filtering, and the YOLOv8 algorithm. The system achieved an average overall processing time of 6663.75 ms, with an average color filtering processing time of 65.89 ms and an average YOLOv8 processing time of 188.43 ms. In addition, the system obtained an average confidence score of 47.08% during the testing process. These results indicate that the proposed system is capable of performing automatic fire detection and processing image data within an acceptable processing time while maintaining real-time monitoring capability. Although the system has demonstrated its functionality in detecting fire and delivering detection results, further improvements are still required, particularly in dataset diversity, model optimization, and hardware performance, to improve detection accuracy, confidence, and overall system reliability in real-world environments. However, data transmission still faces challenges influenced by internet connection stability, as evidenced by the spike in transmission time for the second data set due to network disruptions. In terms of confidence, the average value obtained falls into the moderate category because the system uses a confidence threshold of 0.25. The use of a low threshold aims to ensure the system can still detect small fire indications early on, making it suitable for early fire detection needs. However, a low threshold also increases the likelihood of false positives, so validation using YOLO remains necessary. Overall, the system is already capable of performing its primary function as an early fire detection tool, namely detecting signs of fire, validating detection results, and transmitting detection information. However, the system still requires improvements, particularly regarding network stability, the diversity of the YOLO training dataset, and the reduction of false positives and false negatives to make detection results more accurate and reliable. Based on the results of this research, several improvements can be suggested for future development. First, the hardware used in the system should meet higher specifications than the minimum requirements. Using hardware with better processing capability can help the system run more stably, especially when processing detection algorithms such as YOLO. Hardware

selection should also consider power consumption, durability, and the ability to operate for a long period of time. Further development can consider using cloud server or hosting services for YOLO processing. In this research, Google Drive is mainly used to store and retrieve images, while the detection process still depends on the device and system configuration. By using a cloud server, YOLO processing can be performed on a more powerful server, which may affect the detection and notification time. However, this approach requires additional cost for hosting or server rental, so cost and performance need to be considered carefully. Third, the dataset should be improved by adding more varied images. The dataset should include fire and non-fire objects taken from different angles, distances, lighting conditions, backgrounds, and object positions. The number of images in the fire and non-fire classes should also be balanced so that the model can learn both classes more effectively. With a more varied and balanced dataset, the model is expected to reduce false positives and false negatives and improve overall detection accuracy.

REFERENCES

- [1] D. Arisandi, L. Trisnawati, and A. Syamsuadi, "Sistem Monitoring Deteksi Dini Kebakaran Hutan Berbasis Multiplatform Di Kabupaten Siak Menggunakan SDLC Prototyping," *Jurnal Sistem Komputer dan Informatika (JSON)*, vol. 3, no. 4, p. 410, Jun. 2022, doi: 10.30865/json.v3i4.4136.
- [2] A. B. Abdusalomov, B. M. S. Islam, R. Nasimov, M. Mukhiddinov, and T. K. Whangbo, "An Improved Forest Fire Detection Method Based on the Detectron2 Model and a Deep Learning Approach," *Sensors*, vol. 23, no. 3, Feb. 2023, doi: 10.3390/s23031512.
- [3] R. Han, Y. Liu, X. Kong, Z. Qiu, S. Li, and H. Liu, "Early forest flame and smoke detection based on improved feature extraction module with enhanced image processing inspired by YOLOV7," *Int. J. Wildland Fire*, vol. 33, no. 12, Dec. 2024, doi: 10.1071/WF24050.
- [4] A. Fitria Sari, "TELEGRAM SEBAGAI MEDIA INTERNAL (STUDI KEBUTUHAN INFORMASI KARYAWAN PT.TELKOM WITEL SULTENG)," 2018.
- [5] R. A. Satrio, A. W. Yulianto, and E. Mas'udia, "Design and Build Android-based LoRaWAN Mobile Drivetest Application for Network Planning in the Internet of Things (IoT) Network at PT. Telkom Indonesia," *Journal of Telecommunication Network*, vol. 15, no. 2, 2025.
- [6] ST. , MT. Rieke Adriati Wijayanti, ST. , MT. Nurul Hidayati, and ST. , MT. Ahmad Wilda Yulianto, "WORKSHOP IMAGE PROCESSING," 2022.
- [7] Yudi Pratistha, I Made Sukarsa, and Putu Wira Buana, "Implementasi Google Drive API untuk Upload, Sharing, dan Download Data pada Aplikasi Berbasis Web Pande Made Yudi Pratistha a1 , I Made Sukarsa a2 , Putu Wira Buana b3," Bali, Apr. 2022.
- [8] *ECTI-CON 2012 : 2012 9th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology : 16-18 May 2012, Phetchaburi, Thailand*. IEEE, 2012.
- [9] B. Yun, Y. Zheng, Z. Lin, and T. Li, "FFYOLO: A Lightweight Forest Fire Detection Model Based on YOLOv8," *Fire*, vol. 7, no. 3, Mar. 2024, doi: 10.3390/fire7030093.
- [10] S. P. Tauladani, R. A. Wijayanti, and R. Saptono, "Implementation YOLOv5 Method for Detecting Safety Equipment Completeness Images on Site Tower (Case Study: PT. Nexwave Surabaya)," *Journal of Telecommunication Network*, vol. 14, no. 2, 2024.
- [11] "What Is an API (Application Programming Interface)? | IBM." Accessed: May 14, 2026. [Online]. Available: https://www.ibm-com.translate.google/think/topics/api?_x_tr_sl=en&_x_tr_tl=id&_x_tr_hl=id&_x_tr_pto=tc
- [12] I. Delibasoglu, "Vehicle Detection from Aerial Images with Object and Motion Detection," *Turkish Journal of Mathematics and Computer Science*, vol. 14, no. 1, pp. 174–183, Jun. 2022, doi: 10.47000/tjmcs.1002767.
- [13] H. Pranamurti, A. Murti, and C. Setianingsih, "Fire Detection Use CCTV with Image Processing Based Raspberry Pi," in *Journal of Physics: Conference Series*, Institute of Physics Publishing, May 2019. doi: 10.1088/1742-6596/1201/1/012015.
- [14] P. Jiang, D. Ergu, F. Liu, Y. Cai, and B. Ma, "A Review of Yolo Algorithm Developments," in *Procedia Computer Science*, Elsevier B.V., 2021, pp. 1066–1073. doi: 10.1016/j.procs.2022.01.135.
- [15] R. Ardiansyah, B. Bastiar, A. Adzikirani, D. Marya, and A. Novianti, "Optimasi penerjemahan bahasa asing dengan teknologi IoT pada kelas internasional Politeknik Negeri Malang," *JURNAL ELTEK*, vol. 23, no. 1, pp. 1–8, Apr. 2025, doi: 10.33795/eltek.v23i1.6951.
- [16] M. Zuma, P. A. Owolawi, V. Malele, K. Odeyemi, G. Aiyetoro, and J. S. Ojo, "Intrusion Detection System using Raspberry Pi and Telegram Integration," *Association for Computing Machinery (ACM)*, Dec. 2021, pp. 1–7. doi: 10.1145/3487923.3487928.