International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 13 Issue: 02 | Feb 2026 www.irjet.net p-ISSN: 2395-0072

# VAN RAKSHAK Anti-Logging and Wild Fire Preventation System For De-Forestation

**Kedari V. K.[1], Talekar Siddhesh Popat[2], Shirsikar Aayush Mahendra[3], Mule Yash Nandkishor[4]**

[1]*Lecturer, Dept. of Information Technology, Jaihind Comprehensive Educational Institue's Jaihind Polytechnic Kuran, Maharashtra, India*

[2, 3,4]*Final year Diploma Student, Jaihind Comprehensive Educational Institue's Jaihind Polytechnic Kuran, Maharashtra, India*

---

**Abstract -** *The catastrophic effects of deforestation, which include climate change and other natural disasters, are a scourge on humanity. One of the main reasons for deforestation is human avarice, which leads to actions like clearing forests for agriculture and other uses by cutting down trees and allowing them to burn. A great deal of the planet's vegetation has been cut down as a consequence of these terrible activities, which threatens the survival of many species of plants and animals. In order to prevent loggers from cutting down trees, governments and other organizations take many steps, including monitoring loggers while they are in transit and arresting those responsible. Unfortunately, capturing loggers has shown to be an ineffective solution to the problem of deforestation. So, to prevent loggers and criminals from acting, we have implemented machine learning utilizing the Mobile Net neural network deep learning model. The program is designed to detect when loggers are active using a hidden camera in the forest. It then notifies forest officials and provides a location map so that the criminals can be caught in the act.*

***Key Words***: **Deep Learning, Mobile Net Neural Network, Location Map, Machine Learning, Deforestation.**

## 1. INTRODUCTION

With forest cover dwindling at an alarming rate, climate change intensifying, and human involvement with natural ecosystems on the rise, anti-logging and wildfire protection technologies are more crucial than ever. Deforestation creates a variety of problems, including soil erosion, increased carbon emissions, loss of biodiversity, and habitat disruption due to illegal logging and uncontrolled forest fires. The climate, ecological balance, and local inhabitants' lives are all greatly impacted by forests. Manual patrols and delayed reporting, the backbone of traditional forest monitoring approaches, frequently fall short in expansive, isolated, or densely forested regions. Authorities can react swiftly and limit damage with the help of an intelligent anti-logging and wildfire prevention technology that detects illicit activity and fire outbreaks early on. For the sake of environmental preservation, long-term climate and human life protection, and sustainable forest management, such systems are crucial.

The system employs a mix of machine learning, image processing, sensor networks, and the internet of things to detect illicit logging and possible wildfire dangers. Environmental sensors installed in woodlands record information including relative humidity, temperature, smoke density, noise patterns, and vibration signals produced by chainsaws and other heavy machines. Drones and camera systems record footage and stills of forest areas in real time. In order to standardize sensor values and eliminate noise, data pre-processing techniques are utilized. Important signs like a rapid increase in temperature, a high concentration of smoke, unusually high or low frequencies of sound, or the presence of unauthorized people can be located using feature extraction techniques. Normal forest conditions are distinguished from suspicious activities using classification and anomaly detection techniques. Early detection of anomalous patterns allows for prompt action by forest officials to prevent illicit logging and control wildfires.

Systems designed to prevent wildfires and anti-logging operations employ linear regression to analyze trends and make predictions about continuous environmental factors. Fire risk, fire spread rate, and forest degradation trends are some of the dependent variables that are modelled in this model. Independent variables include weather conditions (such as temperature, humidity, and wind speed) and seasonal factors. Linear regression is useful for predicting when forests are most at danger and where they are most susceptible by examining environmental and incident data from the past. Identifying patterns of deforestation over time, calculating the likelihood of fires, and bolstering prevention efforts are all areas where it shines. Environmental authorities are able to better allocate resources and devise proactive forest conservation programs with the use of linear regression, which offers obvious insights due to its interpretability and simplicity.

With Mobile Net, even devices with limited resources may perform efficient and accurate image-based detection, making it an essential component of this system. If you're looking for a lightweight convolutional neural network that can run in real-time on edge devices like cameras, drones, and embedded systems in forests, go no farther than Mobile Net. To automatically categorize scenarios as safe or dangerous, Mobile Net is trained on

pictures of forest landscapes, patterns of smoke from fires, flames, logging equipment, and human interference. Its depth wise separable convolutions optimize accuracy without sacrificing computational expense. By eliminating the need for cloud processing and minimizing latency, Mobile Net enables the edge-based, real-time identification of wildfire warning indicators and illicit logging operations. Modern anti-logging and wildfire prevention systems can benefit greatly from Mobile Net's speed, low power consumption, and resilience, making it an ideal tool for large-scale deployment in remote forest situations.

1] An accurate event detection technique for forest fire prevention was described by Vishal K. Singh et al., and it is based on fuzzy rules. To accurately infer forest fires using fire indices, the proposed approach was evaluated for classification and intensity discrimination. By comparing the inference accuracy of four variables—temperature, humidity, wind speed, and smoke—the suggested fuzzy logic-based approach classifier is able to infer the forest fire with an accuracy of approximately 98.7 percent. Forest fires can be accurately predicted and prevented using various combinations of smoke, wind speed, humidity, and temperature, according to the study and results.

2] For the purpose of detecting forest fires, Burakkizilkaya et al. developed a framework that is both effective and energy-efficient. In contrast to previous research, this method combines scalar sensors with multimedia sensors that use deep learning models; detection is then carried out by fusing data from several levels of the system's architecture. The use of edge computing in decision-making also improves communication efficiency. The suggested architecture makes advantage of both the precision and efficiency of scalar and multimedia sensors. The novel framework achieves a sustainable emergency surveillance system balance between energy efficiency and detection accuracy through the development of a hierarchical structured structure. author show and analyze the findings of the testbed experiments and the simulations in a comparative manner. The trial results show that energy efficiency improves by 29.94% (for up to 28.50 days). Furthermore, processing can be enabled on edge devices with the help of a new lightweight CNN model. The experimental results show that a 98.28% success rate is reached. Considering all of the current fire detection studies, author's method outperforms all of them in terms of accuracy, with the exception of the one in Reference [36]. author's method's performance is comparable to that of a 22-layer deep convolutional neural network, even as Reference [36] utilizes Google's own network. In this work, the limits of edge computing are considered alongside the existing lightweight architectures in the tests conducted for comparison, as the existing methodologies fail to do so. When it comes to accuracy, the suggested method is on par with the top lightweight architectures.

3] Haolin Yang et al. report that wildfires have become a major threat in recent decades. Wildfires can be mitigated by early detection and monitoring. AI, model refinement, and training. Random Forest appears to outperform DenseNet and other deep learning models. This is likely due to inadequate deep learning model training and tuning. This research revealed that the recently developed method of mathematically constructing burn maps with multiple indices and refining them with an object identification model may produce accurate burn maps. It can't yet produce accurate burn maps even with poor data quality due to interference and noise. Train the object detection model and fine-tune the mathematical procedure to improve the method's accuracy, especially on low-quality photographs. Wildfire forecasting needs additional research. After identifying a wildfire's perimeter, its proximity to the edge and flammability can be used to determine its spread. The Normalized Difference Vegetation Index (NDVI) is already used to determine perimeter, but it could also assess burning susceptibility. Future research could estimate burn severity accurately and reliably. As indicated, background noise and other distractions might affect dNBR maps, making them unreliable for burn severity assessment.

The second part of this paper examines the prior research that was held in high regard as Litereature Survey. Section 3 provides a comprehensive description of the proposed methodology, outlining the path of action. To sum up the article, a conclusion is provided on the current plan. Section 5 delves into prospective improvements, while Part 4 examines the experimental evaluation.

## 2. LITERATURE SURVEY

4] Lertsinsrubtavee Adisorn et al. In this article, author will discuss author's personal experience with building and launching a network to monitor field haze and identify forest fires using a low-cost Internet of Things (IoT) sensor. A number of air pollution parameters can be retrieved from distant sensors using the SEA-HAZEMON platform. author's Canarin sensor nodes are unique among platforms in that they are able to sustain themselves and continue operating actively in forest areas for more than a year. For the purpose of identifying forest fires as they occurred in real time, a basic model was also utilized, which relies on PM2.5 and CO concentrations. Using satellite imagery and data collected by the forest fire control agency during the height of the forest fire season, author tested author's system's ability to accurately detect fires. author were able to get above 80% accuracy with author's model. Nevertheless, due to the low precision and recall levels, author discovered a few false positive events

5] The catastrophic effects of wildfires, as suggested by YIGIT TUNCEL et al., might linger for years after the event has passed. It is essential to take proactive measures to detect and prevent wildfires in order to lessen their impact. Continuous monitoring and improved detection accuracy can

be achieved using sensor suites set near the gridlines. These sensor suites need to be able to run independently. This research introduces a new self-sustaining CPS that optimally controls sensor sample rates to maximize both device longevity and the accuracy of wildfire detection. Using an open-source wildfire simulator and real-life datasets, the suggested method is thoroughly tested using reinforcement learning. A meticulously calibrated heuristic approach is at least 2.5 times slower than the suggested framework, which achieves an uptime of 89% and a reaction time of 2 minutes, according to the simulation findings. Additionally, when contrasted with the daily energy consumption of the sensor suite, the trained policy network consumes a minuscule 0.6 mJ of energy. Consequently, the suggested method permits accurate, ongoing, and detailed wildfire surveillance. Deploying it in outlying locations could greatly improve the efficiency of wildfire detection and prevention efforts.

6] According to Warit Phankrawee et al., author blended the fire data from NASA FIRMS with 'Temperature, Humidity, Rainfall, and Pressure' data by matching latitude and longitude using KDTree, a convenient method for fast nearest-neighbor identification, during data preprocessing. The KDTree is created using the 'lat,' 'long,' and 'datetime' columns, which link nearby meteorological stations to fire sites. Machine learning was constructed using the Extratrees BAG L2 model for prediction. This model incorporates ExtraTrees, a dynamic variation of the random forest technique. It excels at making and merging different decision trees. This contributes to more precise forecasts. In order to better anticipate forest fires in the Thailand region, author's research shows that machine learning models work well. The results provide useful information for controlling and preventing fires.

7] Because it is built entirely in Python, the innovative online deforestation monitoring system suggested by Zhipan Wang et al. can drastically cut down on development costs. Anyone, even those without technical training, can utilize their own high-resolution pictures to track deforestation using this technique. In addition, additional areas of remote sensing application domains can find new ideas in the architecture of this open-sourced system, and professionals can easily create similar online systems by editing the source code. In addition to assisting with UN Sustainable Development Goal 13: Climate Action, author think Open Forest Monitor can make a big splash in the deforestation detection research community.

8] Ioannis Ioannidis etc. Forest fires are becoming an increasingly serious problem, therefore we need new ways to keep an eye out for them, identify them, and react quickly. Using a microservices architecture and container-based virtualization, we have presented a Smart Forest Fire Monitoring and Detection System throughout this work. The system provides an efficient, scalable, and modular framework that can operate in contexts with limited resources by utilizing these modern technologies.

9] A FFSRP model was introduced by Xuan Sun et al. following an unintentional forest fire. To begin, the combustible state, meteorological factors, and topographical conditions are the readily available forest fire affecting variables that have been identified based on pertinent research and real-world situations. The FFSPP model is subsequently created by merging the CA and Wang Zhengfei models. Also, using ML techniques, the FFSRP model can estimate the burned area. At last, to prove the model and approach, we use the Chinese TV show "3.29 Forest Fire" and a real-life fire dataset from Portugal's Montesinho National Forest Park. Compared to the fire behavior simulation models developed by Farsite and Prometheus, the suggested FFSPP model has a lower relative error. In addition, the suggested FFSRP model is capable of producing accurate predictions in situations involving small to medium-sized fires.

10] Using 2020 and 2021 Sentinel-2 satellite imagery and ground-truth data, Mariam Alshehri et al. identified deforested areas in the Brazilian Amazon jungle. The author trained a transformer-based network for DD using these data and developed a bitemporal deforestation dataset. The author performed an extensive search for hyperparameters, trying out several configurations until they found the ones that worked best for their goal. According to the author's research, a color-shifted infrared composite and cross-entropy loss with AdamW optimizer produced the best overall performance in terms of accuracy and balance in detecting changes in class, with an accuracy of 84.70% and a recall of 84.53%, respectively. By comparing their results to those of previous studies in DD, the authors conclude that transformer-based networks can achieve far higher accuracy.

11] A multimodal framework for forest monitoring was introduced by Maha Sliti et al., which combines fiber Bragg grating (FBG) sensor networks with unmanned aerial vehicle (UAV)-mounted diffraction spectroscopy. The design provides a supplementary view of tree physiology not possible with either ground-based or remote sensing alone by integrating canopy-level reflectance with in-stem strain and temperature proxies. Georeferenced stress maps are created by combining UAV and FBG data, which allows for earlier and more accurate identification and targeted management activities such precision watering, insect treatment, and selective thinning before symptoms show up. From an operational standpoint, UAV platforms offer fast, as-needed coverage across varied terrain, while embedded FBG arrays guarantee continuous, passive, multiplexed sensing in the tree canopy. The pipeline meets all of the demanding standards for accuracy and reliability, as shown by the simulation results run under both daily and weather-related variations.

12] Sugi Choi et al. recommended utilizing the Swin Transformer as the Mask-RCNN model's backbone network to identify wildfires better. This model solved the problem of traditional wildfire detection systems failing to discriminate fire from non-fire events by integrating clouds, mist, and chimney smoke. The Swin Transformer detected fire and non-fire situations better, the data showed. Based on the confusion matrix, the Swin Transformer decreased false positives for non-fire categories, with 95% cloud classification accuracy and 97% mist classification accuracy. The model outperformed ResNet-50 with 85% accuracy for chimney smoke, which is similar to wildfire smoke. Despite low sight, the Swin Transformer detected early-stage fires with 79% smoke detection and 64% flame detection. Quantitative analysis employing mean Average Precision (mAP) criteria confirmed the Swin Transformer's durability. The model surpassed ResNet-50 in bounding box identification and segmentation mAP@50 scores of 0.849 and 0.842. The model also performed better with objects of different sizes, with bounding box detection mAP_S = 0.166, M = 0.494, and L = 0.699. The Swin Transformer can detect fire indications at varied scales, from microscopic smoke trails and flames that help detect wildfires early on to larger smoke plumes that indicate a more advanced fire.

13] Following a battery of machine learning tests, the optimal model for forest/non-forest classification was determined by Giacomo Albamonte et al. We trained, assessed, and tested both classic ML methods and DL ones. While all of them reached an Intersection over Union (IoU) of 89.36%, the FPN model that took vegetation indices into account performed the best. For the purpose of methodically organizing entities and interactions across several domains, a custom ontology called SORSOntology was created to expand the capabilities of the framework. Created by merging and expanding upon preexisting ontologies, SORSOntology facilitates easy retrieval of information with a semantic framework from a central repository of knowledge. A task-based evaluation was used to evaluate the ontology.

14] The group headed by Shakti Kundu. Deforestation indications such as tree stumps, trunks, machinery, and human presence can be efficiently identified by combining the autonomous decision-making capability of LangChain Agent with the robust object recognition framework of the YOLOv8 model. The algorithm's great backdrop detection performance demonstrates its capability to effectively manage extensive forested terrain. But the model needs further work to give consistent identification in varied environments because it fails miserably at recognizing smaller, less distinctive objects. Throughout many epochs, the model's training results showed promising tendencies toward reducing significant loss measures as train/box_loss, train/cls_loss, and train/dfl_loss decreased consistently. These findings demonstrate that learning and development are possible throughout training under the model's guidance. Nonetheless, the model needs better generalizing approaches, and the increasing trend in validation loss measures suggests overfitting is likely. To overcome these obstacles and improve the model's performance in the actual world, data augmentation techniques, more diversified datasets, and improved feature extraction are employed.

15] Using official CONAF ground-reported information, Cristian Vidal-Silva et al. present a new, comprehensive wildfire dataset for Chile spanning 1985–2024. The dataset also allows for analysis that are unique to climate zones. It is possible, for instance, to compare wildfire patterns in Chilean towns located within the Mediterranean zone to those in other parts of the world with similar climates. This provides a chance to evaluate policies on a global scale and conduct comparative modeling across landscapes that are prone to fires. The dataset is thoroughly cleaned, validated, and standardised to ensure it is full and reliable; this is crucial for research that involve complex fire modeling and risk assessment. The following are some important takeaways from this study: • The dataset greatly enhances the resources for investigating the frequency, size, human impact, and economic losses caused by wildfires in Chile, a country that is more at risk from these disasters as a result of changes in climate and land use. • Fire prediction, risk mapping, and policy evaluation are among of the many machine learning, deep learning, and statistical modeling tasks that can directly benefit from the use of standardized variables and temporal aggregation [18]. • The dataset is useful for international comparison studies and adds to worldwide efforts to understand and reduce wildfire risks, therefore its scientific importance goes beyond national borders.

## 2. PROPOSED METHODOLOGY

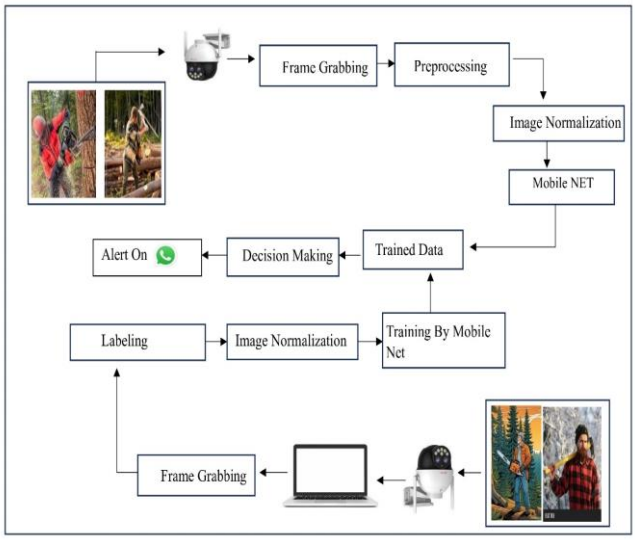


**Figure 1: System overview**

What each phase of the system diagram represented for VAN RAKSHAK: Following these steps will describe an anti-logging and wildfire prevention approach for deforestation .

**Step 1: Dataset Generator**

Creating a dataset is to take images of the forest's many encroaching logging activities using OpenCV. The Video Capture method in the CV2 package takes pictures of the target wild animals so that they can be spotted when testing the model. The dataset folder contains all of the logging equipment. As the model evolves, it incorporates animal models like the chainsaw. In preparation for the next model step, a folder is made to store the collected logging images.

**Step 2: Data Labelling**

The second step is data labeling, which entails annotating the previously captured photos using the labeling program. Once the user imports the image into the labeling software, they can specify the top right corner and bottom left corner coordinates to generate the x1, y1, and x2, y2 of the bottom right corner of the rectangle, respectively. The collected coordinates are saved in an.xml file so that a deep learning network named MobileNet can train the model.

**Step 3: API Installation**

This Google Colab instance requires the TensorFlow Object Detection API to be installed beforehand. All you need is a copy of the [TensorFlow models repository] (https://github.com/tensorflow/models) and a few installation scripts. Pressing the play button will execute the following sections of code. The next step is to set up the conda environment so that we can download and extract cuDNN files. Then, grab the tensorflow models project from GitHub and clone it. The next step was to install the Object Detection API.

**Step 4: Prepare Training Data and Upload Image Dataset**

Uploading the training pictures and running the scripts for TF Record generation are necessary steps to prepare the TensorFlow training data. We need to upload our photos and sort them into a "train," "validation," and "test" folder before we can run the scripts that will make TF Records from our data. First things first, on our local PC, create a single folder called "images.zip" and zip all of our training images and XML files into it. All of the files need to be in the zip folder. We need to use a few commands to configure our picture directories and extract the contents after it's uploaded. The /content subdirectory of the file system is where these folders are created in this specific case. The "Files" icon on the left allows us to navigate the file system.

**Step 5: Divide the picture folders into test, validation, and train.**

You can see the folder icon on the left side of the screen, and among the listed files, you can find our "images.zip" file. The following step, following dataset upload, is to extract its contents and arrange them into picture folders. The /content subdirectory of the file system is where these folders are created in this specific case. The "Files" icon on the left allows us to navigate the file system.

Making a trifecta of the images (train, validation, and test) is the next stage. The following is the function of every set: To train the model, these exact photographs were used. A fresh collection of images from the "train" set is input into the neural network at each training stage. After the network classifies the objects in the photos, it can predict where those objects are. The training algorithm calculates the loss and then adjusts the network weights through back propagation. Training progress can be examined by modifying hyperparameters (like learning rate) and images from the "validation" collection can be used by the training algorithm for validation. During training, these images are used less frequently than the "train" images, which are used in every phase. The neural network is trained independently of these photographs as a test. A human should use these to check the model's accuracy in the end.

**Step 6: Make TF Files**

Finally, we need to convert the images to TF Records, a data file format that TensorFlow uses for training. Python applications are used to automatically convert the data into TF Record format. A class label map has to be set up before we can execute them. By running the code section below, you can create a "labelmap.txt" file that has a list of classes. Create a new line for each of our classes and substitute the words 'class1', 'class2', and 'class3' with them. Next, launch the code by pressing the play button. In doing so, the object detection model's detectable classes are stored in a "labelmap.txt" file.

**Step 7: Set Up Training Configuration.**

In this section, we specify which TensorFlow 1 Object Detection Model we want to use initially. All models come with a configuration file. Training parameters, such as learning rate and total number of steps, and file locations can be set in this file, among other things. Changes are being made to the configuration file of our custom training job at the moment.

In the first portion of the code, you can see a list of models that are available in the TF1 Model Zoo. Then, you'll see the names of the files that will be used to download the model and their configuration. Keeping tabs on the models we're using and adding more as required becomes a breeze with this.In the "chosen_model" field, provide the name of the model that we wish to train with. At this point, the "ssd-mobilenet-v1-quantized" model is chosen for implementation. To specify and download the configuration file and pre-trained model file, proceed to the next three stages and click play.

Once we have downloaded the model and configuration file, we need to add some general training

parameters to the file. Using the following variables, we can control the different training stages:

The total number of steps to utilize when training the model is num_steps. For an initial target, 40,000 steps is a solid choice. If we observe that the loss measures are still falling when training is complete, we can apply additional steps. It takes more time to train if there are more steps. If the loss level reaches the specified point before the training period ends, training can be terminated early. **batch size**: The number of photos to utilize for each training stage. The amount of memory that can be used for training on a GPU determines the maximum batch size, which in turn reduces the number of steps needed to train a model. In most cases, 16 GPUs is an adequate amount for a Colab instance.

Quantization-aware training is the only one that uses quant_delay_steps. Following these extensive procedures, the training algorithm will simulate quantization by inserting "fake" quantization nodes into the network. Half of the total training steps is a decent beginning point. [This article](https://neuralet.com/article/quantization-of-tensorflow-object-detection-api-models/) has more details. In this step, we also assign other pieces of training information, such as the total number of classes, the location of the config file, and the location of the pre-trained model file. Changing the configuration file is necessary to implement the newly defined training parameters. As its own "pipeline_file," this code will use the.config file that you obtained."config" with all the necessary parameters pre-populated.In the main pipeline file, you may create a custom configuration by include our dataset, model checkpoint, and training parameters. The following block deals with updating the training script to save checkpoints every 1000 steps. Just tweak 'num_eval_steps' to make it save checkpoints more or less frequently. The design of the mobile network is shown in Figure 2.

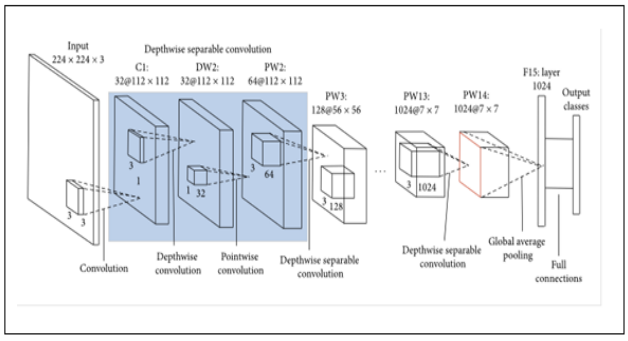


**Figure 2: Mobile net Architecture**

**Step 8: Train Custom TF1 Object Detector**

We will train our object detection model using the Custom TF1 Object Detector. This model is trained using the "model_main_tf2.py" script that is part of the TF Object Detection API. We have defined all the arguments and parameters used by'model_main_tf2.py' in earlier portions of this Collab. A training duration of 2–6 hours is possible, depending on the model, batch size, and number of steps in the training process. This will be useful later on. Using the tflite file to test the model extensively.

**Step 9: Testing the model for Tree Logging**

We put the model through its paces for tree logging: Here, the Python code makes use of the camera to capture video and, by implication, the frames using the cross-platform Droid Cam app. Employing the learned model file. Flite, the logging equipment can be detected within the live stream frames. When we detect tree logging activities, we immediately inform the relevant authorities. They are then contacted by WhatsApp and provided with a map showing the location, direction, and landmark of the camera.

## 3. RESULTS AND DISUSSIONS

The Windows-based system, which has 16 GB of main memory and an Intel Core i7 processor, is used to implement the proposed model. The Anaconda IDE repository for Spyder and Jupyter IDEs is used by the model for the experiment. The created model is rigorously evaluated using the parameters of the confusion matrix. You may understand the parameters of the confusion matrix using the following equations: accuracy, precision, recall, and macro F1.

$$Accuracy=(TP+TN)/(TP+FP+TN+FN) \quad -(1)$$
$$Precision(P)=TP/(TP+FN) \quad -(2)$$
$$Recall(R)=TP/(TP+FP) \quad -(3)$$
$$Macro\text{-}F1=(2*P*R)/(P+R) \quad -(4)$$

At this point, we have TP for true positives, TN for true negatives, FP for false positives, and FN for false negatives.

The obtained results are shown below,

1.Confusion Matrix
2.F1-Confidence Curve
3. Precision Confidence Curve
4. Precision- recall Curve

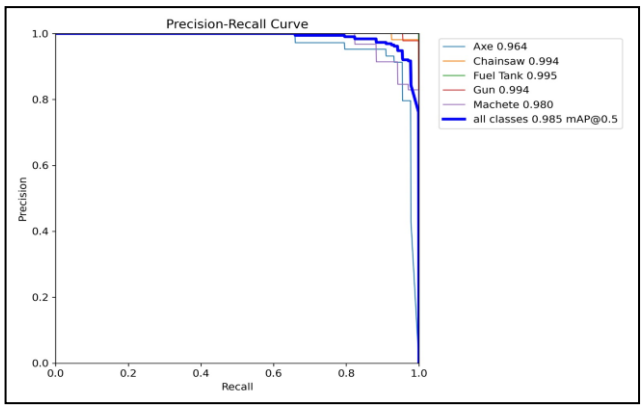


**Figure 3: Precision–Recall (PR) Curve**

The Precision–Recall curve illustrates the detection performance of each class. All classes achieve high precision and recall values, indicating accurate classification with minimal false positives and false negatives. The overall mAP@0.5 score is 0.985, showing excellent model performance. Fuel Tank and Chainsaw classes achieve near-perfect precision values. The curve remaining close to the top-right corner indicates strong detection reliability. This graph confirms the effectiveness of the trained YOLO model.

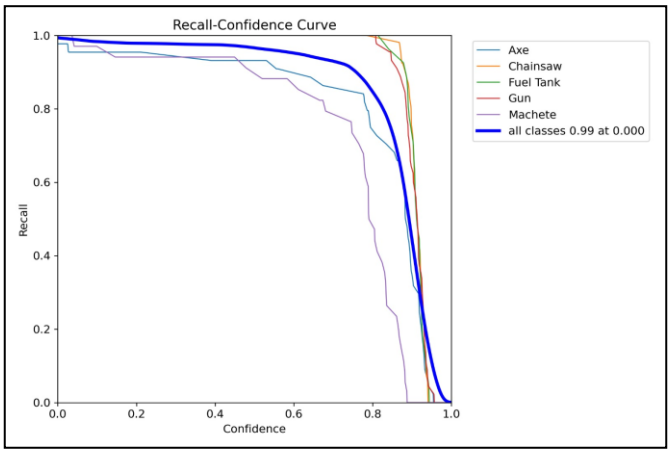


**Figure 4: Recall–Confidence Curve**

The Recall–Confidence curve shows how recall changes with varying confidence thresholds. At lower confidence levels, recall remains close to 1.0 for all classes. As confidence increases, recall gradually decreases, especially for the Machete class. The overall recall across all classes is approximately 0.99 at low confidence. This graph helps in selecting an optimal confidence threshold for deployment. It demonstrates the trade-off between detection sensitivity and confidence level.

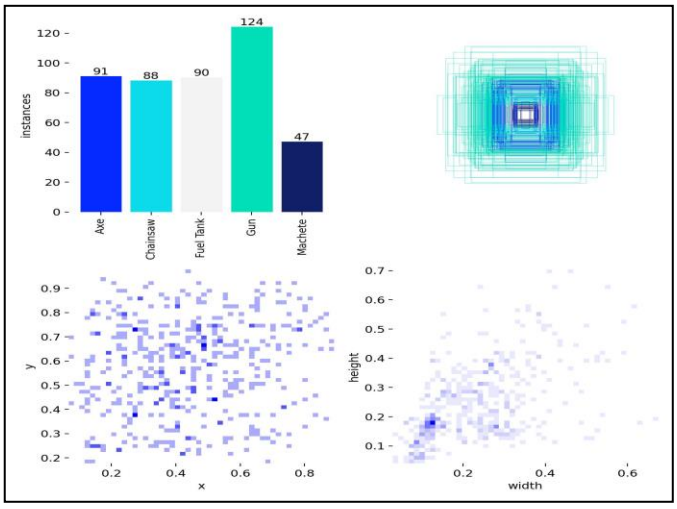


**Figure 4: Dataset Distribution and Bounding Box Analysis**

This graph presents the class-wise distribution and spatial characteristics of the dataset. The bar chart shows that the Gun class has the highest number of instances (124), while Machete has the lowest (47). The scatter plots represent the distribution of object center positions (x, y) and bounding box width–height variations. Most objects are concentrated near the center of the images. The width–height plot indicates variation in object sizes, useful for model generalization. This analysis ensures dataset balance and diversity.


**Figure 5: Dataset Sample Detection Visualization**

This figure shows sample output images from the trained object detection model. Bounding boxes are drawn around detected objects such as Axe, Gun, Chainsaw, Fuel Tank, and Machete along with their confidence scores. The model successfully identifies multiple objects in different positions and orientations. High confidence values (around 0.8–0.9) indicate strong detection performance. This visualization confirms that the trained model can accurately localize and classify weapons in real-world scenarios. It also demonstrates robustness across different frames.

## 5. CONCLUSIONS

An intelligent anti-logging and wildfire protection system can stop deforestation, according to this paper. The growth in illegal logging and forest fires caused by human carelessness and climate change has made the previous techniques of monitoring things ineffective. The proposed system monitors forest regions in real time using cutting-edge technology to detect and respond to threats. By reducing manual surveillance and enhancing situational awareness, the technology helps preserve forests, wildlife, and the environment.

Data analysis and machine learning improve system efficiency and reliability. Linear regression models environmental variables including temperature, humidity,

and seasonal fluctuations to help authorities identify harmful regions and times. Mobile Net image-based monitoring lets edge devices with minimal capabilities identify wildfire indicators and illegal logging in real time. Overall, the recommended technique to control logging and forest fires appears promising for large, susceptible forests. The system facilitates early intervention, environmental impact reduction, and data-driven decision-making to fight deforestation in a scalable way. The technology could improve forest preservation and sustainable environmental management for decades with broad sensor integration, satellite data fusion, and intelligent response coordination. Future plans include a massive anti-logging and wildfire system. An enhancement of the system with real-time weather forecasting, drone surveillance, and satellite imaging can improve early alerts over large forest regions. Advanced artificial intelligence models and ensemble learning can detect complex criminal activities and fire spread patterns in different environments.

Future advances include predictive models for long-term deforestation evaluation, automatic alarm systems connected to emergency services, and edge-cloud collaboration for faster response. Integration with GIS platforms and forest official mobile apps improves visualization and decision-making. Scalable deployment, sensors powered by renewable energy, and policy-driven analytics make the system a promising solution to sustainable forest protection and environmental conservation.

## REFERENCES

1] V. K. Singh, C. Singh and H. Raza, "Event Classification and Intensity Discrimination for Forest Fire Inference With IoT," in IEEE Sensors Journal, vol. 22, no. 9, pp. 8869-8880, 1 May1, 2022, doi: 10.1109/JSEN.2022.3163155.

2] B. Kizilkaya, E. Ever, H. Y. Yatbaz, and A. Yazici, "An Effective Forest Fire Detection Framework Using Heterogeneous Wireless Multimedia Sensor Networks," ACM Trans. Multimedia Comput. Commun. Appl., vol. 18, no. 2, art. 47, May 2022. [Online]. Available: https://doi.org/10.1145/3473037.

3] H. Yang, "Wildfire Detection and Perimeter Mapping using Satellite Imagery and Machine Learning with Hyperopt Tuning," in CSSE 22: Proceedings of the 5th International Conference on Computer Science and Software Engineering, Guilin, China, Oct. 2022, pp. 497–505. doi: 10.1145/3569966.3570097.

4] A. Lertsinsrubtavee, K. G. S. Jayarathna, P. Mekbungwan, T. Kanabkaew, and S. Raksakietisak, "SEA-HAZEMON: Active Haze Monitoring and Forest Fire Detection Platform," in AINTEC '22: Proceedings of the 17th Asian Internet Engineering Conference, Hiroshima, Japan, Dec. 2022, pp. 88–95. doi: 10.1145/3570748.3570761.

5] Y. Tuncel, T. Basaklar, D. Carpenter-Graffy, and U. Ogras, "A Self-Sustained CPS Design for Reliable Wildfire Monitoring," ACM Trans. Embedd. Comput. Syst., vol. 22, no. 5s, art. 135, pp. 1–23, Oct. 2023. [Online]. Available: https://doi.org/10.1145/3608100.

6] W. Phankrawee, N. Pornpholkullapat, T. Savanpopan, and S. Usanavasin, "Wildland and Forest Fire Prediction in Thailand using Satellite Data," in ICISE 23: Proceedings of the 2023 8th International Conference on Information Systems Engineering, Bangkok, Thailand, Dec. 2023, pp. 159–163. doi: 10.1145/3641032.3641062.

7] Z. Wang et al., "A Web-Based Prototype System for Deforestation Detection on High-Resolution Remote Sensing Imagery With Deep Learning," IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, vol. 17, pp. 18593-18606, 2024. doi: 10.1109/JSTARS.2024.3463360.

8] I. Ioannidis, A. Anagnostopoulos, and B. Mamalis, "Smart Forest Fire Monitoring and Detection System using Microservices and Container-based Virtualization," in PCI '24: Proceedings of the 28th Pan-Hellenic Conference on Progress in Computing and Informatics, Egaleo, Greece, Dec. 2024, pp. 368–375. doi: 10.1145/3716554.3716610.

9] X. Sun et al., "A Forest Fire Prediction Model Based on Cellular Automata and Machine Learning," in IEEE Access, vol. 12, pp. 55389-55403, 2024, doi: 10.1109/ACCESS.2024.3389035.

10] M. Alshehri, A. Ouadou, and G. J. Scott, "Deep Transformer-Based Network Deforestation Detection in the Brazilian Amazon Using Sentinel-2 Imagery," IEEE Geoscience and Remote Sensing Letters, vol. 21, pp. 1-5, 2024, Art no. 2502705. doi: 10.1109/LGRS.2024.3364952.

11] M. Sliti, A. Elfikky, A. I. Boghdady, S. A. H. Mohsan and S. Ayouni, "Integrating UAV-Mounted Diffraction Spectroscopy and FBG Sensor Networks for Real-Time Forest Health Monitoring," in IEEE Access, vol. 13, pp. 196195-196205, 2025, doi: 10.1109/ACCESS.2025.3628804.

12] S. Choi, Y. Song and H. Jung, "Study on Improving Detection Performance of Wildfire and Non-Fire Events Early Using Swin Transformer," in IEEE Access, vol. 13, pp. 46824-46837, 2025, doi: 10.1109/ACCESS.2025.3528983.

13] G. Albamonte, G. Falcone, M. Monaco, and S. Senatore, "Constructing a knowledge base from remote sensing indicators for deforestation assessment," Applied Intelligence, vol. 55, art. no. 1014, Oct. 2025. doi: 10.1007/s10489-025-06896-2.

14] S. Kundu et al., "Real-time deforestation anomaly detection using YOLO and LangChain agents for sustainable environmental monitoring," Scientific Reports, vol. 15, art. no. 39961, 2025. doi: 10.1038/s41598-025-23617-4.

15] C. Vidal-Silva et al., "Wildfire Occurrence and Damage Dataset for Chile (1985–2024): A Real Data Resource for Early Detection and Prevention Systems," Data, vol. 10, no. 1, art. no. 14, Jan. 2025. doi: 10.3390/data10010014.