University of Trento

Department of Industrial Engineering

Real-Time Detection of Grapevine Leaf Diseases with Edge AI and IoT Technologies

Pietro Riccardo Ferrario
241237

Individual assignment for the course

IoT

February 3, 2025

***Abstract*—Grapevine diseases and pests pose significant challenges to agricultural productivity, particularly in organic farming, which relies heavily on early detection. Traditional methods of disease detection depends on manual inspection, which is labor-intensive and prone to errors. This project leverages advancements in edge AI and IoT technologies to develop a real-time system for the classification of the health status of grapevine leaf. The Nicla Vision was utilized as edge device to perform on-device inference using a quantized TensorFlow Lite model, achieving an accuracy of 93%. Blob detection was implemented to pre-process input images, ensuring efficient region identification and classification. The system transmits classification results, which include labels, probabilities and respective images, via an *MQTT* broker to a centralized data management pipeline, comprising *Node-RED*, *InfluxDB* and *Grafana*. This framework enables efficient data storage and visualization, making it highly adaptable for vineyard monitoring.**

Index Terms— Edge Computing, Nicla Vision, Grape Disease Detection, TensorFlow Lite, Blob Detection, IoT, Node-RED, InfluxDB, Grafana.

I. Introduction

The cultivation of grapes faces numerous challenges, primarily the presence of diseases and pests, which can significantly reduce field yield and negatively impact the overall agricultural productivity of a region, [1]. In recent years, there has been a growing shift toward organic farming practices. Many agricultural enterprises are opting to use natural fertilizers, pesticides, fungicides, and herbicides, both for ethical considerations and to meet consumer demands for sustainably produced goods, [2], [3]. These methods, however, are often less effective than synthetic alternatives, leaving vineyards more vulnerable to diseases and pests [4]. Given these circumstances, the timely and accurate identification of issues affecting plants has become increasingly critical. Traditional methods for disease detection typically rely on manual inspection, which is labor-intensive, costly and prone to human error, [5]. These limitations highlight the growing necessity for automated and efficient solutions to address the challenges in the industry.

Recent advancements in edge artificial intelligence and embedded systems have made it possible to perform complex tasks, such as identifying the health status of plant leaves, on resource-constrained devices, [6]. This project leverages the Nicla Vision, a compact and versatile platform, to perform real-time detection and classification of grape leaves into two classes, healthy and sick. The Nicla Vision's capability to process data locally, makes it an ideal choice for testing machine learning models. To build and optimize the machine learning model adopted in the works, Edge Impulse was employed, a platform specifically designed for *TinyML*, aimed at simplify the development and deployment of *ML* models on edge devices, [7]. Transfer learning, a technique that re-purposes pre-trained deep neural network weights and fine-tunes them on domain-specific data, was adopted for this project due to its proven effectiveness in reducing training time and improving accuracy [8]. The custom dataset used for fine tuning included images of grape leaves, classified as either healthy or sick, ensuring that the model was tailored to the task of classification of health status of leaves. In addition to the neural network based classification, a computer vision techniques, *blob detection* was employed to pre-process images. This approach was selected for its efficiency and minimal resource requirements, enabling the system to accurately identify and isolate region of interest, such as individual leaves, from the input image, [9]. By pre-filtering the input, the neural network could focus exclusively on determining the health status of detected leaves without the added computational burden of identifying whether the object in question was a leaf or not. The image classification process occurs directly on the Nicla Vision, ensuring real-time inference without relying on external computational resources. The results, which include the classified image and corresponding health labels, are firsttransmitted via wireless to a personal laptop, and then stored in a database using *MQTT* protocol. *Node-RED* was selected to handle the data stream and store classified images in an *InfluxDB* database, organized into separate folders for healthy and sick leaves. Finally, the results are visualized in a user-friendly manner using *Grafana*, enabling the stakeholders to monitor the health status of leaves on real time and make informed decisions for vineyard management. Figure 1 visually represents the workflow of the system.

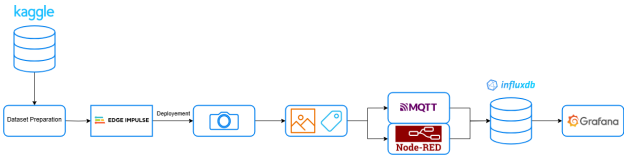


Fig. 1: Workflow Diagram

II. Methodology

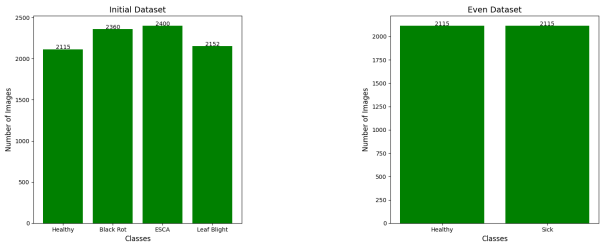
This section describes the systematic approach employed in this project, starting from the selection, acquisition and pre-processing of the dataset, creation of the impulse framework in order to apply transfer learning strategies, model retraining and optimization, to the deployment of the model on the target device and integration of the data handling process for transmission, storage and visualization.

A. Dataset

The dataset used in this project is the *Grapevine Disease Dataset (Original)* [10], which contains 9027 high-quality images depicting both healthy and diseased leaves. The dataset is divided into four categories: three

for specific diseases and one for healthy leaves. The diseases categories considered in this study are as follows:

1) Black Rot, caused by the fungus *Guignardia bidwellii*, thrives in hot and humid conditions and is a serious fungal disease for grape production, particularly present in the Middle East. This disease renders grapres unfit for consumption. [11], [12]
2) Esca is a trunk disease often attributed to the fungus *Phaeoacremonium aleophilum* and tipycally emerges during the summer months of July and August, [13], [14]. In France, Esca affects approximately 13% of vineyard annualy, resulting in cumulative financial losses exceeding 1 billion euros for the region, [15]
3) Leaf Blight, a bacterial infection cause by *Xylophilus ampelinus*, has been known to reduce yields by over 70% in infected vineyards. [16],



(a) Initial Dataset (b) Final Dataset

Fig. 2: Comparison of the initial and modified dataset distributions.

Since the objective of this application is to determine wether a leaf is healthy or diseased, the dataset was modified accordingly. To adapt it for binary classification, *Healthy vs Sick*, the number of healthy leaves was taken as the reference. As highlighted by [17], an imbalanced dataset, where the number of images varies significantly across labels, can adversely impact the performance of classifiers. To address this, an equal number of images were randomly selected from the three diseases categories to match the number of healthy leaves images. The modified dataset consists of 4230 images, evenly split between healthy (2115 images) and sick (2115 images). All images are 256x256 pixels and in RGB format. The initial dataset distribution and the modified dataset are depicted in the figure below 2. Additionally, sample images from the dataset, showcasing both healthy and diseased leaves, are presented for reference, Figure 3.

B. Classification Model

The dataset is then uploaded into Edge Impulse. It is split into training set and test set, with an $80\%, 20\%$ split. This results in 3384 being allocated for training and 846 images for testing.

After the dataset preparation, an *impulse* is created, to process the data and transform it into a format suitable

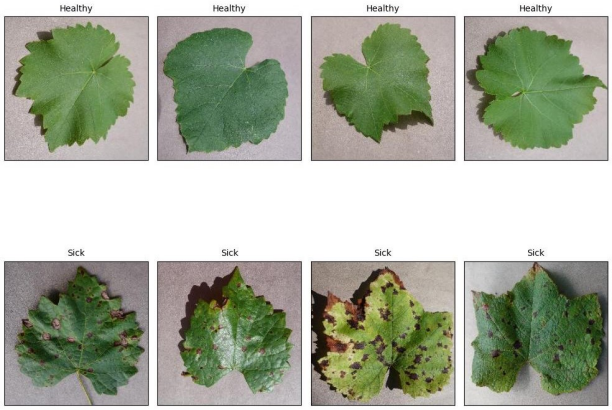


Fig. 3: Sample Images from Dataset

for classification by the model. The images are first rescaled to a size of 96*x*96 pixels to reduce computational complexity while retaining essential features. Next, a *Processing and a Learning Blocks* are added to the *impulse* design. The resulting model is designed as a binary classification system, taking an image as input and outputting one of two possible labels: *Healthy* or *Sick*

Once the data is prepared, transfer learning is employed to adapt a pre-trained deep learning model to a specific dataset. The selected model is *MobileNetV1 96x96*, configured without the final dense layer, with a dropout range of 0.1, a learning rate of 0.0005, 20 training cycles and a default batch size of 32. The chosen model version is the textitQuantized *int8*, optimized for efficient inference on resource-constrained devices.

The pre-trained model achieves an overall accuracy of 85.2%, which could already be considered sufficient for the project's requirements. However, to further improve its performance, the model was retrained using the same architecture and optimized parameters. After retraining, the model achieved a significantly higher accuracy of 93.14%, and a loss of 1.25. The corresponding confusion matrix is presented in the Figure 4. The primary

	HEALTHY	SICK	UNCERTAIN
HEALTHY	91.3%	3.5%	5.2%
SICK	2.8%	95.0%	2.1%
F1 SCORE	0.94	0.96	

Fig. 4: Confusion Matrix of the Retrained Model

requirement is to accurately identify diseased leaves, therefore false negatives for healthy leaves could be acceptable, whereas false negatives for sick leaves are not permissible. Detecting sick leaves is critical for this preliminary analysis of vineyard leaf samples: missing a diseased leaf could result in undetected diseases spreading within the vineyard. The retrained model performs in a highly satisfactory manner for both labels. The detailed evaluation metrics on the validation set of the retrained

model are presented in table I.

Category	Value
Area under ROC Curve	0.95
Weighted Average Precision	0.95
Weighted Average Recall	0.95
Weighted Average F1 Score	0.95

TABLE I: Summary of Model Performance and Dataset.

The *ROC curve* is a graph that plots the *True Position Rate (TPR)* against the *False Positive Rate (FPR)*, while the *Area Under the Curve (AUC)* represents the measure of the classifier to distinguish between classes. The *Weighted Average Precision* is the ratio of correctly predicted positive observation to the total predicted positives, while the *Weighted Average Recall* is the ratio of correctly predicted positives to the all actual positives. The *Weighted Average F1 Score* is the harmonic mean of *Precision* and *Recall*

With these improved metrics, the model is now ready to be exported from Edge Impulse in a *tflite* file extension, which is the *TensorFlow Lite* format optimized for deployment on edge devices. This compact format ensures efficient storage, making it perfectly suited to be deployed on the Nicla Vision. The model is exported as a quantized model, which uses *8-bit* float point number, reducing memory footprint.

C. Deployment on a Edge Device

The Nicla Vision is a compact device developed by Arduino, specifically designed for real-time computer vision applications in resource constrained environments. It integrates a *Cortex-M7* microcontroller and an *ARM Cortex-M4* co-processor, along with a *2MP* color camera, ideal for object detection and classification tasks. Additionally it supports *Wi-Fi* and *BLE (Bluetooth Low Energy)* for wireless communication. The device is compatible with *TinyML* and *MicroPython*, the latter being adopted to write the code deployed on the device. The trained model was deployed on the Nicla Vision along with the corresponding *MicroPython* code which executes automatically on device startup. The Nicla Vision features 2 MB of flash memory, a non-volatile memory used for storing the firmware and model, and 1 MB or RAM, which is utilized during runtime for temporary computations and data storage, [18]. The device's ability to establish a *Wi-Fi* connection was leveraged to enable wireless transmission of classification data. The device operated in *Access Point, (AP) mode*, broadcasting a dedicated network with a predefined *SSID (Service Set Identifier)* and password. This approach ensures independence from external networks, making the system suitable for deployment both in remote vineyard environments and storehouses, where standard *Wi-Fi* access may not be available. Once a client device, which in this case is the laptop, connects to the *AP*, a *TCP socket server* handles communication. The *Transmission Control Protocol (TCP)* is an *Internet Protocol (IP)* that ensure reliable delivery of data, guaranteeing that packets arrive in the correct order and without corruption [19]. To ensure the correct functioning and debugging of the deployed model and custom code, the *OpenMV IDE* was utilized, with *MicroPython* as the programming language of choice for developing and debugging the system, thanks to its lightweight nature, simple and flexible syntax, and the availability of integrated resourceful libraries. One feature derived directly from computer vision that was implemented in the code is *Blob Detection*. This techniques is used to identify and isolate the pixels forming regions in an image with specified properties, such as brightness and color, distinguishing them from their surroundings. These regions, referred as *Blobs* correspond to the area of interest of an image, [9]. The detection is achieved through color thresholding, where a predefined range for green hues isolates the areas of the frame likely corresponding to the grapevine leaf. The identified region, referred to as the *ROI (Region of Interest)* is then cropped, resized and passed to the model for classification. This process not only improves the clarity of the input for the model but also ensures efficient storage of cropped images for later analysis. An example of the detected leaf in the camera frame is showed in the figure 5

Fig. 5: Detected Leaf

D. Data Collection

Once the model and the custom software have been deployed, the system is ready to be operative. The system involves two scripts running concurrently, one on the Nicla Vision, which initiates automatically on power-up, and the other on a separate machine, in this case, a personal laptop.

- Nicla Vision: Image Processing and Classification. The Nicla serves as the data acquisition and processing unit.
    1) The device operates in *AP mode*, creating a *Wi-Fi* network to which the laptop can connect.
    2) Capture images and run *Blob Detection* on every frame to identify potential leaves. The

identified *ROI* is cropped and resized for analysis.
3) The trained *TensorFlow* model performs real-time classification to label the leaf: Healthy or Sick, and it associates confidence probability for the prediction.
4) The device transmits the results to a *TCP* socket server:
   a) Classification Label
   b) Confidence Probability
   c) Cropped image in a compressed *JPEG* format

- Laptop: Data Management and Publication.
  The laptop acts as the receiving unit
  1) The laptop connects to the Nicla Vision *Wi-Fi* network as a client through a *TCP* socket connection.
  2) It sets up the *MQTT (Message Queuing Telemetry Transport)* client.
  3) It receives the data from the device,
  4) Perform a *Similarity Check*.
  5) Encode the image data in *Base64* format.
  6) It publishes image, label and probability to the *MQTT* broker in a *JSON* payload format.

The system is designed with robust error-handling mechanisms to ensure continuous operations and minimize disruptions in the data processing and transmission. LED indicators on the Nicla Vision provide visual feedback to notify the user of the device status. A continuous green led indicates that the device is actively running and processing frames in real-time. When a *Blob* is detected, the LED flashes red, notifying the user of a successful detection and processing. In the event of a *Wi-Fi* disconnection, the laptop automatically attempts to reconnect to the Nicla Vision up to a predefined number of attempts. Similarly, automatic reconnection mechanisms are in place for the *MQTT* broker, ensuring uninterrupted data transmission. Additionally, the system provides status messages in the laptop terminal, keeping the user informed about the workflow and any connectivity or processing issues.

E. Data Publication and Node-RED Integration

The data publication process in the system leverages on *MQTT* protocol as the central communication hub. MQTT is a lightweight, publish/subscribe messaging protocol, designed for reliable communication in resource-constrained environments. It operates over TCP/Ip and is optimized for miminal bandwidth usage and low resource consumption. The protocol is built around a *broker-based* architecture, where *Clients* acts as *publishers* or *subscribers*. A publisher sends messages to specific *topics* to the broker, then the broker forwards it to all clients subscribed to the topic. In the project the laptop acts as an *MQTT* client, re-formatting and publishing the processed data to the *MQTT* brokers under two pre-configured topics:

1) *leaf/prediction* for valid classifications.
2) *leaf/too_similar* for an image too similar to previous published one.

To detect redundancy, the current image is compared with the previous one using the *Structural Similarity Index (SSIM)*. [20]. If the similarity score exceeds a predefined threshold, the image and related data are published under the *leaf/too_similar* topic. MQTT Explorer was utilized during the project to monitor, debug and setup the MQTT broker. This open-source tool provides a structured view of the broker's topic hierarchy, enabling efficient visualization of message flow and payload, along facilitating real-time testing and verification of the system's publish/subscribe communication.

Node-RED is a flow-based programming tool, used to process and visualize the data published to the *MQTT* broker [21]. The *Node-RED* flow subscribes to the desired *MQTT* topic, in this case both of them, and parses the incoming *JSON* payload. Then, it routes the information based on its content. A dynamic dashboard is implemented within *Node-RED*, featuring:

- Text Display: for labels and prediction probabilities.
- Image Rendering: to dynamically display cropped leaf images.
- Storage Control: a toggle to enable or disable data storage.

The dashboard is showed in image 6

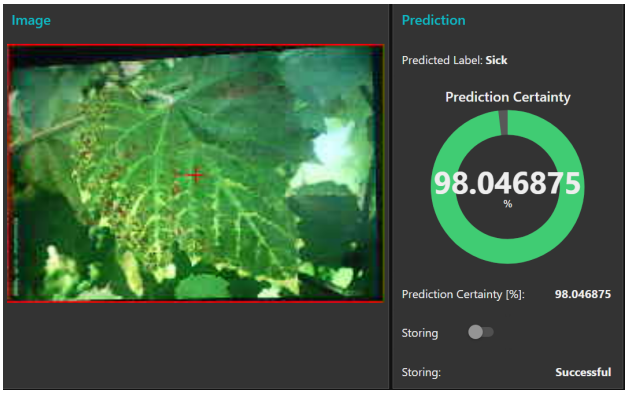


Fig. 6: Node-RED dashboard

The Node-RED flow also incorporates a filtering mechanism to ensure only high-confidence predictions are stored in the database. Specifically, images associated with prediction probabilities below 85% are discarded, in order to maintain data quality and relevance. The selected processed data is then stored in an *InfluxDB* database for further analysis.

F. Database

InfluxDB is an open-source *Time Series Database (TSDB)* specifically designed for efficient handling of time-stamped data. It utilizes *Flux*, a functional query language optimized for retrieving and transforming time-series data, [22]. The stored data is organized into *Buckets*, which are logical containers that include:

- Measurements: groups of related data
- Tags: metadata
- Fields: numerical or textual values

. For this project, the query operates on the *Leaf Database Bucket*, which contains storing time-series data related to the grapevine leaf health predictions. Within this *Bucket*:

- The *Label Prediction* measurements represent the classification results, allowing filtering between labels (*Healthy or Sick*).
- The fields include:
  1) Image: encoded images of grapevine leaves
  2) Probability: The confidence level of the classification

The **Bucket** structure is presented in image 7.

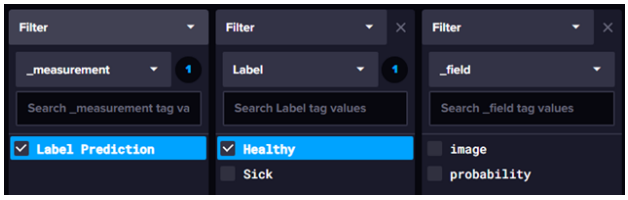


Fig. 7: InfluxDB Bucket structure

The database is visualized using *Grafana*, an open-source platform for creating dynamic and interactive dashboards, [22]. *Grafana* fetches data directly from the *InfluxDB* database, providing real-time monitoring and visualization of stored information. For this project, two primary panels corresponding to the classification labels, *Healthy and Sick*, are implemented to display the processed images, as showed in image 8.

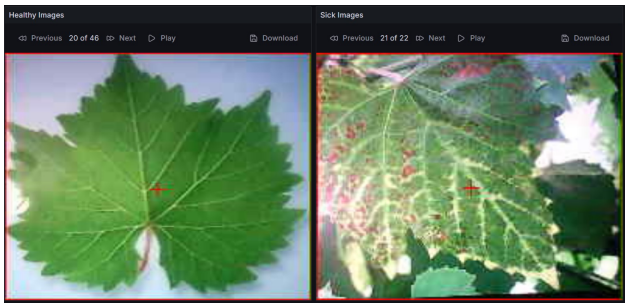


Fig. 8: Grafana main panels

Additionally, smaller panels below each main panel provide the confidence probabilities associated with the classification labels. A summary panel is included to offer an overview of all the data stored in the database, such as image counts, classifications and probabilities, Figure 9. This ensures a quick and comprehensive summary of the system's activity, allowing for efficient monitoring and analysis.

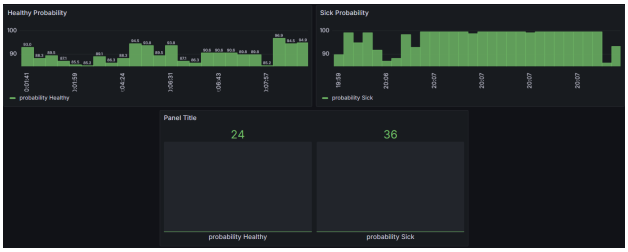


Fig. 9: Grafana secondary panels

III. Results

The project sucessfully achieved its objectives by designing and implementing a robust system for real-time detection and classification of grapevine leaf health, leveraging the capabilities of edge AI and IoT technologies. The intregation of the Nicla Vision for on-device inference with an IoT data management and visualization framework provided an efficient, scalable and autonomous solution for vineyard monitoring. The key outcomes of the project are:

- Accurate Classification.
  The model selected, after retraining achieved an accuracy 93.14%. The focus on correctly idenfitying sick leaves ensures its reliability as a preliminary diagnostic tool for vineyard health monitoring.
- Real-Time Processing.
  The deployment of a quantized *TensorFlow Lite* model enabled real-time inference directly on the Nicla Vision, minimizing latency and eliminating the need for external computational resources. This highlights the system's suitability for resource-constrained environments.
- Data Transmission and Visualization.
  Classification results, including pictures, labels and confidence probabilities were efficiently transmitted via an *MQTT* broker, stored in a *InfluxDB database* and visualized through an interactive *Grafana board*. In addition, the system provided a *Node-RED* dashboard for monitoring of data flow, enabling the user to dynamically decide whether to store the processed images.

The main advantages of the design choices of the system are highlighted as follow:

- Exploiting the Nicla Vision edge computing capabilities, demonstrating the viability of deploying AI models on embedded devices for real world application.

- The use of capability of the Nicla Vision *Wi-FI Access Point (AP) mode*, enabling direct communication with a laptop or other client devices, removing the need for physical connection and making the system increasingly portable and practical for agricultural environments. The automated workflow, which include image acquisition, pre-processing, classification, data transmission and storage, significantly streamlines operations. Additionally, visual feedback on the device via LED indicators, coupled with printed update on the laptop terminal ensure immediate diagnostic and enhance overall usability. The incorporation of blob detection, image similarity filtering and certainty value thresholding before storing the image in the database ensures that only relevant and non-redundant data is stored. This minimizes storage requirements while maintaining data quality for analysis. The modular intregation of *Node-RED*, *InfluxDB*, and *Grafana* provides a scalable framework that can be extended to additional crops or enhanced monitoring capabilities, demonstrating its adaptability to various agricultural contexts.

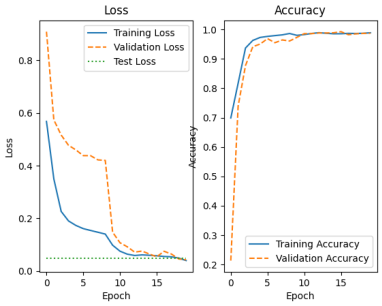
IV. Challenges and Limitations

Significant challenges were encountered during the development and deployment of the machine learning model, particularly in adapting it for use on a resource-constrained device. The initial approach involved implementing a custom, Classification Neural Network using the *PyTorch* library. This model was trained, pruned and fine-tuned to optimize its performance and reduce its size for deployment on the Nicla Vision. The characteristic of the custom model are illustrated in image 10b.

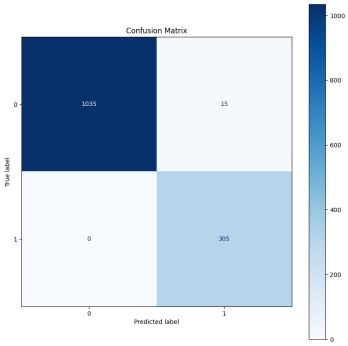
Despite its compact size of only *37 kB* and accuracy of approximately 98%, deploying the *PyTorch* based model on the Nicla Vision was ultimately unceccessful. The primary issue arose from a mismatch in the input tensor conventions between *PyTorch* and *TensorFlow Lite* which is the format supported by *MicroPython* and *OpenMV* on the Nicla Vision.

- *PyTorch* Input Convention: Channel-first format $[1,3,96,96]$, *[batch, channel, width, height]*
- *TensorFlow Lite* Input Convention: Channel-last format $[1,96,96,3]$, *[batch, width, height, channel]*

Efforts to convert the model from *PyTorch* to *TensorFlow Lite* were made using multiple approaches, but none were successful in resolving the format incompatibility. As a result, the Edge Impulse platform was adopted as a solution for generating a *TensorFlow Lite*-compatible model that could be deployed directly on the Nicla Vision. While the model generated by Edge Impulse performs adequately, it came at the cost of increased model size (*87 kB*).



(a) Training of the custom NN



(b) Confusion Matrix of the custom NN

Fig. 10: PyTorch Model.

Blob detection is a simple and straightforward approach. However, alternative methods for detecting leaves before passing them to the model could be explored. One potential improvement could involve implementing a model capable of directly identifying leaves and classifying them as healthy or sick. While this approach might offer a more integrated solution, it could result in less precise labeling and higher computational demand on the edge device, especially given that the system is specifically designed for grapevine leaves.

Lastly, the dataset selected could be a limitation of the system. The task required the classification of individual leaves, leading to the choice of a dataset containing only clear and distinct images of leaves. However, in real-world environment, this is not always the case, since the system would ideally need to perform effectively also on images of multiple overlapping leaves.

V. Future Work and Improvement

Different areas of the project could be improved to enhance its functionality, security and usability:

1) Model Implementation. The most straightforward improvement would be to redesign the custom model originally developed in *PyTorch* using *TensorFlow*. This approach would allow the deploy-

ment of a model using similar compactness and accuracy.

2) Security Enhancements. The current application lacks security measures, leaving both the system and the data vulnerable to potential risks. Implementing encryption for data transmission and storage would significantly improve the system's security.
3) To monitor the system comprehensively, both *Node-RED* and *Grafana* dashaboards need to be open simultaneously. This fragmented setup could be improved by designing a unified dashboard, offering a centralized interface for data management and monitoring.
4) The *Blob* detection is a simple and direct approach, but other approach to detect leaves before feeding them to the model could be implemented. Such a method could be having a model directly able to identify leaves, dividing them between healthy and sick. Such model could be however less precises on the labeling part and heavier on the edge-device.

These improvements would address key limitations of the project and make the system more robust, secure and user-friendly.

VI. Conclusion

This project successfully demonstrated the feasibility and effectiveness of deploying machine learning model on resource-constrained edge device for agricultural applications. The Nicla Vision, equipped with a quantized tensorflow Lite model, proved capable of performing real-time inference with high accuracy. The integration of pre-processing computer vision techniques and a robust IoT pipeline for data transmission, storage and visualization showcased the potential of edge AI in agricultural context. However, the manual tuning of blob detection highlights areas for improvement. Future enhancements, including security measures and unified dashboards, can further optimize the system for practical deployment in real-world vineyard environments.

References

[1] D. Pimentel, *Pest control in world agriculture*, AGRICULTURAL SCIENCES – Vol. II - Pest Control in World Agriculture.

[2] grandviewresearch, *Organic wine market size, share trends analysis report by type (red, white), by packaging (bottles, cans, others), by distribution channel (on trade, off trade), by region, and segment forecasts, 2024 - 2030*, https://www.grandviewresearch.com/industry-analysis/organic-wine-market-report.

[3] WiensCellars, *The rise of organic winemaking: Practices and principles*, https://www.wienscellars.com/the-rise-of-organic-winemaking-practices-and-principles//.

[4] M. S. McLaughlin, M. Roy, P. A. Abbasi, O. Carisse, S. N. Yurgel, and S. Ali, "Why do we need alternative methods for fungal disease management in plants?" *Plants*, vol. 12, no. 22, 2023, ISSN: 2223-7747. DOI: 10.3390/plants12223822. [Online]. Available: https://www.mdpi.com/2223-7747/12/22/3822.

[5] Y. Ampatzidis, L. De Bellis, and A. Luvisi, "Ipathology: Robotic applications and management of plants and plant diseases," *Sustainability*, vol. 9, no. 6, 2017, ISSN: 2071-1050. DOI: 10.3390/su9061010. [Online]. Available: https://www.mdpi.com/2071-1050/9/6/1010.

[6] K. Sun, X. Wang, X. Miao, and Q. Zhao, "A review of ai edge devices and lightweight cnn and llm deployment," *Neurocomputing*, vol. 614, p. 128 791, 2025, ISSN: 0925-2312. DOI: https://doi.org/10.1016/j.neucom.2024.128791. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925231224015625.

[7] S. Hymel, C. Banbury, D. Situnayake, *et al.*, *Edge impulse: An mlops platform for tiny machine learning*, 2023. arXiv: 2212.03332 [cs.DC]. [Online]. Available: https://arxiv.org/abs/2212.03332.

[8] H. Bichri, A. Chergui, and M. Hain, "Image classification with transfer learning using a custom dataset: Comparative study," *Procedia Computer Science*, vol. 220, pp. 48–54, 2023, The 14th International Conference on Ambient Systems, Networks and Technologies Networks (ANT) and The 6th International Conference on Emerging Data and Industry 4.0 (EDI40), ISSN: 1877-0509. DOI: https://doi.org/10.1016/j.procs.2023.03.009. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1877050923005446.

[9] J. M. Joseph Howse, *Learning OpenCV 5 Computer Vision with Python: Tackle computer vision and machine learning with the newest tools, techniques and algorithms, 4th Edition*, 4th ed. Packt Publishing - ebooks Account, 2025, ISBN: 1803230223; 9781803230221.

[10] R. Mandal, *Grapevine disease dataset (original)*, 2023. DOI: 10.34740/KAGGLE/DS/2928682. [Online]. Available: https://www.kaggle.com/ds/2928682.

[11] M. Szabó, A. Csikász-Krizsics, T. Dula, *et al.*, "Black rot of grapes (guignardia bidwellii)—a comprehensive overview," *Horticulturae*, vol. 9, no. 2, p. 130, 2023.

[12] D. Molitor and M. Beyer, "Epidemiology, identification and disease management of grape black rot and potentially useful metabolites of black rot pathogens for industrial applications–a re-

view," *Annals of applied biology*, vol. 165, no. 3, pp. 305–317, 2014.

[13] E. Beris, M. Selim, D. Kechagia, and A. Evangelou, "Overview of the esca complex as an increasing threat in vineyards worldwide: Climate change, control approaches and impact on grape and wine quality," in *Recent Advances in Grapes and Wine Production*, A. M. Jordão, R. Botelho, and U. Miljić, Eds., Rijeka: IntechOpen, 2022, ch. 3. DOI: 10.5772/intechopen.105897. [Online]. Available: https://doi.org/10.5772/intechopen.105897.

[14] J. Kenfaoui, N. Radouane, M. Mennani, *et al.*, "A panoramic view on grapevine trunk diseases threats: Case of eutypa dieback, botryosphaeria dieback, and esca disease," *Journal of Fungi*, vol. 8, no. 6, p. 595, 2022.

[15] L. Ouadi, E. Bruez, S. Bastien, *et al.*, "Ecophysiological impacts of esca, a devastating grapevine trunk disease, on vitis vinifera l.," *PloS one*, vol. 14, no. 9, e0222586, 2019.

[16] T. Komatsu and N. Kondo, "Winter habitat of xylophilus ampelinus, the cause of bacterial blight of grapevine, in japan," *Journal of General Plant Pathology*, vol. 81, pp. 237–242, 2015.

[17] P. Mooijman, C. Catal, B. Tekinerdogan, A. Lommen, and M. Blokland, "The effects of data balancing approaches: A case study," *Applied Soft Computing*, vol. 132, p. 109 853, 2023, ISSN: 1568-4946. DOI: https://doi.org/10.1016/j.asoc.2022.109853. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1568494622009024.

[18] arduino, *Abx00051-datasheet*, https://docs.arduino.cc/hardware/nicla-vision/.

[19] A. O. OROGUN, *Brief insight into transmission control protocol (tcp)*, 2017.

[20] Z. Wang, A. Bovik, H. Sheikh, and E. Simoncelli, "Image quality assessment: From error visibility to structural similarity," *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 600–612, 2004. DOI: 10.1109/TIP.2003.819861.

[21] M. Lekić and G. Gardašević, "Iot sensor integration to node-red platform," in *2018 17th International Symposium INFOTEH-JAHORINA (INFOTEH)*, 2018, pp. 1–5. DOI: 10.1109/INFOTEH.2018.8345544.

[22] M. D. Matteo Nardello Davide Brunelli, *Iot course notes 2024, university of trento*.