CORN LEAF DISEASE DETECTION

Ilenia Ficili

September 30, 2023

1 Project Introduction

The agricultural sector plays a fundamental role in sustaining human life and ensuring food security. Corn, as a staple crop, is a significant contributor to global nutrition and economy. However, diseases affecting corn plants pose a substantial threat to agricultural productivity and food supply. Timely and accurate detection of diseases in corn plants is vital for effective disease management and ensuring optimal crop yields. Leveraging advanced technologies and machine learning, project objective is to provide farmers and agricultural stakeholders with a tool that can swiftly identify and categorize potential diseases based on visual cues present in the plant's leaves. By providing a means to swiftly identify diseases, farmers can take proactive measures to mitigate the impact of these diseases. Early detection allows for targeted interventions such as precise pesticide application or other disease control strategies, ultimately leading to better crop health and higher agricultural productivity. This project is conceived with a multifaceted vision that integrates cutting-edge technology and machine learning to revolutionize agricultural surveillance. While the primary focus is the development of a machine learning model for disease detection in corn plants, we envisage a broader implementation that leverages drone technology equipped with advanced imaging capabilities. The envisioned integration of drones would enable aerial surveys of agricultural regions, offering a comprehensive view of the farm's health and empowering farmers to identify potential problem areas that may require closer inspection. The integration of machine learning models, particularly in disease detection, and the potential incorporation of drone technology for precision monitoring presents an innovative solution to the challenges faced by modern agriculture. The disease detection model developed as part of this project allows rapid identification and classification of diseases based on visual cues present in the plant's leaves. Looking ahead, the envisioned utilization of drones would further extend the capabilities of this system, offering a bird's-eye view of the entire agricultural landscape and enabling farmers to make data-driven decisions for optimized crop management.

2 Dataset

The dataset that has been used is the 'Corn or Maize Leaf Disease Dataset' available on Kaggle. This dataset has been created using two already existing ones: PlantVillage and PlantDoc datasets. During the formation of the dataset certain images have been removed beacuse were not found to be useful.

The dataset is divided in 4 folders each representing different states or health status of the corn plant. The folder are: 'Common Rust', that contains 1306 images, 'Gray Leaf Spot' with 574 images, 'Blight' with 1146 images and 'Healthy' with 1162 images.

2.1 Common Rust

Common Rust, caused by the fungus Puccinia sorghi, is a pervasive fungal disease that affects corn plants. The lifecycle of this pathogen involves alternate hosts, specifically a type of Euphorbia plant. During spring, the fungus releases spores that can be transmitted over long distances by wind and rain. Infection begins when the spores land on the leaves. The disease thrives in conditions of high relative humidity (around 100%), dew, rain, and temperatures ranging between 15 and 20°C. Younger leaf tissues are more susceptible to fungal infection than fully developed leaves. Infected plants exhibit fine speckling on both sides of the leaves, which eventually turns into small, dark, and slightly raised spots. These spots become golden-brown pustules, irregularly scattered on the leaf surfaces. The

stems of the infected plants weaken and become more susceptible to breakage. Younger leaf tissues are more susceptible to fungal infection, leading to weakened stems and increased susceptibility to breakage. Infected plants may exhibit yellowing due to chlorophyll deficiency, potentially resulting in plant death if the upper leaves are severely affected. Efficient disease management strategies involve planting resistant corn varieties, crop rotation, and timely fungicide application when necessary. Early detection and appropriate preventive measures are crucial for mitigating the impact of Common Rust and ensuring healthy corn yields.

Figure 1: insert caption and add reference in the text

2.2 Gray Leaf Spot

Corn Gray Leaf Spot, caused by the fungus Cercospora zeae-maydis, is a prominent fungal disease that poses a significant threat to corn plants. The lifecycle of this pathogen revolves around spores that typically overwinter in infected plant debris. As the growing season begins, the spores become airborne and disperse to nearby plants. The infection occurs when these spores land on corn leaves, especially favoring the lower leaves. The initial symptoms manifest as small, tan to brown lesions with distinct yellow halos, often observed on the leaves. Under favorable conditions such as high humidity and moderate temperatures, these lesions can coalesce and grow, forming larger, irregularly shaped spots. As the disease progresses, the spots may turn gray, giving the disease its characteristic name. Gray Leaf Spot significantly impacts the plant's ability to perform photosynthesis, as the lesions compromise the leaf's surface area and functionality. The infected leaves gradually turn yellow and may die prematurely, further reducing the plant's ability to produce energy. This has a cascading effect on the plant's overall health and productivity, potentially leading to reduced yields. Management and prevention strategies for Gray Leaf Spot involve crop rotation, proper disposal of infected plant material, and, in severe cases, application of fungicides. Early detection and prompt intervention are essential to minimize the disease's impact on the corn crop and ensure a healthy and productive harvest.

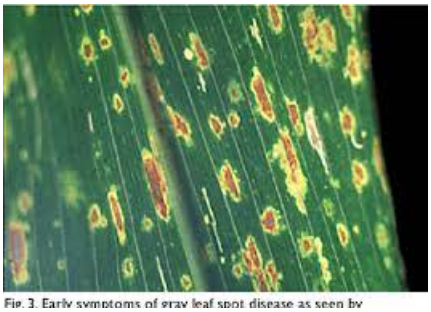

Figure 2: insert caption and add reference in the text

2.3 Blight

Blight is a plant disease affecting corn plants, caused by various types of fungi such as Exserohilum turcicum and Helminthosporium maydis. It is a common and potentially devastating disease that

can significantly impact crop yields if not managed effectively. The disease manifests in the form of irregular, elongated lesions on leaves, stalks, and ears of corn plants. These lesions are initially small, dark brown or tan spots that can rapidly expand, forming large, deadened areas. The lesions are often surrounded by yellow halos and may produce a dark fungal growth, particularly under moist conditions. Blight typically spreads through airborne spores, rain splash, or direct contact with infected plant material. Warm, humid weather provides an ideal environment for the fungus to thrive and spread rapidly. In severe infections, the entire plant can become blighted, leading to stunted growth, reduced vigor, and ultimately, decreased crop yield. To manage and prevent blight, farmers employ a range of strategies including planting resistant varieties, crop rotation, timely fungicide application, and ensuring good air circulation between plants. Early detection and immediate action are crucial in minimizing the disease's impact and safeguarding the corn crop.

Figure 3: insert caption and add reference in the text

3 Data Preparation and Preprocessing

To ensure a representative distribution of classes in each subset, a stratified approach was adopted for dividing the dataset into training, validation, and test sets. The train_test_split function, provided by the scikit-learn library, was employed for this purpose. The training set constitutes the majority of the dataset and serves as the primary set for training the machine learning model. Subsequently, the validation set, a smaller proportion of the dataset, was further extracted from the training set to evaluate the model's performance during training. Finally, a distinct test set was maintained to assess the model's performance after its training and validation. To enhance the robustness and generalization of the model, image augmentation was employed. This involved applying random rotations and horizontal flips to diversify the dataset without changing the underlying class semantics. The structured dataset, comprising image file paths and corresponding labels, was then utilized to create a DataFrameIterator for each subset (training, validation, and test). Each image was resized to a standardized dimension of 224x224 pixels, and the color representation was set to RGB. The class labels were encoded in a categorical format, ensuring compatibility with the model architecture. To validate the preprocessing and augmentation, a subset of the training set was visually examined. Images were rescaled and displayed, providing a glimpse into the dataset's preprocessed form. This step was crucial to ensure that the images were appropriately processed and ready for model ingestion.

Undoubtedly, a notable challenge lies in the significant imbalance present within the dataset. Certain classes, such as 'Healthy leaves,' are notably overrepresented, boasting a significantly larger number of images compared to other classes like 'Blight.' This disparity can introduce a bias in the model during training, causing it to incline towards predicting the majority class (Healthy leaves) at the expense of accurately identifying the diseases (minority classes). To mitigate this bias are implemented specialized techniques during model training. To address the issue of imbalanced class distribution, one effective strategy is to incorporate class weights during the model training process. Class weights provide a mechanism to amplify the influence of minority classes (e.g., Blight, Common Rust, Gray Leaf Spot) by assigning them higher weights, while reducing the weight assigned to the majority class (Healthy leaves). This deliberate weighting encourages the model to prioritize learning from the underrepresented disease classes, enhancing its proficiency in accurately detecting and categorizing them. The weights are computed based on the respective class frequencies, aiming to balance the impact of each class on the overall loss function during training. By employing this approach, the model is guided

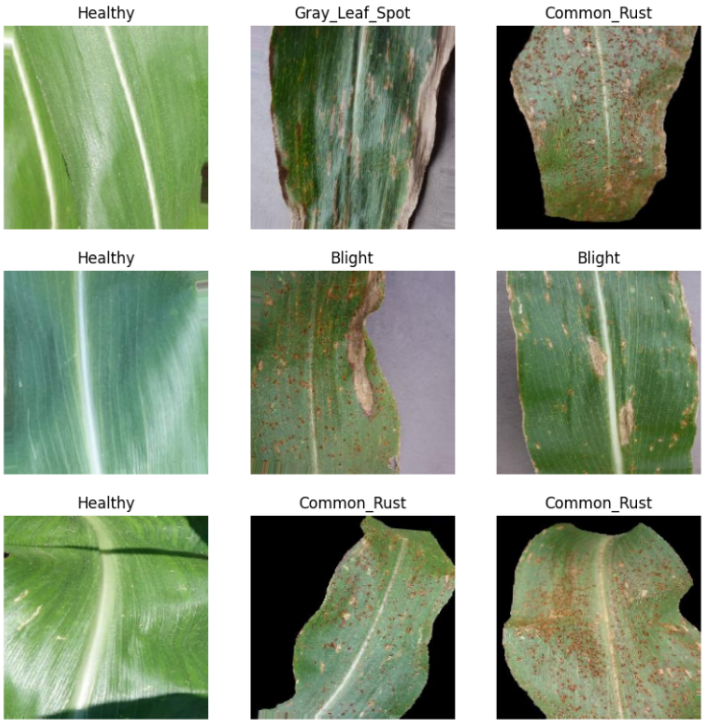

Figure 4: insert caption and add reference in the text

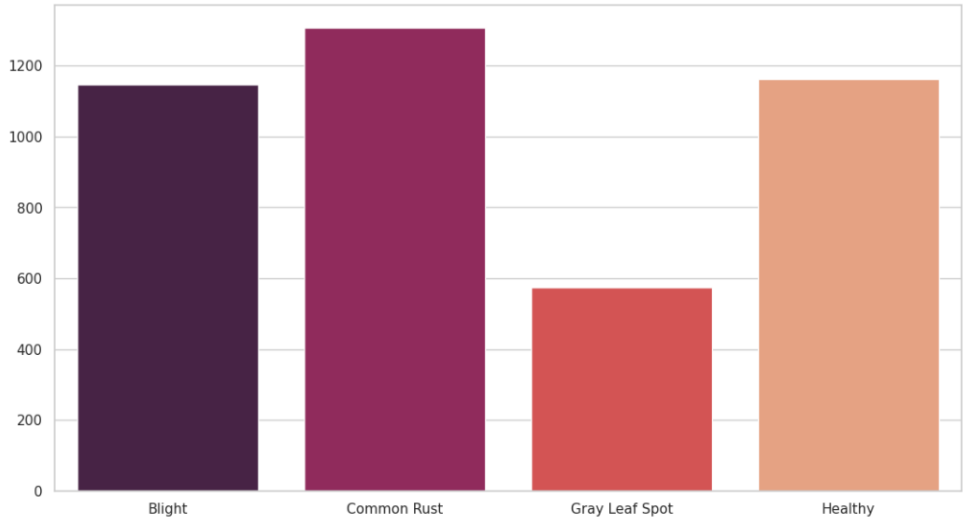

Figure 5: insert caption and add reference in the text

to effectively distinguish between all classes, regardless of their initial imbalances in the dataset, thus ensuring a more comprehensive and equitable learning process.

4 Convolutional Neural Network

In this project, a MobileNetV2 model pre-trained on the ImageNet dataset was chosen as the fundamental architecture for the machine learning model. MobileNetV2, recognized for its efficiency, lightweight design, and efficacy in image-related tasks, provided an excellent foundation for this project.

The MobileNetV2 model was configured with an input shape of (224, 224, 3), aligning with the preferred input dimensions for MobileNetV2. The top classification layers were excluded (include_top=False), and pre-trained weights were utilized ('imagenet') to harness the learned features from the extensive ImageNet dataset. To tailor the pre-trained model for the specific task of classifying corn plant leaf diseases, a custom classifier was constructed on top of the MobileNetV2 base. Employing a functional API, global average pooling was applied to condense the spatial dimensions of the base model's output. A dense layer with 4 units (representing the disease classes) and a softmax activation function was subsequently added to yield the final predictions. During model training, the Adam optimizer was employed, utilizing categorical cross-entropy as the loss function. Given the substantial class imbalance in the dataset, class weights were integrated to address this challenge. These weights were computed based on class frequencies, striving to ensure effective learning of all classes, irrespective of their initial imbalance. The training transpired in two phases. Initially, the model underwent training for 15 epochs. Subsequently, the last 11 layers of the base MobileNetV2 model were unfrozen for fine-tuning. The learning rate of the optimizer was adjusted, and fine-tuning persisted for an additional 10 epochs. The training process was meticulously tracked using various metrics, including accuracy, precision, recall, training loss, and validation loss, to validate the model's performance and convergence.

4.1 Structure of the model

The model architecture consists of a pre-trained MobileNetV2 convolutional neural network (CNN), acting as the base model. This CNN is utilized for its proficient feature extraction capabilities, leveraging pre-trained weights from the ImageNet dataset. The MobileNet v2 architecture is based on an inverted residual structure where the input and output of the residual block are thin bottleneck layers opposite to traditional residual models which use expanded representations in the input. MobileNet v2 uses lightweight depthwise convolutions to filter features in the intermediate expansion layer. The base model takes an input image of dimensions (224, 224, 3), where 3 represents the RGB color channels. Its primary role is to recognize intricate patterns and features within the image. Following the base model, a Global Average Pooling 2D layer is applied. This layer aggregates spatial information by computing the average of each feature map. This process significantly reduces the number of parameters and consolidates the information for subsequent layers. The final layer is a Dense layer, employing a Softmax activation function. This layer acts as the classifier, mapping the extracted features to distinct classes. In this case, there are four classes: Blight, Common Rust, Gray Leaf Spot, and Healthy. The Softmax activation ensures that the output is a valid probability distribution across these classes. The sequential arrangement of these layers enables the transformation of input images into meaningful predictions. The base model extracts intricate features, the Global Average Pooling layer condenses high-dimensional features, and the Dense layer with Softmax activation produces the probability distribution required for accurate classification. This structural design facilitates effective feature extraction and precise classification, crucial for disease detection in corn leaves.

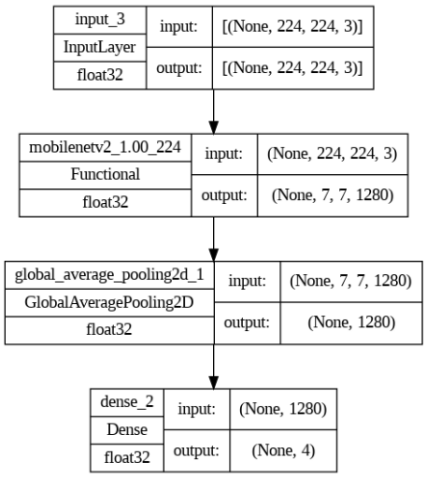

Figure 6: Model structure

5 Metrics and results

The effectiveness and performance of the implemented machine learning model were evaluated through rigorous training, validation, and testing procedures. The model was trained on a carefully curated dataset and fine-tuned to achieve optimal results. In this section, we present a comprehensive analysis of the results, including training and validation performance, as well as insights from the test set. The model was initially trained over a specified number of epochs, utilizing a training dataset with augmented images to enhance generalization. The training process involved optimizing the categorical cross-entropy loss function using the Adam optimizer, while monitoring key metrics such as accuracy, precision, and recall. The training and validation loss curves, depicted in Figure 7, illustrate the model's progression throughout the training epochs. The training loss decreased steadily, indicating effective learning and adaptation to the dataset. Concurrently, the validation loss exhibited a similar trend, confirming the model's ability to generalize well to unseen data. Following the initial training, the

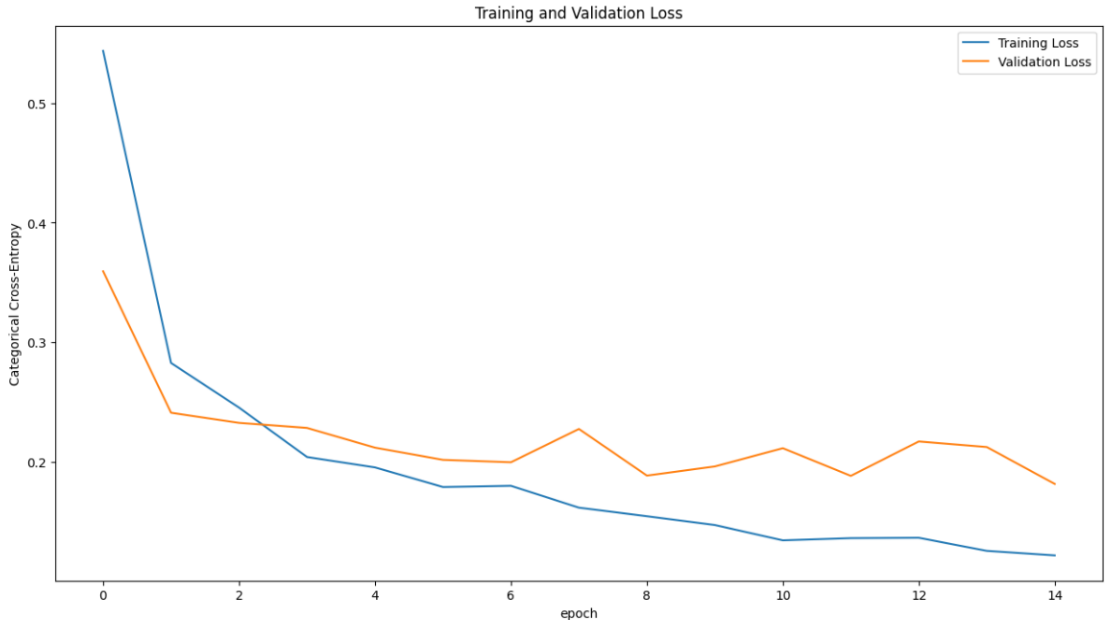

Figure 7: Training and validation loss

model underwent a fine-tuning phase, involving the unfreezing of specific layers for further adaptation. This refinement was facilitated by an additional 10 epochs and the refined model's progression during this phase is captured in Figure 8, showcasing its continued evolution.

After thorough evaluation on the test dataset, the model achieved compelling performance with a loss of 0.1903. Furthermore, key classification metrics include a precision of 0.9314, a recall of 0.9236, and an accuracy of 92.60%. These metrics collectively highlight the model's robustness in effectively identifying and classifying the corn plant diseases. To gain further insights into the model's performance, a confusion matrix was generated using the test set. The confusion matrix provides a detailed breakdown of the model's predictions compared to the actual labels. A confusion matrix is a representation widely used to evaluate the performance of a machine learning classification model, particularly in tasks where the model is intended to predict categorical outcomes. In a classification problem, the confusion matrix provides a clear visualization of the model's performance in predicting the true labels of the dataset, displaying a summary of the predictions made by the model against the actual true labels. The resulting confusion matrix, illustrated in Figure 9, offers a comprehensive breakdown of the model's predictions with the actual labels.

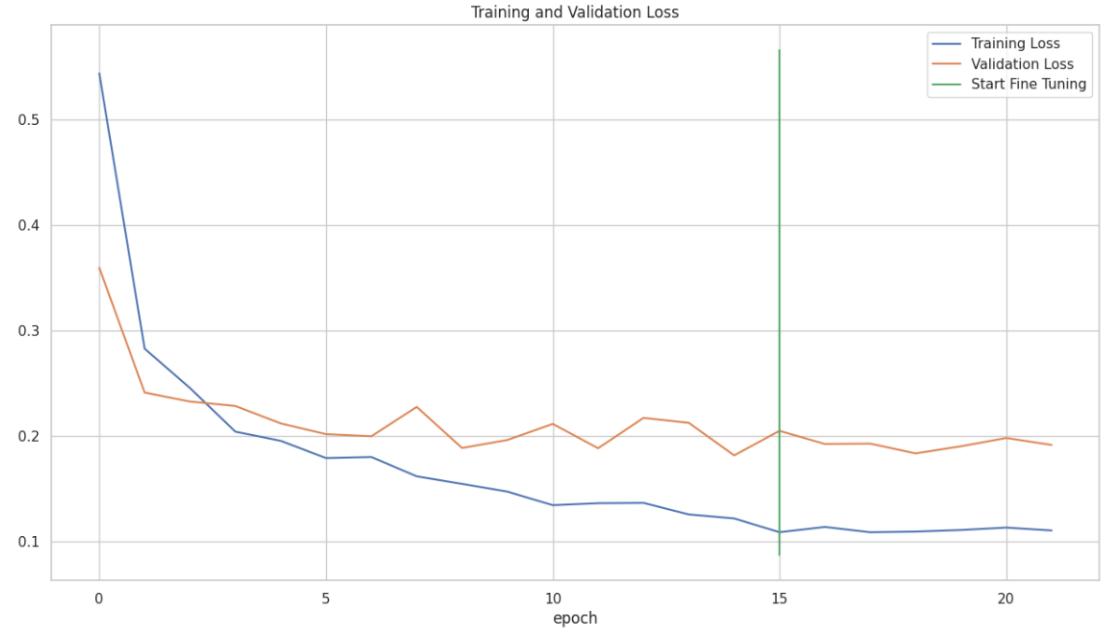

Figure 8: Training and validation loss after fine-tuning

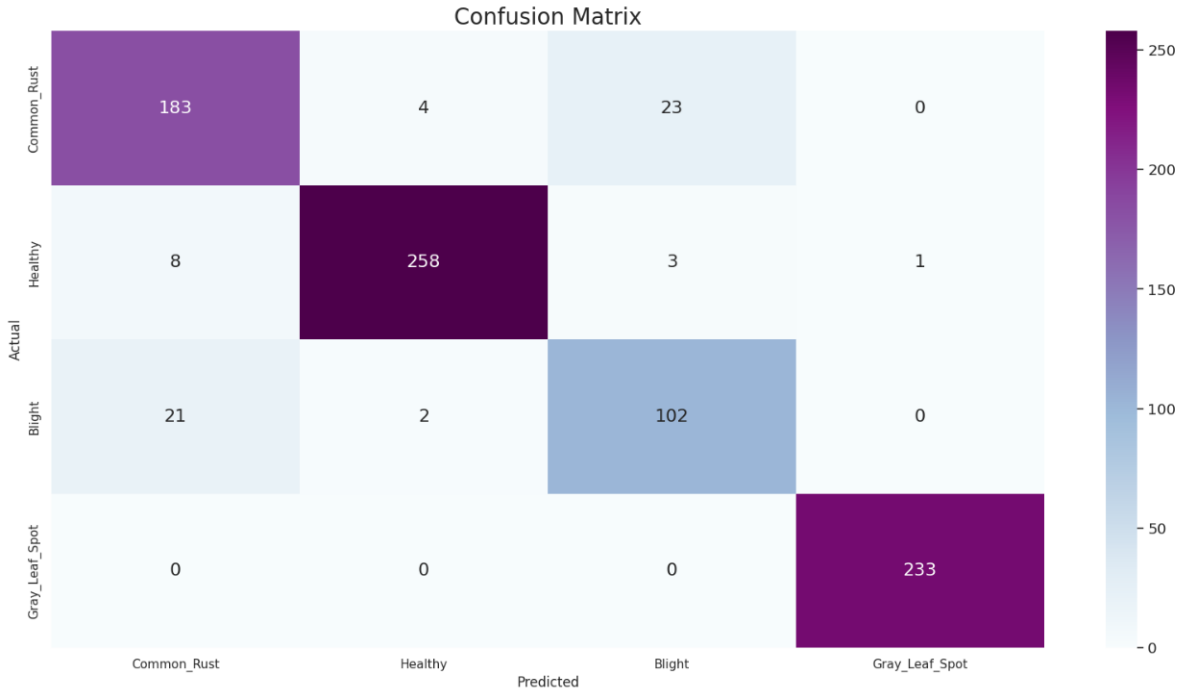

Figure 9: Confusion matrix

6 Stack4Things

The rapidly evolving landscape of Internet of Things (IoT) within Information and Communication Technology (ICT) has witnessed the proliferation of dispersed sensor- and actuator-hosting platforms, presenting both a challenge and an opportunity. Addressing this, Stack4Things emerges as a noteworthy framework, fundamentally rooted in the Sensing-and-Actuation-as-a-Service paradigm.
Leveraging the power of OpenStack, a well-established Cloud computing resource management framework, Stack4Things extends its capabilities to comprehensively manage the dynamic infrastructure of sensing and actuation resources in the IoT realm. By employing an infrastructure-oriented approach and adopting Cloud-focused design principles, it effectively navigates the intricate communication requirements and scalability concerns inherent in IoT, positioning itself as an important tool.
Stack4Things is an innovative project embodying a bottom-up approach, integrating a blend of cutting-edge technologies and protocols. At its core, the project leverages technologies like OpenStack, WebSocket, and the Web Application Messaging Protocol (WAMP). A fundamental principle of Stack4Things is its commitment to openness, exemplified through the development of all software components as open source and freely accessible on the web.
The initiative explores the synergy between Internet of Things (IoT) and Cloud technologies, with Cloud acting as a vital infrastructure for service composition. Notably, Stack4Things places emphasis on managing IoT infrastructure using a Cloud- and service-centric approach, focusing on high-level IoT resource management and interactions with the infrastructure manager. Communication challenges are addressed through the adoption of a WAMP-based protocol layered on an IaaS subsystem, powered by OpenStack.
The core concept is embodied in Sensing-and-Actuation-as-a-Service (SAaaS), which involves the management of sensing and actuation devices much like standard Cloud resources, allowing for remote control and seamless provisioning. In this envisioned scenario, contributors actively provide the essential resources, while end-users wield these resources to mold and oversee the SAaaS infrastructure. This dynamic allows for the creation of a broad spectrum of applications and services. Nodes, operating as hardware-constrained units, assume a crucial role in integrating and facilitating the interaction of sensing and actuation resources within this innovative framework.

Stack4Things (S4T), due to its robust infrastructure for managing sensing and actuation resources, proves valuable for the machine learning project defined in this chapter, where there is also to take in consideration the presence of drones. S4T, based on OpenStack, offers a flexible framework that can be tailored to efficiently integrate various sensors and actuators on drones.
By utilizing S4T's communication capabilities like Websockets, real-time data transmission and control can be achieved, essential for machine learning algorithms. The ability to manage and process data collected by drone sensors, and enable real-time communication and control, is pivotal. Additionally, S4T's features for remote device management and customization empower remote configuration and adaptation of drone behavior based on machine learning algorithms. Its scalability and resource management features are crucial for optimizing resource allocation, a key aspect especially when dealing with a significant number of devices.
The idea of the project is to have each drone equipped with a camera and performs classification based on the designed ML model. S4T's architecture allows for seamless integration of drones, efficiently managing the data collected by their cameras. The real-time communication facilitated by S4T ensures prompt data transmission and model predictions, enabling timely decision-making during drone operations.
Moreover, the platform's remote device management feature is invaluable, allowing to update and fine-tune the ML model across drones without manual intervention. As each drone processes its captured data through the ML model, S4T ensures that these computations are optimized and the results are efficiently shared and stored. This integration optimizes data processing, enhances classification accuracy, and enables a dynamic and adaptive system for the ML classification project.

6.1 Stack4Things Architecture

The design of the S4T platform is split between an OpenStack data-center where a subsystem called IoTronic is deployed and a set of geographically distributed IoT devices running the S4T device-side agent named Lightning-Rod (LR).

As hardware setup for IoT devices, are considered relatively smart embedded devices such as the Single Board Computers (SBCs) powered with a (low power) microprocessor (MPU) units such as Raspberry Pi and modern Arduino boards. Being MPU equipped, such devices are fit to host a minimal Linux distro (e.g., OpenWRT) and thus, they become able to host a range of runtime environments (e.g., Python, Node.js) as well as some Linux-based tools required by the board-side agent, LR.
The communication between IoT devices and the Cloud relies on tailored mechanisms and protocols, particularly utilizing Websockets with a reverse tunneling approach to navigate NATs and firewall restrictions.
What distinguishes S4T (IoTronic) is its compatibility with OpenStack, designed to align seamlessly with the standard architecture of OpenStack services (Figure 3.10). Consequently, this thoughtful design allows for the integration of IoTronic with various OpenStack subsystems like Keystone, Neutron, enhancing the user experience.
Within this framework, S4T offers a range of capabilities: user authentication and authorization are managed effectively using OpenStack's identity service, Keystone; remote access and management are facilitated through Cloud-enabled service forwarding, allowing users easy access to their IoT devices regardless of their location or network configurations, thanks to a reverse WebSockets tunneling mechanism; and tailored customization and contextualization are empowered, enabling users to define application logic within the Cloud as functions and deploy them dynamically, adhering to authorization and privacy policies, on their remote IoT devices. S4T also provides a choice between Python and Node.js as runtime environments, ensuring flexibility and a user-friendly experience.

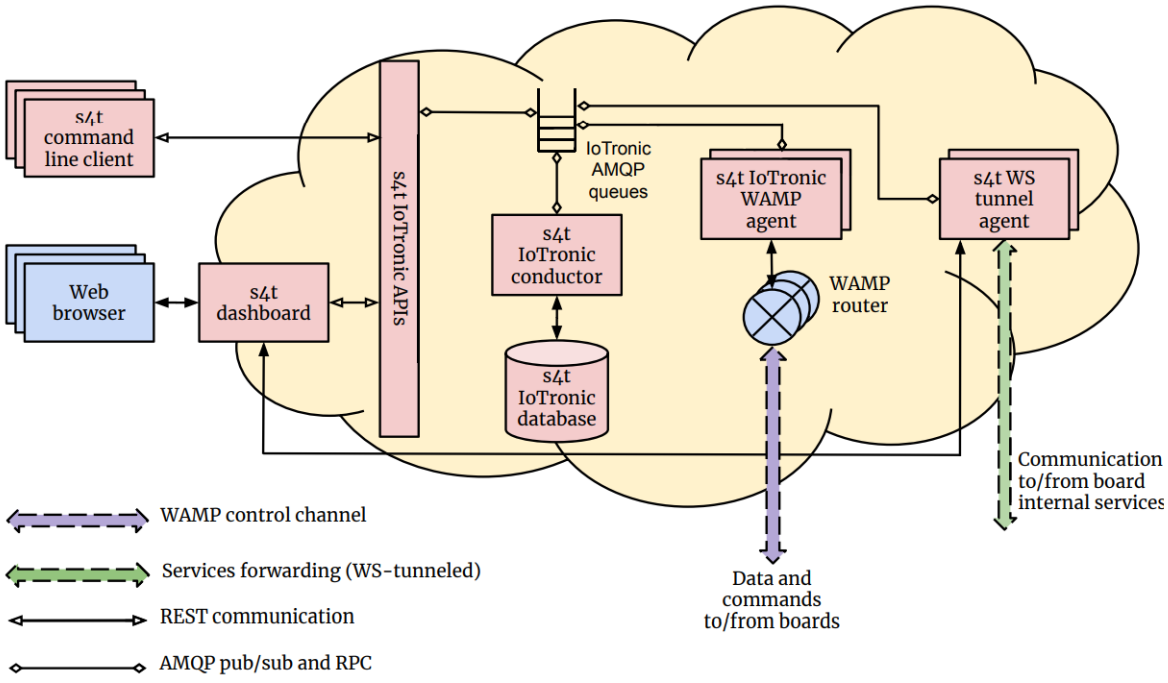

Figure 10: The S4T Cloud-side architecture design.

7 Drone simulation

For implement the drone movement on a designed area, it was simulated the project scenario using ROS 2 Foxy and PX4-Autopilot.

7.1 PX4 Autopilot

PX4 Autopilot is an open-source autopilot system primarily used in unmanned aerial vehicles (UAVs), commonly known as drones. It provides a comprehensive flight control software solution for a variety of autonomous vehicles. The PX4 project includes flight control algorithms, drivers for various sensors and peripherals, and a communication middleware for interfacing with ground stations and other software. In addition, for simulation, gazebo-classic will be used, which simulates multiple robots in a 3D environment, with extensive dynamic interaction between objects.

7.2 ROS 2 Foxy

ROS 2 (Robot Operating System 2) Foxy Fitzroy is one of the releases of ROS, which is a flexible and robust open-source framework for developing robotic systems. ROS 2 is an evolution of ROS, aiming to improve on its limitations. Foxy is one of the versions of ROS 2 and offers various features and improvements over previous versions, including better real-time support, enhanced security, and improved performance. The main characteristics of ROS 2 include support for systems with multiple robots rather than applications based on individual robots. It also provides Quality of Service (QoS) support, granting programmers of publisher and subscriber nodes greater control over message exchange. ROS 2 uses the DDS middleware for message exchange, offering enhanced flexibility and suitability for distributed systems. ROS 2 offers three different types of interfaces, and consequently, three different types of nodes:

- Topic: This node type is ideal for ongoing data streams, persistently publishing data without being dependent on any subscriber. It establishes a many-to-many connection.
- Service: Nodes of this type operate on a request/response mechanism, swiftly concluding routine calls. A service involves a client/server pair.
- Action: Actions are employed to execute routines that endure longer than Services, yet they utilize a communication method similar to that of Services.