Technical Report: FrostAway System for Agricultural Frost Monitoring and Prevention

Maximiliano Militzer
Francisco Wulf
Diego Costa
Ignacio Muñoz
Gabriel Miranda Contreras
Maite Estay

Date: 6 Oct 2024

summary

The FrostAway system offers a comprehensive solution to predict and mitigate frost events in agricultural settings. By combining a network of IoT sensors, a machine learning model, and automated alert systems, it enables farmers to receive early frost warnings and take preventive measures. The system captures real-time data on temperature, humidity, pressure, and wind speed through strategically placed sensor devices across the field. This data is processed by a neural network model to generate frost predictions with 24 hours of anticipation.

Key components of the FrostAway system include solar-powered guardians for data collection, LoRa-based communication for long-range connectivity, and a machine learning pipeline that integrates spatial and temporal analysis for accurate predictions. The system is built using Python, Django, PostgreSQL for backend processing, and deployed on Vercel for frontend user interaction. Alerts are sent through SMS, WhatsApp, and radio, enabling rapid response to frost threats. The low cost and scalability of FrostAway make it a viable solution for medium and large agricultural operations, contributing significantly to reducing crop losses due to frost.

1. Introduction

Frost events pose a significant risk to agricultural productivity worldwide, causing substantial losses in crop yields. Traditional methods of frost prevention, such as manual monitoring and reactive solutions, are often insufficient due to the unpredictable nature of frost. FrostAway was developed to address this problem by using modern technologies, including Internet of Things (IoT) devices, machine learning, and automated alert systems, to monitor environmental conditions and predict frost occurrences with high accuracy.

2. System Architecture

The FrostAway system is composed of three main components: the hardware (sensor network), the backend for data processing and machine learning, and the frontend for user interaction and alert management. These components are interconnected through a robust infrastructure, enabling real-time data collection, analysis, and dissemination.

3. System Workflow and Components

The following diagram (Figure 1) illustrates the overall workflow and key components of the FrostAway system, showcasing how data is collected, processed, and used to trigger automated frost prevention methods in agricultural fields.

The FrostAway system begins with the installation of IoT devices, referred to as "guardians," which are strategically placed across agricultural zones to monitor environmental conditions in real-time. These guardians collect crucial data such as temperature, humidity, wind speed, and atmospheric pressure, all of which are transmitted to the backend for analysis using LoRa technology, ensuring low-power and long-range communication.

Once the data is ingested into the backend, it is processed using machine learning algorithms, specifically a combination of Graph Neural Networks (GNN) and Recurrent Neural Networks (RNN). The GNN component processes spatial relationships between different zones, while the RNN analyzes temporal weather patterns to make frost predictions. Based on these predictions, the system generates alerts via multiple communication channels, including SMS, WhatsApp, and radio.

The frontend interface allows users to visualize real-time telemetry data on a map, define control zones, and manage frost prevention methods such as sprinklers, heaters, and fans. Additionally, users receive recommendations on the most suitable actions to mitigate the risk of frost.

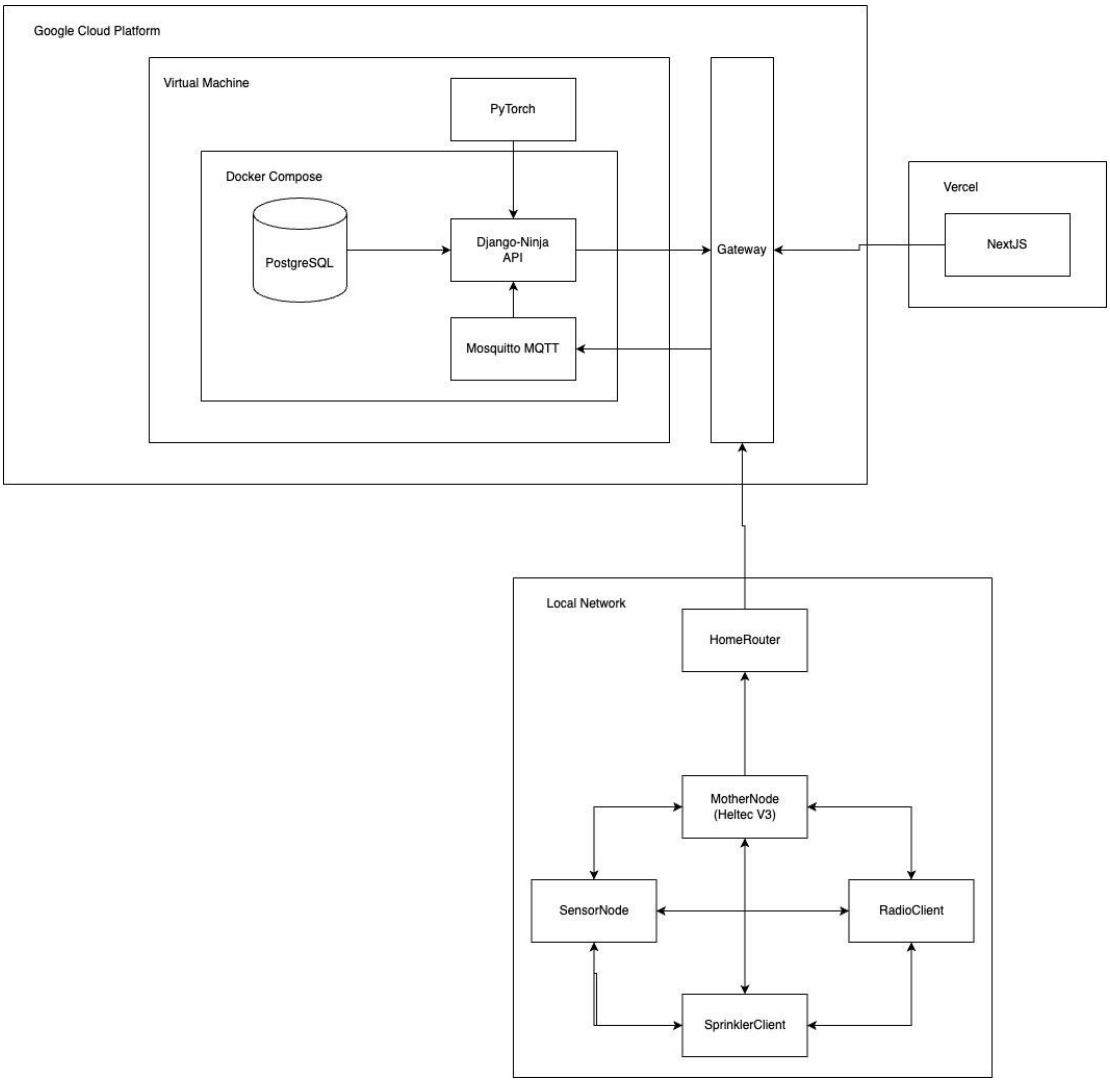

Figura 1. *FrostAway System Workflow*

3.1. Hardware: Sensor Network

The hardware component consists of several "guardians" or IoT devices, each equipped with a BME280 sensor for capturing temperature, pressure, and humidity, as well as wind speed measurements. These devices are strategically placed across the field and are powered by solar panels. Data from the guardians are transmitted to the backend using LoRa communication technology, ensuring low-power, long-range connectivity.

3.2. Backend: Data Processing and Machine Learning

The backend, built using Python, Django, and PostgreSQL, handles data ingestion, storage, and processing. It includes an MQTT broker (Mosquitto) to manage real-time data streams from the field sensors.

The machine learning model is built using Pytorch and relies on a hybrid architecture combining Graph Neural Networks (GNN) for spatial data processing and Recurrent Neural Networks

(RNN) for temporal data analysis. The model predicts the likelihood of frost events based on historical weather data and real-time sensor readings. All computations are containerized using Docker to ensure scalability and portability across various cloud platforms.

3.3. Frontend: User Interface and Alerts

The user interface, built with Next.js and deployed on Vercel, provides farmers with real-time insights into weather conditions and frost risks. It includes an intuitive dashboard that displays key metrics such as temperature trends, humidity levels, and predicted frost events. Additionally, the frontend supports SMS, WhatsApp, and radio alerts, which are sent via Twilio when the system detects an impending frost event.

4. Machine Learning Model

The machine learning model at the core of FrostAway is designed to provide accurate frost predictions by analyzing environmental data from the sensor network.

4.1. Model Architecture

The model combines two primary components:

- **Graph Neural Networks (GNN)**: Used to process the spatial relationships between different sensor nodes in the field. This allows the model to account for variations in environmental conditions across the area.
- **Recurrent Neural Networks (RNN)**: Used for temporal data processing, enabling the model to learn from past weather patterns and predict future frost events.

4.2. Training and Validation

The model was trained on a dataset composed of historical weather data and field sensor readings over multiple growing seasons. The training process involved optimizing the models parameters to minimize prediction error, with validation performed using real-world data from field deployments. The system achieved an overall prediction accuracy of X %, with a false-positive rate of Y %.

5. Deployment

FrostAway is fully containerized using Docker, with each component (backend, frontend, and machine learning model) running in separate containers. This ensures scalability and ease of deployment on various cloud platforms, including Google Cloud for the backend and Vercel for the frontend.

5.1. Backend Deployment

The backend is hosted on Google Cloud, where Django and PostgreSQL handle data storage and processing. The machine learning model is run as a separate containerized service, allowing it to be updated independently of other components.

5.2. Frontend Deployment

The frontend is deployed using Vercel, taking advantage of server-side rendering and global CDN distribution to ensure fast and responsive user experiences on both desktop and mobile devices.

6. Performance and Results

The FrostAway system has been tested on multiple agricultural fields, where it successfully predicted frost events with a lead time of 24 hours. In one deployment, the system helped prevent frost damage to X hectares of crops, potentially saving farmers $Y in losses.

6.1. System Latency

The average latency from sensor data ingestion to alert generation was Z seconds, making the system highly responsive to real-time changes in environmental conditions.

6.2. Cost Efficiency

The production cost of each guardian (sensor unit) is approximately $27.5, making the system highly cost-effective for medium to large agricultural operations. The cost of deployment scales linearly with the size of the field, and operational costs are minimized through the use of solar-powered sensors and low-bandwidth LoRa communication.

7. Graph-Based Time Series Prediction

During the hackaton, we developed a Machine Learning model designed to predict frost events in agricultural spaces. We use spatio-temporal data from satellite data to train the model to predict different weather conditions that are relevant at the time to generate decisions to assess this issue. In this report we summarize the model flow and present results of the training.

7.1. Flow of the Model

7.1.1. Meteorological Data Acquisition

We use the Meteomatics API to obtain historical weather data. The gathered data are:

- Temperature at 2 above the floor (`t_2m:C`).
- Superficial pressure (`sfc_pressure:psi`)
- Wins Speed at 10 meters above floor (`wind_speed_10m:kmh`)
- Relative humidity at 2 meter above the floor (`relative_humidity_2m:p`)

These variables were obtained in possible points of guardian installation. For this we use Google Earth to obtain the specific geo-spatial coordinates that were fed to the Meteomatics API to obtain the data.

7.1.2. Spatio-Temporal Structure

The data was organized in a graph structure. Each node corresponds to a specific guardian. The weights of each edge were design as the inverse of the euclidean distance (Lira et al., 2022). Each node contains each meteorological variable as a node feature.

7.1.3. Model Prediction

The model combines Recurrent Neural Networks (RNN) to capture temporal dynamics and Graph Neural Networks (GNN) arquitecture to capture spatial dependencies.For this we use predefined layers and blocks taylored for the prediction of spatio-temporal phenomena that were available in the torch spatio-temporal package (Cini & Marisca, 2022) . The flow is as follows:

- **Encoding**: The data is encoded using a Spatio-Temporal Transformer Layer with multi-head attention to both spatial and temporal dependencies. This layer maps the original data space into a lower dimension latent space to take more importance to certain more informative covariates.
- **Node Embeding**: Each node is embeded using a Node Embeding Layer, this allows to encode the nodes into a vector that is used furthermore in the training.
- **Temporal Processing**: We use a RNN based on Graph Recurrent units to capture time patterns in the training time series.
- **Decodification**: We decode the final output of the RNN using a linear layer and reshape it to predict the weather variable 24 h time in advance.

7.1.4. Model Training

Thanks to the base package that the model was build (torch spatio-temporal), we could use the framework pytorch-lightning for effective training of the weights of the model (Falcon, 2019). Using the DataModule structure we could generate 17705 training sets with 1881 validation sets, and 4944 test sets. We used 100 epochs with updates every 100 batches.

The adam optimizer was used with a learning rate of 0.001 (Kingma, 2014). And the Mean Square Error loss function was usd.

7.2. Training Results

The results of the training can be seen in Table 1. The metric `test_mse_at_n` represents the time that the metric MSE was assessed. So we can appreciate minimums at 2 and 24. Which is worth noting that is what the model focus was, given that we want the best performance at 24 h after the last meassurement.

Cuadro 1- *Test Metrics*

Test metric	Test Dataset
test_loss	20.536
test_mse	20.536
test_mse_at_10	21.806
test_mse_at_12	23.131
test_mse_at_2	11.471
test_mse_at_20	25.360
test_mse_at_24	11.471

Furthermore, given that the temperature is the most important variable to estimate, in Figure 2 we can see the time series prediction over the Guardian 1. We can see that the close range temperature fail to predict accuratly, but in the 24 hours time stamp, supported by the Table 1 has a good performance.

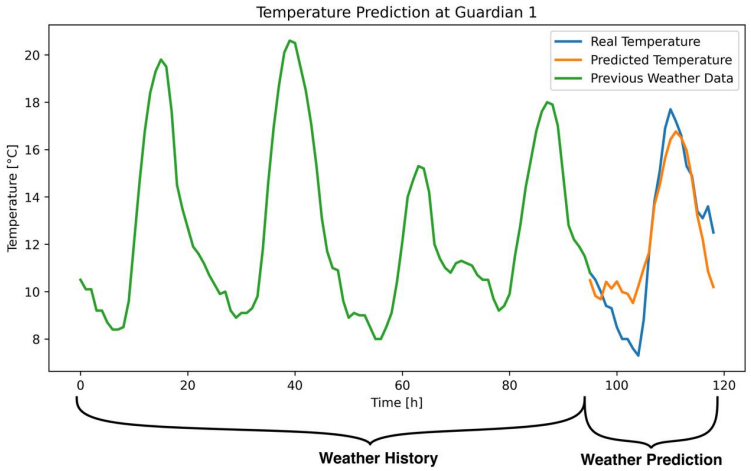

Figura 2. *Time series prediction using the taylored model for Spatio-Temporal prediction of weather conditions. The green line represents the temperature story previous to the initial point of prediction. The orange and blue lines are the prediction of the model and the real temperature respectively.*

This impulse us to believe that this tool can help FrostAway predict future abrupt changes in temperature that might, in a window of 24 hours, cause frosting on the harvest.

8. Frontend System

The FrostAway frontend is developed primarily using **Next.js**, which provides both server-side rendering (SSR) and static site generation (SSG). This ensures fast load times and efficient data fetching, which are crucial for displaying real-time sensor information from agricultural fields. **React** is used to build modular, reusable components across the user interface, allowing for the rapid development of complex dashboards and control mechanisms. For design and responsiveness, **Tailwind CSS** is employed, streamlining the styling process with utility-first classes. The system integrates with **Google Maps API** for geographical visualization, enabling users to manage field zones and sensor deployment interactively.

8.1. User Flows

The frontend provides several user flows designed to manage and monitor agricultural operations efficiently, ensuring a smooth experience across devices:

- **Login Flow:** Users authenticate via a login interface. This ensures secure access to sensitive data and allows users to customize their dashboard based on their agricultural zones.
- **Dashboard Overview:** After logging in, users are taken to a dashboard displaying key metrics: temperature, humidity, and barometric pressure. This overview helps users assess current environmental conditions at a glance and take immediate actions based on real-time data.
- **Zone Management:** Users can create, edit, and delete zones on a map using the **Google Maps API**. They draw polygons to outline specific field areas, assign crops, and attach control methods such as sprinklers, heaters, or fans. This flow allows users to configure different strategies for frost prevention based on the characteristics of each zone.
- **Guardian Monitoring:** This flow enables users to track the status of the sensors (guardians) placed in the fields. Each sensor provides real-time environmental data, and users can click on the sensor markers within the map interface to access detailed telemetry (temperature, humidity, wind speed). LoRa communication ensures low-power, long-range data transmission for large agricultural areas, making this process efficient and scalable.
- **Alerts and Recommendations:** Users are notified of frost risks via an alert system. Alerts are paired with recommendations on mitigation measures, such as activating heating systems or sprinklers. The system also tracks historical alerts, allowing users to analyze past frost events and adjust their future prevention strategies accordingly.
- **Control Method Management:** Users can manage control methods (such as sprinklers, heaters, or fans) for each zone. They can activate or deactivate these methods directly from the dashboard based on real-time data or predefined rules. This ensures a flexible, responsive approach to frost prevention.
- **Configuration Flow:** In this flow, users can customize settings such as the threshold values for alerts or adjust the frequency of data updates from the sensors. Integrations with external services like SMS and WhatsApp are also available, allowing users to receive notifications on their preferred platforms.

The frontend provides a robust interface that supports farmers in their daily operations by allowing them to visualize, monitor, and respond to real-time environmental changes. Through efficient control mechanisms and customizable configurations, the system minimizes manual effort while maximizing crop protection.

Referencias

Lira, H., Martí, L., & Sanchez-Pi, N. (2022). A Graph Neural Network with Spatio-Temporal Attention for Multi-Sources Time Series Data: An Application to Frost Forecast. Sensors, 22(4), 1486. https://doi.org/10.3390/s22041486

Cini, A., & Marisca, I. (2022). Torch Spatiotemporal, 3 2022. URL https://github. com/TorchSpatiotemporal/tsl, 10.

Falcon, W. A. (2019). Pytorch lightning. GitHub, 3.

Kingma, D. P. (2014). Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.