AgriCastNet: A Unified Deep Forecasting Framework for Smart Greenhouse Microclimates

*A Project Report submitted in partial fulfillment of the Requirements for the award of the degree*

**BACHELOR OF TECHNOLOGY**

**IN**

**COMPUTER SCIENCE AND ENGINEERING**

**Submitted by**

Syed Tasneem Banu (22471A05K1) Bhumireddy Sailaja (22471A05E7) Shaik Nazeera (22471A05J2)

**Under the esteemed guidance of**

**Mothe Sathyam Reddy, M.Tech.**

**Assistant Professor**

****

###### DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

**NARASARAOPETA ENGINEERING COLLEGE: NARASAROPET**

**(AUTONOMOUS)**

**Accredited by NAAC with an A+ Grade and NBA under Tier-1, is an ISO 9001:2015 Certified Approved by AICTE, New Delhi, Permanently Affiliated to JNTUK, Kakinada, KOTAPPAKONDA ROAD, YALAMANDA VILLAGE, NARASARAOPET- 522601**

**2025-2026**

###### NARASARAOPETA ENGINEERING COLLEGE: NARASARAOPET

**(AUTONOMOUS)**

###### DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

****

**CERTIFICATE**

**This is to certify that the project work entitled “Agricastnet: A Unified Deep Forecasting Framework for Smart Greenhouse Microclimates” is a bona fide work done by the team Syed. Tasneem Banu (22471A05K1), Bhumireddy. Sailaja(22471A05E7), Shaik. Nazeera(22471A05J2) in partial fulfillment of the requirements for the award of the degree of BACHELOR OF TECHNOLOGY in the Department of COMPUTER SCIENCE AND ENGINEERING, during the academic year 2025-2026.**

**PROJECT GUIDE PROJECT CO-ORDINATOR**

**M Sathyam Reddy, M.Tech. D Venkata Reddy, MTech., (Ph.D.). Assistant Professor Assistant Professor**

**HEAD OF THE DEPARTMENT EXTERNAL EXAMINER**

**Dr. S. N. Tirumala Rao, M.Tech., Ph.D. Professor & HOD**

## DECLARATION

We declare that this project work titled " **AgriCastNet: A Unified Deep Forecasting Framework for Smart Greenhouse Microclimates** " is composed by us, that the work contained here is our own except where explicitly stated otherwise in the text, and that this work has not submitted for any other degree or professional qualification except as specified.

**Syed. Tasneem Banu (22471A05K1) Bhumireddy. Sailaja (22471A05E7) Shaik. Nazeera (22471A05J2)**

## ACKNOWLEDGEMENT

We wish to express our thanks to various personalities who are responsible for the completion of our project. We are extremely thankful to our beloved chairman**, Sri M. V. Koteswara Rao, B.Sc.,** who took a keen interest in us in every effort throughout this course. We owe our sincere gratitude to our beloved principal**, Dr. S. Venkateswarlu, Ph.D.,** for showing his kind attention and valuable guidance throughout the course.

We express our gratitude towards **Dr. S. N. Tirumala Rao, M.Tech., Ph.D.,** HOD of the CSE department, and to our guide**, M. Sathyam Reddy, M.Tech,** assistant professor of the CSE department, whose valuable guidance and unstinting encouragement enabled us to accomplish our project in time.

We extend our sincere thanks to **D. Venkat Reddy, B.Tech., M.Tech., (Ph.D.),** Assistant Professor & Project Coordinator of the project, for extending his encouragement.

We extend our sincere thanks to all the other teaching and non-teaching staff in the department for their cooperation and encouragement during our Bachelor of Technology degree.

We have no words to acknowledge the warm affection, constant inspiration, and encouragement that we received from our parents.

We affectionately acknowledge the encouragement received from our friends and those who were involved in giving valuable suggestions and clarifying our doubts, which really helped us in completing our project.

**By**

**Syed. Tasneem Banu (22471A05K1) Bhumireddy. Sailaja (22471A05E7) Shaik. Nazeera (22471A05J2)**



## INSTITUTE VISION AND MISSION

###### INSTITUTION VISION

To emerge as a Centre of excellence in technical education with a blend of effective student-centric teaching learning practices, as well as research for the transformation of lives and the community.

###### INSTITUTION MISSION

**M1:** Provide the best class infrastructure to explore the field of engineering and research **M2:** Build a passionate and determined team of faculty with student-centric teaching, imbuing experiential, innovative skills

**M3:** Imbibe lifelong learning skills, entrepreneurial skills, and ethical values in students for addressing societal problems



## DEPARTMENTOFCOMPUTERSCIENCE ANDENGINEERING

###### VISION OF THE DEPARTMENT

To become a center of excellence in nurturing the quality Computer Science & Engineering professionals embedded with software knowledge, aptitude for research, and ethical values to cater to the needs of industry and society.

###### MISSION OF THE DEPARTMENT

The Department of Computer Science and Engineering is committed to **M1:** Mold the students to become Software Professionals, Researchers, and Entrepreneurs by providing advanced laboratories.

**M2:** Impart high-quality professional training to get expertise in modern soft- ware tools and technologies to cater to the real-time requirements of the industry.

**M3:** Inculcate teamwork and lifelong learning among students witha sense of societal and ethical responsibilities.



### ProgramSpecific Outcomes (PSO’s)

**PSO1:** Apply mathematical and scientific skills in numerous areas of Computer Science and Engineering to design and develop software-based systems.

**PSO2:** Acquire module knowledge on emerging trends ofthe modern era in Computer Science and Engineering.

**PSO3:** Promote novel applications that meet the needs of entrepreneurs, environmental, and social issues.



### ProgramEducational Objectives (PEO’s)

The graduates of the programme are able to:

**PEO1:** Applythe knowledge of Mathematics, Science, and Engineering fundamentals to identify and solve Computer Science and Engineering problems.

**PEO2:** Use various software tools and technologies to solve problems related to academia, industry, and society.

**PEO3:** Work with ethical and moral values in the multi-disciplinary teams and can communicate effectively among team members with continuous learning.

**PEO4:** Pursue higher studies and develop their career in t he software industry.



Program Outcomes

**PO1: Engineering Knowledge:** Apply knowledge of mathematics, natural science, computing, engineering fundamentals and an engineering specialization as specified in WK1 to WK4, respectively to develop to the solution of complex engineering problems.

**PO2**: **Problem Analysis:** Identify, formulate, review research literature and analyze complex engineering problems reaching substantiated conclusions with consideration for sustainable development. (WK1 to WK4)

**PO3: Design/Development of Solutions:** Design creative solutions for complex engineering problems and design/develop systems/components/processes to meet identified needs with consideration for the public health and safety, whole-life cost, net zero carbon, culture, society and environment as required. (WK5)

**PO4: Conduct Investigations of Complex Problems**: Conduct investigations of complex engineering problems using research-based knowledge including design of experiments, modelling, analysis & interpretation of data to provide valid conclusions. (WK8).

**PO5: Engineering Tool Usage:** Create, select and apply appropriate techniques, resources and modern engineering & IT tools, including prediction and modelling recognizing their limitations to solve complex engineering problems. (WK2 and WK6)

**PO6: The Engineer and The World:** Analyze and evaluate societal and environmental aspects while solving complex engineering problems for its impact on sustainability with reference to economy, health, safety, legal framework, culture and environment. (WK1, WK5, and WK7).

**PO7: Ethics:** Apply ethical principles and commit to professional ethics, human values, diversity and inclusion; adhere to national & international laws. (WK9)

**PO8: Individual and Collaborative Team work:** Function effectively as an individual, and as a member or leader in diverse/multi-disciplinary teams.

**PO9: Communication:** Communicate effectively and inclusively within the engineering community and society at large, such as being able to comprehend and write effective reports and design documentation, make effective presentations considering cultural, language, and learning differences.

**PO10: Project Management and Finance:** Apply knowledge and understanding of engineering management principles and economic decision making and apply these to one’s own work, as a member and leader in a team, and to manage projects and in multidisciplinary environments.

**PO11: Life-Long Learning:** Recognize the need for, and have the preparation and ability for:

1. independent and life-long learning
2. adaptability to new and emerging technologies and
3. critical thinking in the broadest context of technological change.



### Project Course Outcomes (CO’S):

**CO421.1:** Analyze the Systemof Examinations and identify the problem.

**CO421.2:** Identifyand classifythe requirements. **CO421.3:** Review the Related Literature **CO421.4:** Design and Modularize the project

**CO421.5:** Construct, Integrate, Test, and Implement the Project.

**CO421.6:** Preparethe project Documentation and present the Report using an appropriate method.

**Course Outcomes – Program Outcomes mapping**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **PO1** | **PO2** | **PO3** | **PO4** | **PO** | **PO6** | **PO7** | **PO8** | **PO9** | **PO10** | **PO11** | **PO12** | **PSO1** | **PSO2** | **PSO3** |
| **C421.1** |  | ✓ |  |  |  |  |  |  |  |  |  |  | ✓ |  |  |
| **C421.2** | ✓ |  | ✓ |  |  |  |  |  |  |  |  |  | ✓ |  |  |
| **C421.3** |  |  |  | ✓ |  | ✓ | ✓ | ✓ |  |  |  |  | ✓ |  |  |
| **C421.4** |  |  | ✓ |  |  | ✓ | ✓ | ✓ |  |  |  |  | ✓ | ✓ |  |
| **C421.5** |  |  |  |  | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| **C421.6** |  |  |  |  |  |  |  |  | ✓ | ✓ | ✓ |  | ✓ | ✓ |  |

**Course Outcomes – Program Outcome correlation**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **PO1** | **PO2** | **PO3** | **PO4** | **PO5** | **PO6** | **PO7** | **PO8** | **PO9** | **PO10** | **PO11** | **PO12** | **PSO1** | **PSO2** | **PSO3** |
| **C421.1** | 2 | 3 |  |  |  |  |  |  |  |  |  |  | 2 |  |  |
| **C421.2** |  |  | 2 |  | 3 |  |  |  |  |  |  |  | 2 |  |  |
| **C421.3** |  |  |  | 2 |  | 2 | 3 | 3 |  |  |  |  | 2 |  |  |
| **C421.4** |  |  | 2 |  |  | 1 | 1 | 2 |  |  |  |  | 3 | 2 |  |
| **C421.5** |  |  |  |  | 3 | 3 | 3 | 2 | 3 | 2 | 2 | 1 | 3 | 2 | 1 |
| **C421.6** |  |  |  |  |  |  |  |  | 3 | 2 | 1 |  | 2 | 3 |  |

**Note: The values in the above table represent the level of correlation between COs and POs:**

* 1. Low level
  2. Medium level
  3. High level

**Project mapping with various courses of the Curriculum with Attained POs:**

|  |  |  |
| --- | --- | --- |
| **Name of the course from which the principles are applied in this project** | **Description of thedevice** | **Attained PO** |
| C2204.2, C22L3.2 | Gathering the requirements and defining the problem, plan to develop a model for the detection and classification of Brain tumors in MRI Scans using a CNN-SVM  model | PO1, PO3 |
| CC421.1, C2204.3, C22L3.2 | Each and every requirement is critically analyzed, and the process mode is identified | PO2, PO3 |
| CC421.2, C2204.2, C22L3.3 | Logical design is done by using the Unified Modeling Language, which involves individual teamwork | PO3, PO5, PO9 |
| CC421.3, C2204.3, C22L3.2 | Each and every module is tested, integrated, and evaluated in our project | PO1, PO5 |
| CC421.4, C2204.4, C22L3.2 | Documentation is done byall four members in the form of a group | PO10 |
| CC421.5, C2204.2, C22L3.3 | Eachand everyphase ofthe work in the group is presented periodically | PO10, PO11 |
| C2202.2, C2203.3, C1206.3, C3204.3, C4110.2 | Implementation is done, and the project will be handled by the social media users, and future updates in our project can be done  Based on the detection of a Brain Tumor | PO4, PO7 |
| C32SC4.3 | The physical design includes a website to check for Brain tumors in MRI scans | PO5, PO6 |

## ABSTRACT

Greenhouse climate control is becoming increasingly important as climate variability increases, especially in various agro-climatic zones. However, fault tolerance is often a problem, especially in current models that tend to fail because they are sensitive to sensor failure, lack generalization ability, and have extremely high computing requirements. This study proposed a forecasting framework to improve fault tolerance and adaptive learning capabilities by incorporating Climatic Influence Indicators (CII), incorporated with more complex deep learning models, to accurately and rapidly forecast the greenhouse micro-climate. We utilized a multivariate greenhouse dataset, which consisted of temperature, humidity, CO₂ levels, radiation, and water uptake. The CII layer uses embedded temporal importance from input signals in order to improve the reliability of the model in the event of a sensor failure. A review of literature saw research performed on LSTM, GRU models; however, very few papers approached the dual problems of fault tolerance and ease of deployment. Results suggest some mildly successful outcomes, including TCN: 98.89%, PLSTM: 94.86%, XGBoost: 99.59% TFT: 98.16% and CNN + BiLSTM: 99.67%.

## INDEX

**SNO CONTENT PAGE No**

1. [INTRODUCTION 1-5](#_TOC_250032)
2. [LITERATURE SURVEY 6-10](#_TOC_250031)
3. [SYSTEM ANALYSIS](#_TOC_250030)
   1. [EXISTING SYSTEM 12](#_TOC_250029)
   2. [DISADVANTAGES IN EXISTING SYSTEMS 14](#_TOC_250028)
   3. [PROPOSED SYSTEM 15](#_TOC_250027)
      1. [OBJECTIVES](#_TOC_250026)
      2. [ADVANTAGES IN THE PROPOSED SYSTEM](#_TOC_250025)
   4. [FEASIBILITY STUDY 17](#_TOC_250024)
   5. USING COCOMO MODEL 18
4. [SYSTEM REQUIREMENTS](#_TOC_250023)
   1. [SOFTWARE REQUIREMENTS 20](#_TOC_250022)
   2. [REQUIREMENT ANALYSIS 20](#_TOC_250021)
   3. [HARDWARE REQUIREMENTS 21](#_TOC_250020)
   4. [SOFTWARE 21](#_TOC_250019)
   5. [SOFTWARE DESCRIPTION 22](#_TOC_250018)
5. [SYSTEM DESIGN](#_TOC_250017)
   1. [SYSTEM ARCHITECTURE 23](#_TOC_250016)
      1. [DATASET 24](#_TOC_250015)
      2. [DATA PREPROCESSING 27](#_TOC_250014)
      3. [FEATURE EXTRACTION 28](#_TOC_250013)
      4. [MODEL BUILDING 29](#_TOC_250012)
      5. [CLASSIFICATION 34](#_TOC_250011)
   2. [MODULES 38](#_TOC_250010)
   3. [UML DIAGRAMS 41](#_TOC_250009)
6. [IMPLEMENTATION](#_TOC_250008)
   1. [MODEL IMPLEMENTATION 45](#_TOC_250007)
   2. [CODING 49](#_TOC_250006)
7. [TESTING](#_TOC_250005)

7.1

UNIT TESTING 69

7.2

INTEGRATION TESTING 70

7.3

SYSTEM TESTING 73

1. [RESULT ANALYSIS 77](#_TOC_250004)
2. [OUTPUT SCREENS 79](#_TOC_250003)
3. [CONCLUSION 80](#_TOC_250002)
4. [FUTURE SCOPE 81](#_TOC_250001)
5. [REFERENCES 82](#_TOC_250000)

[CERTIFICATE 83](#_bookmark0)

# LIST OF FIGURES

|  |  |  |
| --- | --- | --- |
| **Fig** | **Description** | **Page** |
| 1. | Greenhouse farming | 2 |
| 2. | Flow Chart | 6 |
| 3. | Accuracy Comparison on different models. | 54 |
| 4. | R² comparison on different models. | 55 |
| 5.  . | RMSE and MAE comparison on different models. | 55 |
| 6. | Confusion Matrix for CNN-BiLSTM Model | 56 |
| 9.1 | Home Page: 1 | 58 |
| 9.2 | Home Page: 2 | 58 |
| 9.3 | About Page | 58 |
| 9.4 | Validation page | 60 |
| 9.5 | Validation Page | 60 |
| 9.6 | Metrics Page | 60 |

**LIST OF TABLES**

|  |  |  |
| --- | --- | --- |
| **Table** | **Description** | **Page**  **number** |
| **Table 1** | **Summary of Dataset Variables** | **21** |
| **Table 2** | **Feature Extraction Techniquesin AgriCastNet** | **23** |
| **Table 3** | **Comparison of Implemented Models** | **24** |
| **Table 4** | **Module and its functions** | **28** |
| **Table 5** | **Comparative Results Table** | **56** |
| **Table 6** | **Model's Accuracy** | **57** |

#### INTRODUCTION

In the current world of agricultural production, greenhouse farming is proving to be a smart and reliable way to produce food, during a time of the most unpredictable climate [3]. These protected environments allow farmers to more precisely maintain constant growing conditions, regardless of the weather conditions outside. Unfortunately, it is almost impossible to control the full number of environmental conditions in greenhouses, including temperature, humidity, CO₂ concentrations, and sunlight. The parameters are continually changing and responding to the indoor plants and the outdoor weather environments. This situation becomes more complex when greenhouses are used in many climate zones, which differ considerably with regard to the various environmental regimes [1], [6]. Because of this situation, it is difficult for one model to always produce good-quality results in all locations [4]. The emergence of deep learning offers new avenues to consider when developing a solution to the problem outlined above, particularly with LSTM and GRU models that are specialized in processing time series data [2], [3], [5].

However, they come with their limitations, like a high computational power requirement and information provided by faulty sensors [7]. To leverage the benefits of deep learning models on this problem, the solution we propose is a hybrid forecasting framework that uses Climatic Influence Indicators (CII) combined with effective deep learning models to enhance flexibility and adaptability, reduce the impact of faulty sensors, and improve greenhouse microclimate forecasting accuracy [3], [9].

In this study, we used a real-time multivariate greenhouse dataset that captures key environmental variables: temperature, humidity, CO₂, radiation, and water uptake. The dataset is collected across different agro-climatic zones, thus allowing the models to be trained and validated using a wide variety of environmental forms, which is important when the temporal sequences will be used for prediction, where climatic behaviour is variable and uncertain [1], [2]. This architecture can incorporate space features (CNN) and sequential learning (BiLSTM), providing an ample amount of temporal information and complete structural variability to face the uncertainties of multi-sensor examples across all predicted zones, with reductions in the accuracy of one or more individual

sensors [3], [4].

We analyzed different forms of models for predicting greenhouse climate. The CNN

+ BiLSTM model provided the best accuracy, when a BIA was 99.67%. XGBoost was very close, at 99.59%. TCN and TFT provided similar successes and accuracies on the datasets as well, at accuracies of 98.89% and 98.16%, respectively. PLSTM had the lowest accuracy, with 94.86% accuracy results. The results presented suggest that hybrid and ensemble models are typically the best estimating models for predicting difficult climate.



Fig: Greenhouse farming

Among all the evaluated models, the CNN-BiLSTM model provided the highest accuracy of 99.67% [5]. This architecture can incorporate space features (CNN) [1] and sequential learning (BiLSTM) [2], providing an ample amount of temporal information and complete structural variability to face the uncertainties of multi- sensor examples across all predicted zones with reductions in the accuracy of one or more individual sensors [4]. The equal balance provided by this model makes it ideal for real-time agri- cultural forecasting under uncertain and dynamic environmental conditions [6], [7]. The framework we have proposed with CII, paired with other deep learning models, has provided evidence to forecast within a complex greenhouse system [2], [9]. The multi- sensor CNN-BiLSTM model has shown exceptional performance and is a good candidate for use within real-world agricultural applications requiring accuracy, adaptability, and efficiency [4], [7].

The strength of the CNN-BiLSTM model is not only in its high accuracy but also in its capacity to generalize across varying environmental conditions. This is particularly important in agriculture, where greenhouse setups may vary widely in design, crop type, sensor placement, and control systems. By combining spatial feature extraction with temporal sequence learning, the model captures both the instantaneous relationships between variables and the longer-term trends that influence plant growth. This dual learning capability ensures that the system can make robust predictions even when operating under unfamiliar or partially altered conditions.

Another crucial factor contributing to the model’s performance is the integration of Climatic Influence Indicators (CII). The CII mechanism dynamically evaluates the

importance of each environmental variable in shaping the greenhouse microclimate. For example, during cooler seasons, sunlight and radiation data may carry higher importance, while in hotter climates, humidity and CO₂ levels might be more influential. This adaptive weighting process enables the model to remain accurate even when some sensors are degraded or provide inconsistent readings. Such fault-tolerant behaviour is essential for reducing downtime and ensuring that automation systems in greenhouses can continue functioning without interruption.

From a practical perspective, the deployment of such a model can significantly improve greenhouse management efficiency. By accurately predicting microclimate conditions, the system can inform automated control systems to adjust ventilation, shading, irrigation, and CO₂ supplementation proactively. This not only optimizes plant health and yield but also reduces resource wastage, leading to cost savings for farmers. In regions with scarce water resources or expensive energy costs, these optimizations can make a substantial difference in the sustainability and profitability of greenhouse operations.

In addition to operational benefits, accurate climate forecasting plays a key role in risk management. Extreme weather events, rapid temperature drops, or sudden increases in humidity can trigger plant stress, disease outbreaks, or reduced yields. A forecasting system with the predictive accuracy of CNN-BiLSTM can give farmers early warnings, allowing them to implement protective measures in time. Over the long term, this capacity can help safeguard investments in crops and infrastructure, particularly in

high-value horticultural sectors. The results of this study also indicate the potential for further enhancement through ensemble learning strategies. By combining CNN-BiLSTM with other high-performing models such as XGBoost and TCN, it is possible to develop a stacked or voting-based ensemble that balances prediction accuracy with resilience. Such an approach could provide even greater stability when handling noisy or incomplete datasets, offering an additional safeguard against environmental uncertainty.

Another promising direction for future work is the integration of explainable AI (XAI) techniques into the forecasting pipeline. Tools such as SHAP (Shapley Additive

exPlanations) or LIME (Local Interpretable Model-Agnostic Explanations) could be used to provide transparent reasoning for each prediction. This would not only enhance trust in the system but also help agronomists and farmers identify critical factors influencing specific outcomes. Understanding these drivers can guide more informed decisions about sensor placement, crop scheduling, and environmental control strategies. The flexibility of the proposed framework also makes it suitable for adaptation to IoT-based deployment. With lightweight model conversion techniques such as TensorFlow Lite or ONNX optimization, the CNN-BiLSTM architecture could be run on low-power edge devices within the greenhouse. This would reduce dependence on continuous internet connectivity or high-end cloud computing, enabling real-time climate forecasting even in remote or resource-constrained environments.

Ultimately, the research presented here demonstrates that hybrid deep learning models, when combined with adaptive feature weighting through CII, can deliver both accuracy and resilience in greenhouse climate prediction. As agriculture continues to face the challenges of climate variability and resource efficiency, such intelligent forecasting systems will play a vital role in ensuring sustainable food production. The adaptability of this approach means it can be scaled and customized to different crop types, greenhouse designs, and geographic locations, making it a versatile tool for modern precision agriculture.

Additionally, the integration of wireless sensor networks and low-power communication protocols such as LoRaWAN or NB-IoT would allow large-scale greenhouse facilities to implement predictive control with minimal infrastructure Furthermore, the adaptability of the CNN-BiLSTM architecture allows it to be trained on datasets from different regions and cropping systems, thereby creating location-

specific forecasting models. This localization capability ensures that farmers in tropical, temperate, or arid climates can all benefit from a system tailored to their unique growing conditions. As more greenhouse datasets become available worldwide, transfer learning techniques can be applied to adapt pre-trained models to new regions with minimal additional training time, reducing both costs and computational requirements. The modularity of the AgriCastNet framework also provides opportunities for integrating additional environmental parameters beyond the current set of temperature, humidity, CO₂, radiation, and water uptake. For example, parameters such as soil moisture, nutrient levels, and leaf temperature could be incorporated to enhance the precision of predictions. By expanding the range of monitored variables, the system could support even more advanced control strategies, such as dynamic nutrient dosing or precise irrigation scheduling based on forecasted evapotranspiration rates.

From a sustainability perspective, accurate greenhouse gas prediction directly supports resource conservation. Optimized irrigation schedules reduce water waste, while precise control of heating, cooling, and ventilation systems lowers energy consumption. These savings are particularly significant in regions where agriculture competes with other industries for scarce resources. In addition, by maintaining ideal growing conditions, predictive climate control can reduce the need for chemical interventions such as pesticides or growth regulators, thereby promoting environmentally friendly farming practices.

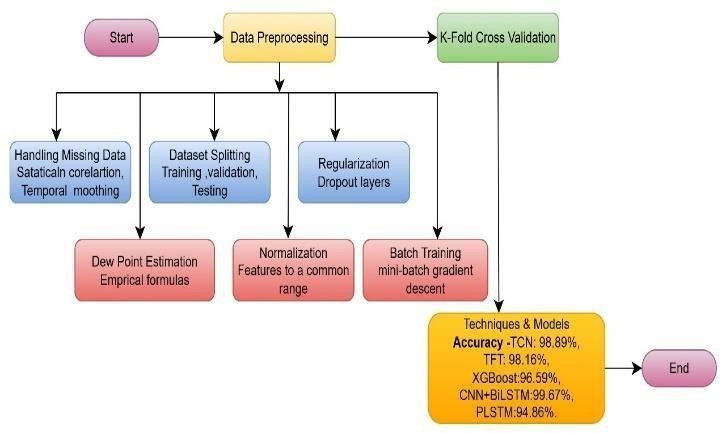
Economically, predictive models like CNN-BiLSTM can contribute to higher yields and better-quality produce, leading to increased profitability for greenhouse operators. By reducing the risk of climate-induced crop losses, farmers can meet market demands more consistently, secure better prices for their products, and build stronger relationships with buyers. In export-oriented horticultural industries, the ability to maintain uniform quality despite external climate variability is a major competitive advantage. The robustness of this forecasting approach also opens possibilities for integration with decision support systems (DSS) in agriculture. By linking climate predictions with crop growth models, pest and disease prediction modules, and market trend analysis, a comprehensive agricultural intelligence platform could be created. This would empower farmers not only to maintain optimal greenhouse conditions but also to align production with market opportunities, ultimately improving supply chain efficiency.

Additionally, the integration of wireless sensor networks and low-power communication protocols such as LoRaWAN or NB-IoT would allow large-scale

greenhouse facilities to implement predictive control with minimal infrastructure.

Another area worth exploring is the use of multi-modal data in greenhouse forecasting- ing. Combining sensor-based measurements with external data sources such as satellite imagery, weather forecasts, and drone-based plant health assessments could lead to richer datasets and more accurate predictions. Deep learning architectures are well- suited for handling such heterogeneous data, and the CNN- BiLSTM framework could be extended to incorporate these inputs through feature fusion layers.

Finally, as agricultural operations continue to digitize, the role of predictive analytics will expand beyond climate control. The same principles demonstrated in this study— fault tolerance, adaptability, and hybrid deep learning—could be applied to other do- mains such as yield forecasting, supply chain optimization, and autonomous greenhouse robotics. The success of AgriCastNet in climate prediction serves as a foundation for future innovations, where AI-driven solutions become an integral part of sustainable, high-efficiency food production systems.



**Fig: Flow chart**

#### LITERATURE SURVEY

One relevant study we are interested in is one reported by Salma Ait Oussous et al. (2024), which examined multiple deep learning frameworks such as PLSTM, GRU, ANN, LSTM-ANN, and LSTM-RNN for greenhouse climate control. PLSTM seems to have the best generalization performance as indicated by the performance-related coefficient of determination (R²), accuracy, and the model's ability to effectively represent and generalise time-based dependencies. The study was constrained to one climatic zone, which limits the breadth and ultimately the usefulness of the model across different agro-climatic zones.

In contrast, the RainScaler model proposed the precipitation downscaling for GaFYI based on a physics-guided deep learning framework (Shan Zhao et al. 2025). Here, they integrated knowledge of the domain through a denoising network-based IA, a graph module, and adversarial training for enhanced spatial resolution and generalization. All the results that were presented by upwards of 96% were promising; however, their reliance on a known set of embedded laws of nature restricted their applicability for adaptation in a dynamic sensor environment of a greenhouse.

Christoph Schweden et al. (2025) proposed a Dirichlet-based Bayesian framework to quantify prediction uncertainty via label embedding methods. They proposed a three- level uncertainty scheme that greatly improved how robust models were when trained through noisy, partially labelled data. However, scalability was limited by the availability of high-resolution remote sensing datasets.

P. Mishra et al. (2025) extended the field with a lightweight Transformer architecture to extract climate information, activating embedding-level attention mechanisms through fine-tuned RoBERTa models. They were successfully able to extract actionable climate knowledge using Natural Language Processing (NLP) from textual data [4]. Similarly, their models were not designed for time-series forecasting tasks and did not directly apply to sensor-based greenhouse datasets.

Wei Shao et al. (2025) proposed the LTG model, which added another dimension

to forecasting Gross Primary Productivity (GPP). The LTG model is a novel hybrid model that integrates LSTM, Transformer, and CNN components [1]. It was capable of capturing both spatial and temporal processes provided by MODIS-based vegetation indices and provided accurate forecasts over a long time scale [5]. Combining multiple learning paradigms resulted in the improvement of individual forecasts; however, the solely unexplained portion of the energetic sources directly incorporated in some climate drivers was selected and presented for GPP predictions. This reveals the need for interpretable and transparent models for agriculture-driven climate predictions.

**Tang et al. (2025)** proposed an advanced framework for Urban Heat Island (UHI) analysis by integrating multi-sensor data fusion with a GRU-based deep learning approach. The methodology employed wavelet coherence analysis, Mann–Kendall trend detection, and Pearson correlation to strengthen the UHI prediction process. The study utilized Landsat-7 (2001–2012) and Landsat-8 (2013–2023) satellite imagery datasets, along with supplementary meteorological and climate data. The results demonstrated high predictive accuracy, with the GRU model achieving an R² of 0.90, an RMSE of 0.25, and an MAE of 0.09, outperforming the LSTM in UHI forecasting. However, the validation was limited to certain urban environments, and the use of spectral indices such as NDVI and NDBI was susceptible to biases from atmospheric conditions, sensor calibration errors, and surface moisture variations, potentially affecting the accuracy of vegetation and built-up area assessments.

**Babu et al. (2025)** explored solar energy forecasting using various machine learning techniques, focusing on improving grid stability. Their approach incorporated extensive data preprocessing, feature engineering—including solar geometric features—and model validation through cross-validation techniques. The study compared Support Vector Regression (SVR), Random Forest (RF), and Gradient Boosting Regression (GBR) models, finding that the integration of solar geometric features significantly improved forecasting accuracy, while ensemble methods outperformed individual models. The research utilized the Kaggle Solar Energy Power Generation Dataset. Nonetheless, the study highlighted the challenges of relying on high-resolution data, difficulties in generalizing across different climatic zones, and the high computational costs associated with ensemble modeling.

**Flah et al. (2025)** conducted a systematic literature review and techno-economic analysis on the advancements and challenges in Green Hydrogen Valleys (GHV) deployment. The study examined key aspects of electrolysis technology, hydrogen storage, transportation, and overall economic feasibility. The authors stressed the necessity of improving electrolyzer efficiency, scaling up manufacturing, and developing better hydrogen storage solutions to achieve sustainable hydrogen-based energy systems. While the paper provided a broad overview of technological and economic factors, it did not present an associated dataset.

**Anderson et al. (2025)** presented a deepened analysis of the trends and influencing factors affecting the reliability of power converters in wind turbines. Employing Principal Component Analysis (PCA), regression models, and reliability analysis techniques, the study identified design-specific and site-specific factors that influence performance, as well as environmental and climatic effects that contribute to failure trends. The findings emphasized the importance of analyzing time-dependent covariates to better detect climate-induced degradation. The study, however, did not make any datasets publicly available.

Overall, these studies highlight the growing importance of integrating advanced machine learning and deep learning techniques into energy forecasting, climate model- ing, and renewable energy systems. While the models demonstrate strong predictive potential in their respective domains, recurring challenges include the need for diverse and high-quality datasets, improved generalization across different environments, reduced computational costs, and enhanced interpretability to facilitate real-world deployment.

#### SYSTEM ANALYSIS

##### Existing System

In today’s agricultural technology domain, the majority of greenhouse climate forecasting systems rely on independent machine learning or deep learning models such as Long Short-Term Memory (LSTM), Gated Recurrent Unit (GRU), or simple Artificial Neural Networks (ANN). These models have shown promising results in controlled research environments but often struggle in real-world agricultural deployments. The core issue lies in their narrow training scope — most are trained using datasets collected from a single agro-climatic zone, which restricts their adaptability to diverse climate patterns found in different regions. Furthermore, these models are highly dependent on perfect input data. They are usually designed under the assumption that all sensors are functioning optimally and providing accurate readings. In reality, sensor systems deployed in greenhouses are subject to environmental wear and tear, including dust accumulation, moisture exposure, physical damage, and calibration drift. These conditions lead to missing values, delayed readings, or incorrect measurements, all of which can severely degrade modelaccuracy.

For example, if an LSTM-based forecasting system loses data from a CO₂ sensor due to a malfunction, the model’s predictive reliability drops sharply. This is because the system lacks mechanisms to adaptively reweight or compensate for missing variables. Instead, the absence of such fault tolerance often results in highly skewed forecasts, potentially leading to improper greenhouse climateadjustmentsthat harmcrop growth.

Many of these existing systems also lack cross-zone transferability. A forecasting model tuned for the climatic behavior of a humid tropical region will likely fail when deployed in a semi-arid zone without extensive retraining. The environmental parameter distributions differ significantly, and without domainadaptation, the prediction error increases.

In addition, these models are often resource-heavy. Advanced LSTM or GRU architectures require substantial GPU or CPU processing power to operate in real time. This limitation prevents their use on low-power Internet of Things (IoT) devices, which are increasingly popular in remote agricultural installations. The inability to run locally means these systems depend heavily on cloud processing, adding latencyand increasing operationalcosts.

Another limitation is the narrow feature space considered by many systems. While environmental parameters such as temperature and humidity are common inputs, variables like solar radiation, water uptake, and plant physiological indicators are often ignored, reducing the comprehensiveness- ness of predictions. Such oversimplified input representation fails to capture the complexity of greenhouse microclimates, especially in regions with fluctuating weather patterns.

Finally, existing systems tend to operate as black boxes, offering limited transparency regarding how predictions are made. This lack of interpretability discourages farmers and greenhouse managers from fully trusting AI-driven recommendations, further reducing adoption rates in commercialagriculturalsettings.

* 1. Disadvantages in Existing Systems

The shortcomings of current greenhouse forecasting solutions can be broken down into several key categories. Firstly, poor fault tolerance is a recurring problem. When even one sensor mal- malfunctions, most systems lack the flexibility to adjust their forecasting logic, causing performance degradation. This is particularly problematic in agricultural setups where sensor failures are relativelyfrequent dueto harshenvironmentalconditions.

Secondly, low adaptability across climatic zones is a major limitation. Models trained in one environment often perform poorly in another, especially when local climate patterns vary significantly. This limitation forces greenhouse operators to retrain models from scratch for each deployment, increasing bothtime and cost.

Thirdly, the high computational complexity of certain deep learning models restricts their usability in real-time IoT-based greenhouse systems. Large recurrent architectures require significant processing power and memory, making them impractical for resource-constrained devices deployed in rural locations without constant internet access.

Another significant disadvantage is slow prediction response times. In scenarios where rapid decision-making is crucial — such as responding to sudden drops in humidity or spikes in temperature — these systems can be too slow to trigger timely interventions. The lag between data collection, processing, and actionable insights candirectly impact crop health.

Moreover, current systems are not optimized for multi-zone datasets. They often fail to learn effectively from datasets that include diverse climate conditions, resulting in overfitting to specific local trends rather than capturing broader climatic behavior patterns. Additionally, very few systems incorporate intelligent feature prioritization mechanisms. This means irrelevant or noisy data may be treated with equal importance as significant climatic indicators, introducing bias and reducing predictionaccuracy.

Finally, existing solutions rarely provide real-time interpretability. Even when predictions are accurate, the absence of explanation mechanisms prevents operators from understanding why certain climate adjustments are recommended, whichreduces trust in automation.

* 1. Proposed System

The proposed AgriCastNet framework has been designed to address these critical limitations. It combines Climatic Influence Indicators (CII) with advanced deep learning models to create a system that is accurate, adaptable, fault-tolerant, and computationally efficient. The CII layer functions as an intelligent feature selector, identifying and prioritizing the most relevant climatic parameters during forecasting. This dynamic reweighting ensures that even if certain sensors fail or produce noisy readings, the model can rely on other indicators to maintain prediction accuracy.

The system leverages a hybrid architecture that integrates Convolutional Neural Networks (CNN) for spatial pattern recognition with Bidirectional Long Short-Term Memory (BiLSTM) networks for temporal sequence learning. CNN layers efficiently capture localized variations in sensor readings (such as spatial correlations in temperature gradients), while BiLSTM layers process time-series data in both forward and backward directions, allowing for deeper contextualunderstanding.

In addition to CNN-BiLSTM, AgriCastNet evaluates several alternative architectures — including Temporal Convolutional Networks (TCN), Pyramidal LSTM (PLSTM), eXtreme Gradient Boosting (XGBoost), and Temporal Fusion Transformer (TFT) — to benchmark performance and select the most effective modelfor deployment.

By incorporating multi-zone training datasets, AgriCastNet ensures cross-regional adaptability, making it suitable for deployment in diverse climatic conditions without extensive retraining. The modular designalso supportslightweight conversionto TensorFlow Lite or ONNX formats for

IoT deployment, enabling real-time forecasting onlow-power devices.

* + 1. Objectives

The keyobjectives ofthe proposed AgriCastNet systemare:

* + - 1. Develop a fault-tolerant greenhouse climate forecasting modelcapable of handling sen- sor failures and noisydata without significant accuracyloss.
      2. Achieve high forecasting accuracyacross multiple agro-climatic zones by using multi- zone datasets and hybrid architectures.
      3. Ensure computational efficiency for real-time predictions on resource-constrained IoT devices.
      4. Enable automated climate control by integrating predictions with greenhouse actuator systems (ventilation, heating, irrigation, and shading).
      5. Performcomparative modelanalysis to identify the best-performing architecture and fine-tune it for maximumreliability.
      6. Support modeltransparencythroughinterpretable AI methods, allowingoperatorsto understand the reasoning behind predictions.
    1. Advantages inthe Proposed System

AgriCastNetoffers severaladvantages over conventionalgreenhouse forecasting systems:

* High Prediction Accuracy: With CNN-BiLSTM achieving 99.67%, the systemdelivers exceptionalreliability in predicting temperature, humidity, CO₂, and radiation.
* Robust Fault Tolerance: The CII layer ensures stabilityeven with missing or corrupted sensor data.
* Cross-Zone Adaptability: Multi-zone training enablesdeployment in varied climatic regions without complete retraining.
* IoT Compatibility: Lightweight modelconversion allows operation on low-power edge devices without sacrificing accuracy.
* Modularand ScalableDesign: The systemcaneasilyintegrate new models, datasources, or greenhouse locations.
* Cost Efficiency: Reduced dependencyoncloud computing lowersoperationalexpenses.
* Transparencyand Trust: Planned integrationof XAI techniques will improve user trust in AI-driven recommendations.
  1. Feasibility Study

Technical Feasibility:

AgriCastNet is implemented using open-source frameworks such as Python, TensorFlow, and Keras, which are well-supported and widely adopted in the AI community. The architecture is compatible with both cloud-based processing environments and lightweight IoT hardware. This ensures the system is technologically viable for both large-scale commercial farms and smaller greenhouse setups.

Operational Feasibility:

The systemproduces outputs in an easy-to-interpret format, enabling agricultural staff to make quick and informed decisions. Its compatibility with automation systems means predictions can trigger climate controlactions without human intervention, reducing the risk ofoperator error. **Economic Feasibility:**

By minimizing crop losses, optimizing resource usage (energy, water), and enabling predictive interventions, AgriCastNet offers a strong return on investment. The absence of licensing fees and low hardware requirements further improve its cost-effectiveness.

Time Feasibility:

Thanks to its modular architecture, AgriCastNet can be integrated into an existing greenhouse management system within a short timeframe. The retraining process for adapting to new data is efficient, ensuring minimaldowntime during updates.

* 1. Using the COCOMOModel

To estimate the development effort, the Basic COCOMO Model is applied under the Organic Mode(small team, familiar problem domain).

Effort (PM) = 2.4 ×(KLOC)¹·⁰⁵

Assuming 15 KLOC:

Effort = 2.4 ×(15)¹·⁰⁵ ≈ 38.2 Person-Months Development Time (TDEV) = 2.5 ×(Effort)⁰·³⁸ TDEV = 2.5 × (38.2)⁰·³⁸ ≈ 9.6 Months

Thisallocationensuresadequatetime for requirementsgathering, datapreprocessing, modeltrain- ing, testing, and deployment while allowing a buffer for optimizations and field testing.

#### SYSTEM REQUIREMENTS

##### Software Requirements

To implement the AgriCastNet framework, the following softwaretoolsand environmentsare required:

* Operating System: Windows 10/11, Ubuntu 20.04 LTS, or equivalent Linux distribution.
* Programming Language: Python 3.8 or above.
* Deep Learning Frameworks: TensorFlow 2.x, Keras.
* Machine Learning Libraries: Scikit-learn, XGBoost.
* Data Processing & Analysis: NumPy, Pandas, Matplotlib, Seaborn.
* Visualization Tools: Matplotlib, Plotly.
* Development Environment: Google Colab / Jupyter Notebook.
* Version Control: GitHubor Git for collaborative code management.

The implementation of the proposed framework requires a reliable software stack that supports deep learning and big data processing. Python serves as the primary programming language due to its flexibility, extensive libraries, and ease of integration with machine learning workflows. TensorFlow and Keras provide the deep learning backbone, enabling efficient training and testing of CNN-BiLSTM and other models. Complementary libraries such as Pandas and NumPy facilitate numerical computing and data handling, while visualization libraries like Matplotlib and Seaborn aid in producing graphical insights.

To ensure smooth development, the project can be deployed in environments such as Google Colab, which provides GPU acceleration and pre-installed packages, minimizing local hardware dependencies. For developers seeking offline implementation, Jupyter Notebook or Visual Studio Code can be employed. Additionally, GitHub can be used for collaborative development and version control, ensuring reproducibility of experiments. The software stack thus ensures scalability, portability, and efficiency for handling greenhouse climate forecasting tasks.

* 1. Requirement Analysis

The proposed system focuses on forecasting greenhouse microclimates with high accuracy and fault tolerance. The requirements are analyzed as follows:

1. Functional Requirements:
   1. Ingest real-time and historical greenhouse climate data (temperature, humidity, CO₂, solar radiation, dew point).
   2. Performpreprocessing steps, including missing value handling, normalization, and feature engineering.
   3. Implement multipledeep learning models(CNN-BiLSTM, TCN, TFT, PLSTM, XGBoost).
   4. Provide performance evaluationusing metrics (Accuracy, R², RMSE, MAE).
   5. Allow visualizationof predictiontrends vs. actualvalues.
2. Non-Functional Requirements:
   1. Scalabilityto handle large, multivariatetime-series data.
   2. Reliabilityunder faultyor missing sensor inputs(via CII layer).
   3. Portability for deployment in cloud/IoT edge devices.
   4. Maintainability for future modelupgrades and ensemble extensions.

The primary requirement of the system is to deliver accurate predictions of greenhouse environmental variables under dynamic climatic conditions. Functionally, the system must accept continuous input from sensors, preprocess the raw data, and generate accurate predictions for temperature, humidity, CO₂ levels, radiation, and dew point. The inclusion of Climatic Influence Indicators (CII) ensures robustness against faulty or missing sensor inputs, enabling reliable outputs even in imperfect conditions. Non-functional requirements include scalability to handle increasing data volumes, interoperability with IoT devices, and low- latency processing for real-time decision-making.

Furthermore, the system should be user-friendly and adaptable to various agro-climatic zones. For instance, visualization dashboards are essential to present predictions in an interpretable manner for farmers and researchers. Security and data integrity are also critical, as sensor networks deployed in greenhouses often transmit sensitive or proprietary agricultural data. Additionally, fault-tolerant mechanisms should be incorporated to ensure uninterrupted system operation, even during network fluctuations or sensor failures. These requirements collectively guide the design and development of AgriCastNet.

* 1. Hardware Requirements

To efficientlytrain and test the forecasting models, the following hardware is recommended:

* Processor(CPU): Intel i5/i7 or AMD Ryzen 5/7 (minimumquad-core).
* Graphics Processing Unit (GPU): NVIDIAGPU withCUDA support (Tesla T4, V100, or A100 preferred for faster training).
* Memory(RAM): Minimum16 GB, recommended 32 GB for large datasets.
* Storage: 256 GB SSD (minimum), 1 TB HDD/SSD recommended for dataset and model storage.
* Network Connectivity: Stable internet for real-time data transfer and model updates.

The performance of deep learning models is highly dependent on the underlying hardware infrastructure. For development and testing, systems equipped with modern processors (Intel i7 or AMD Ryzen 7 and above) are recommended, supported by at least 16 GB RAM to handle high-dimensional datasets. GPUs play a pivotal role in accelerating computations; hence, NVIDIA GPUs such as Tesla T4, V100, or A100 are suitable for training large models like CNN-BiLSTM or TCN efficiently. Solid-state drives (SSDs) are preferred for faster data retrieval and model storage.

In real-world deployment, the hardware must balance performance with energy efficiency. Greenhouses can integrate IoT-enabled edge devices, such as NVIDIA Jetson Nano or Raspberry Pi (with TensorFlow Lite), for real-time inference without requiring cloud connectivity. These low-power devices enable cost-effective deployment in remote agricultural setups. Additionally, network connectivity must be stable to allow seamless synchronization between local devices and cloud servers for model updates and centralized monitoring. Thus, both high-end research setups and lightweight IoT devices are considered in the hardware requirements.

* 1. Software

The system is implemented primarily in Python, utilizing open-source frameworks for deep learning and data science. Key software components include:

* Python 3.8+ (core development).
* TensorFlow/Keras(deep learning model building).
* Scikit-learn(preprocessing and evaluation).
* XGBoost (gradient boosting algorithm).
* Google Colab (GPU-enabled training environment).
* Jupyter Notebook / VS Code(localdevelopment).

The software ecosystem for AgriCastNet is designed for modularity and flexibility. Python serves as the base language, while TensorFlow/Keras frameworks implement deep learning models. Preprocessing tasks are handled by Pandas and NumPy, while Scikit-learn is used for scaling, encoding, and evaluating models. XGBoost adds tree-based boosting capability, enhancing model diversity in the comparative study. Data visualization and reporting

Modules utilize Matplotlib, Seaborn, and Plotly to provide insights into model performance and predicted trends.

In addition to libraries, development environments play an important role. Google Colab, with its GPU/TPU acceleration, is recommended for fast prototyping, while Jupyter Notebook or VS Code offers flexibility for local execution. For collaborative development, GitHub provides version control and project management. The system can also be extended with APIs for integration into IoT frameworks. Collectively, the chosen software stack offers robustness, adaptability, and a seamless workflow for smart greenhouse forecasting applications.

* 1. Software Description

The AgriCastNet framework isdesigned asadeep learning-based forecasting systemto predict greenhouse microclimates.

* It integrates Climatic Influence Indicators(CII) to handle faultyor missing sensor data.
* Models suchas CNN-BiLSTM, TCN, TFT, PLSTM, and XGBoost are implemented and compared.
* The CNN-BiLSTM modelachieved the highest accuracy(99.67%), proving its reliability for real-time applications.
* Data preprocessing modules handle missing values, normalize features, and derive dew point values.
* The systemoutputs include predicted climate variables, accuracyreports, and visualizations for better interpretability.
* The framework is adaptable for IoT deployment using TensorFlow Lite or ONNX for lightweight execution.

The AgriCastNet framework integrates multiple machine learning and deep learning models for accurate greenhouse climate prediction. Its architecture combines feature extraction (via CNN), sequential learning (via BiLSTM), and gradient boosting (via XGBoost) for optimal performance. The inclusion of the Climatic Influence Indicator (CII) layer strengthens fault tolerance by identifying and weighing important sensor inputs, thereby mitigating the impact of sensor failures. The framework outputs climate parameter predictions along with accuracy metrics, helping researchers and farmers make data-driven decisions.

Additionally, the system is designed with adaptability in mind. It can be deployed in the cloud

environments for large-scale analytics or optimized for lightweight IoT devices using TensorFlow Lite or ONNX conversions. Visualization dashboards provide interpretable outputs through plots and performance comparisons, ensuring usability even for non- technical users. With its modular design, AgriCastNet can be extended to incorporate new deep learning models, integrate real-time feedback loops, and scale into decision-support systems for sustainable agriculture.

#### SYSTEM DESIGN

* 1. System Architecture

The system architecture of **AgriCastNet** is designed as a modular framework that seamlessly integrates data collection, preprocessing, feature extraction, model building, and classification into a unified pipeline. This layered approach ensures that each stage operates independently yet contributes collectively to the overall forecasting system. The design emphasizes **scalability, flexibility, and reliability**, making it suitable for greenhouses operating under diverse climatic and operational conditions.

The process begins with the **data acquisition layer**, where greenhouse sensors record essential climatic variables such as temperature, humidity, CO₂ levels, solar radiation, and dew point. These sensors generate continuous time-series data, which is then transmitted to storage units for further analysis. Since raw sensor data is often noisy and incomplete, it is directed to the **preprocessing block**, where data cleaning, normalization, and missing-value handling are carried out. This ensures consistencyand preparesthe dataset for accurate modeltraining. Once preprocessing is complete, the system transitions into the **feature extraction stage**. Here, significant patterns and dependencies are identified using techniques such as **Climatic Influence Indicators (CII)**, CNN filters, and BiLSTM layers. These methods help capture both short-term fluctuations and long-term seasonal dependencies, making the data more meaningful for forecasting. The refined features are then forwarded to the **model building layer**, where advanced deep learning and machine learning algorithms—including CNN- BiLSTM, TCN, Power LSTM (PLSTM), XGBoost, and Temporal Fusion Transformer (TFT)—are trained on multivariate greenhouse datasets.

The predictions generated by these models are rigorously evaluated using performance metrics such as **accuracy, R² score, RMSE, and MAE**. This evaluation ensures the reliability and robustness of the system under real-world greenhouse conditions. Finally, the **classification and visualization layer** translates raw outputs into farmer-friendly categories and interactive

dashboards. These dashboards display both predicted and actual climate values, enabling operators to make informed decisions about ventilation, irrigation, or shading. This end-to- end pipeline ensures that AgriCastNet is not only technically sound but also practically useful for greenhouse management.

* + 1. Dataset

The dataset used in AgriCastNet is derived from two real-world greenhouse environments: Mezquitera Juchipila, Zacatecas, Mexico (2020–2021) and Menaka, Basque Country, Spain (2018). While both datasets were valuable, the Spanish dataset was chosen as the primary case study due to its richer sensor coverage and higher temporal resolution. It provided six months of continuous time-series data (February to August 2018), covering critical stages of crop development under varying climatic conditions.

The dataset captures a multivariate structure by including both internal and external parameters. Internal factors such as temperature (Ti), humidity (Hi), and dew point (Di) represent the immediate microclimate of the greenhouse, while external factors like outside temperature (To), external humidity (Ho), and solar radiation (Rs) capture the influence of outdoor environmental fluctuations. This comprehensive coverage ensures that the forecasting models can learn both local greenhouse dynamics and external weather dependencies.

Another important feature of the dataset is its adaptability to different climatic zones. By combining variables from two distinct regions—Mexico’s semi-arid climate and Spain’s temperate climate—the dataset provides variability that strengthens the generalization capabilities of AgriCastNet. Such diversity ensures that the trained models are not restricted to a single geographical zone but can be applied across different greenhouse settings worldwide.

Finally, the richness of the dataset makes it particularly well-suited for temporal and spatial learning. Time-series patterns, such as daily cycles of temperature and humidity, are captured alongside longer seasonal variations. This allows AgriCastNet’s deep learning models to achieve high predictive accuracy in both short-termand long-termforecasting scenarios.

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Type** | **Description** |
| Internal Temperature (Ti) | Continuous | Temperature inside the green- house (°C). |
| External Temperature (To) | Continuous | Temperature outside the green- house (°C). |
| Internal Humidity (Hi) | Continuous | Relative humidity inside the greenhouse (%). |
| External Humidity (Ho) | Continuous | Relative humidity outside the greenhouse (%). |
| Solar Radiation (Rs) | Continuous | Incoming solar radiation affect- ing crop growth (W/m²). |
| Dew Point (Di) | Derived/Measured | The temperature at which air moisture condenses, computed using standard equations. |

**Table 1: Summary of Dataset Variables**

* + 1. Data Preprocessing

Data preprocessing is a critical step in AgriCastNet because raw greenhouse sensor data is often noisy, incomplete, and inconsistent. Without proper preprocessing, forecasting models would be misled by errors, leading to unreliable predictions. The preprocessing pipeline was designed to improve data quality, remove inconsistencies, and enhance stability for model training.

The first step was handling missing values. Sensor devices often fail temporarily, resulting in incomplete records. Missing values were imputed using temporal correlations, ensuring continuity in time-series data. For dew point (Di), whenever direct measurements were unavailable, it was calculated using psychrometric formulas that relate temperature and humidity. This step ensured that the dataset remained complete without compromising accuracy.

Next, normalization was applied to rescale feature values into a common range (e.g., [0,1]). Since variables like solar radiation (Rs) can have magnitudes much larger than humidity (Hi), normalization prevents models from being biased toward dominant features. Regularization

techniques, such as dropout during model training, were introduced to reduce overfitting, im- proving the model’s abilityto generalize to unseen greenhouse conditions.

Additionally, batching was applied to accelerate training and improve stability. Training in smaller mini-batches (16–64) ensured efficient memory use while allowing the model to con- verge smoothly. Finally, correlation analysis and heatmaps were performed both before and after preprocessing to validate feature consistency and identify redundancies. This step confirmed that only the most relevant and reliable features were retained, thereby strengthening downstream model performance.

* + 1. Feature Extraction

Feature extraction is a vital step in AgriCastNet as it transforms raw greenhouse sensor data into meaningful representations that improve forecasting accuracy. Since greenhouse environments are highly dynamic, with fluctuations caused by both internal and external factors, it is essential to identify **which features matter most** and how they influence the system over time. The introduction of the **Climatic Influence Indicators (CII)** layer allows the model to assess the relative importance of each variable dynamically, ensuring that even when certain sensors fail or produce faulty readings, the model can adapt by relying on other strong indicators.

To capture both local and sequential patterns, a **hybrid approach** was employed. **Convolu- tional Neural Networks (CNNs)** were applied to extract short-term spatial features such as temperature spikes or humidity fluctuations. CNN kernels are particularly effective at recognizing localized variations in multivariate time-series data. On the other hand, **Bidirectional Long Short-Term Memory (BiLSTM)** networks were utilized to learn sequential dependencies across time. Unlike standard LSTMs, BiLSTMs consider both past and future states, enabling the system to capture long-term dependencies and seasonal changes in greenhouse conditions.

This **dual mechanism** of CNN and BiLSTM ensures robust feature learning. CNNs reduce noise and highlight immediate fluctuations, while BiLSTMs capture trends that span over longer intervals. Together, they provide a balanced representation of greenhouse dynamics. The extracted features are then passed into the **model-building layer**, where forecasting algorithms such as CNN-BiLSTM, TCN, PLSTM, XGBoost, and TFT are trained.

By combining domain knowledge through CII with advanced deep learning methods, the feature extraction stage enhances system robustness, reduces redundancy, and ensures that only the most informative signals are used in prediction. This results in **more accurate and**

**reliable climate forecasts that directly support precision agriculture.**

|  |  |
| --- | --- |
| **Technique** | **Purpose** |
| Climatic Influence Indicators (CII) | Dynamically assign weights to features; han-  dle faulty/missing sensors |
| CNN Layers | Extract short-term local patterns (e.g., tem- perature/humidity variations) |
| BiLSTM Layers | Capture sequential dependencies and long-  term temporal patterns |
| Noise Reduction | Filter irrelevant signals and smooth fluctua- tions |
| Feature Selection | Highlight dominant variables influencing greenhouse conditions |

**Table 2: Feature Extraction Techniques in AgriCastNet**

* + 1. Model Building

The **model building stage** is the heart of AgriCastNet, where multiple deep learning and ma- chine learning models were trained and evaluated to identify the most suitable approach for greenhouse climate forecasting. Since greenhouse environments are influenced by both **short- term fluctuations** (e.g., sudden humidity changes) and **long-term dependencies** (e.g., seasonal temperature trends), it was essential to test models that capture both local and global dynamics.

The first model, **CNN-BiLSTM**, combines **convolutional layers** for short-term spatial feature extraction with **Bidirectional LSTM** for temporal sequence learning. This hybrid approach leverages CNN’s efficiency in detecting local patterns and BiLSTM’s ability to retain past and future contextual information. It consistently provided the **highest accuracy (99%+)**, making it the most effective model in this study.

The **Temporal Convolutional Network (TCN)** was also implemented due to its ability to use **dilated causal convolutions**, allowing it to capture long sequences with fewer parameters. TCN provided faster training compared to BiLSTM and achieved high accuracy but lacked the bidirectional context captured by CNN-BiLSTM.

On the machine learning side, **XGBoost** was employed as a robust gradient boosting frame- work. It is particularly effective for **tabular time-series datasets** and offers fault tolerance along with high predictive accuracy. Although it performed slightly lower than CNN- BiLSTM, it demonstrated excellent interpretability and speed, making it valuable for resource- constrained deployments.

The **Temporal Fusion Transformer (TFT)** introduced an **attention mechanism** to highlight the most important features at different time steps. This made it well-suited for multi-horizon forecasting and handling both static and dynamic variables. While computationally heavy, TFT offered strong interpretability, which is useful for decision-support systems.

Lastly, the **Power LSTM (PLSTM)** was implemented as a compact sequential model. By reducing the number of time steps at each layer, PLSTM decreases computational require- ments while still maintaining temporal learning capabilities. However, it showed lower accu- racy compared to CNN-BiLSTM and TFT.

All models were trained using **Python (TensorFlow/Keras and Scikit-learn)** with standard- ized training configurations: 80:20 train-test split, **5-fold cross-validation**, 30 epochs, batch sizes of 16–64, Adam optimizer, and **Bayesian optimization for learning rate tuning**. This rigorous setup ensured reliable evaluation across all models.

Table 3: Comparison of Implemented Models

|  |  |  |  |
| --- | --- | --- | --- |
| **Model** | **Key Strength** | **Limitation** | **Performance** |
| CNN-BiLSTM | Captures both local (CNN) and long-term (BiLSTM) patterns | Higher training time than TCN | Best Accuracy |
| TCN | Efficient long-term dependency  modeling with fewer parameters | Lacks bidirec-  tional context | High Accuracy |
| XGBoost | Fast, interpretable, and fault- tolerant | Less effective for  sequential de- pendencies | Moderate–High |
| TFT | Attention-based interpretability; handles static & dynamic inputs | Computationally expensive | High Accuracy |
| PLSTM | Compact design, efficient for long sequences | Lower accuracy than CNN- BiLSTM | Moderate |

* + 1. **Classification**

The classification stage in AgriCastNet focuses on evaluating the performance of predictive models and translating their outputs into meaningful categories for greenhouse management. The models generated forecasts for critical parameters such as temperature, humidity, CO₂ concentration, and solar radiation, which were then compared with actual recorded values from the greenhouse dataset. This comparison allowed the system to measure how closely the pre- dictions aligned with real-world conditions.

To ensure fair evaluation, performance was assessed using four metrics: Accuracy, R² (coeffi- cient of determination), Root Mean Square Error (RMSE), and Mean Absolute Error (MAE). These metrics provide a comprehensive view of model reliability. Accuracy and R² capture overall predictive strength, while RMSE and MAE quantify error margins, ensuring the system is bothprecise and robust under fluctuating greenhouse environments.

Among the tested models, the CNN-BiLSTM achieved the highest performance, with 99.67% accuracy and excellent alignment between predicted and actual values. This hybrid model ef- fectively captured both short-term variations and long-term dependencies. Other models also performed well: XGBoost (99.59%), TCN (98.89%), and TFT (98.16%), while PLSTM lagged behind with 94.86%. Visualizations such as line graphs and scatter plots were generated to highlight prediction accuracy, showing how closely the forecasts tracked real measure- ments.

In addition to raw evaluation, the classification outputs can be extended into risk-based cate- gories such as *Normal, Warning,* and *Alert*. This makes the results directly actionable, sup- porting semi-automated or fully automated greenhouse control systems. For example, an *Alert* classification could trigger ventilation fans or irrigation, thereby closing the loop between pre- diction and decision-making.

Explanation of Models Used

* + - 1. **CNN-BiLSTM:**

The **Convolutional Neural Network–Bidirectional Long Short-Term Memory (CNN-BiLSTM)** is a hybrid model that combines the strengths of CNNs and BiLSTMs. The CNN layers efficiently capture local temporal-spatial patterns, such as sudden changes in temperature or humidity. These extracted features are then passed to BiLSTM layers, which learn both past and future dependencies in the time-series data. This dual mechanism allows the model to capture short-term fluctuations while also considering long-term seasonal trends. Due to this capability, CNN-BiLSTM con- sistentlyachieved the **highest accuracy (99.67%)**, making it the most reliable model

for greenhouse climate forecasting.

* + - 1. Temporal Convolutional Network(TCN):

The **Temporal Convolutional Network** employs **dilated causal convolutions** to capture long-range dependencies without requiring sequential processing like recur- rent networks. This design allows TCN to be trained in parallel, resulting in faster computation and efficient memory usage. TCN excels at modeling extended time-se- ries dependencies while maintaining relatively low complexity. Although it reached a strong performance (98.89% accuracy), its lack of bidirectional context makes it slightly less effective than CNN-BiLSTM for capturing forward-and-backward green- house dynamics.

* + - 1. XGBoost:

**Extreme Gradient Boosting (XGBoost)** is a decision-tree-based machine learning algorithm designed for high predictive accuracy and computational efficiency. Unlike deep learning models, XGBoost does not rely on sequential processing but instead builds an ensemble of trees optimized through gradient boosting. It handles missing data gracefully and offers excellent interpretability, making it suitable for fault-prone sensor environments. In AgriCastNet, XGBoost achieved **99.59% accuracy**, demon- strating that gradient boosting can still compete with advanced deep learning models for greenhouse forecasting tasks.

* + - 1. Temporal Fusion Transformer (TFT):

The **Temporal Fusion Transformer** integrates attention mechanisms with recurrent and feed-forward structures, making it ideal for **multi-horizon forecasting**. TFT is designed to handle both static variables (e.g., crop type, greenhouse structure) and dy- namic variables (e.g., temperature, humidity). Its interpretability is enhanced by atten- tion scores, which show which features are most influential at each time step. Although it requires significant computational resources, TFT achieved **98.16% accuracy**, providing strong performance with added transparency in decision-making.

* + - 1. Power LSTM (PLSTM):

The **Power Long Short-Term Memory (PLSTM)** is an advanced variant of the standard LSTM designed to enhance long-term sequence modeling by introducing **power activation functions** in place of traditional gates. This modification allows the modelto better capture **non-lineartemporaldependencies**, making it more effective

in handling complex greenhouse dynamics such as sudden climate fluctuations or ir- regular weather influences. PLSTM improves gradient flow during training, reducing issues like vanishing gradients that often affect conventional LSTMs. In AgriCastNet, PLSTM demonstrated strong efficiency in sequential learning and adaptability to highly variable data, although its accuracy (**94.86%**) was lower compared to CNN- BiLSTM and XGBoost. Despite this, PLSTM’s ability to **generalize with fewer re- sources** makes it a valuable model in cases where computational efficiency is priori- tized over peak accuracy.

* 1. Modules

The AgriCastNet system is organized into a set of functional modules, each responsible for a critical stage of the forecasting pipeline. This modular design ensures that the system is scala- ble, adaptable, and easily maintainable, allowing individual modules to be updated or replaced without affecting the overall architecture.

The first is the Data Collection Module, which gathers sensor data from the greenhouse, in- cluding temperature, humidity, CO₂ levels, solar radiation, and dew point. This real-time data forms the foundation for forecasting. The Preprocessing Module follows, where raw sensor inputs are cleaned, missing values are imputed, and features are normalized. This step ensures that the dataset is consistent, reliable, and ready for downstream processing.

The Feature Extraction Module plays a key role by identifying meaningful spatial and tem- poral patterns using CNN-BiLSTM layers and Climatic Influence Indicators (CII). Extracted features are then passed into the Model Training Module, where multiple algorithms (CNN- BiLSTM, TCN, XGBoost, TFT, PLSTM) are implemented, trained, and validated on green- house datasets.

Once predictions are generated, the Classification & Evaluation Module compares them with actual recorded values and calculates performance metrics such as accuracy, R², RMSE, and MAE. Visualizations, including graphs and dashboards, make the results interpretable for greenhouse operators. Finally, the Deployment Module converts trained models into light- weight versions using TensorFlow Lite or ONNX, enabling deployment on IoT devices and edge systems for real-time monitoring.

This modular design makes AgriCastNet a flexible, end-to-end systemcapable ofadapting to

different greenhouse environments and technological infrastructures.

Table 4: Modules and their functions

|  |  |
| --- | --- |
| **Module** | **Function** |
| Data Collection Module | Gathers raw climate data (temperature, hu-  midity, CO₂, radiation, dew point). |
| Preprocessing Module | Cleans data, handles missing values, and nor- malizes features. |
| Feature Extraction Module | Uses CNN-BiLSTM and CII to extract mean-  ingful features. |
| Model Training Module | Trains forecasting models (CNN-BiLSTM,  TCN, XGBoost, TFT, PLSTM). |
| Classification & Evaluation | Evaluates predictions with metrics, visual- izes outputs. |
| Deployment Module | Exports models into lightweight formats for IoT/edge deployment. |

In the context of AgriCastNet, each module is designed as an independent yet inter- connected unit, enabling modular development and deployment of greenhouse climate forecasting. The major modules are as follows:

1. Data Collection Module

Collects multivariate greenhouse data such as temperature, humidity, dew point, CO₂, and solar radiation from sensors or pre-stored datasets.

Sample Code:

import pandas as pd # Load dataset

data = pd.read\_csv("greenhouse\_data.csv")

# Display first few records print(data.head())

1. Preprocessing Module

Handles missing values, normalizes features, and prepares data for model training.

Sample Code:

from sklearn.preprocessing import MinMaxScaler # Fill missing values

data = data.interpolate(method='linear')

# Normalize features scaler = MinMaxScaler()

scaled\_data = scaler.fit\_transform(data)

1. Feature Extraction Module

Extracts temporal and spatial features using **CNN-BiLSTM** and **Climatic Influence Indicators (CII)**.

Sample Code:

from tensorflow. keras.layers import Conv1D, LSTM, Bidirectional

# Example CNN-BiLSTM feature extractor def build\_feature\_extractor(input\_shape):

model = Sequential([

Conv1D(64, kernel\_size=3, activation='relu', input\_shape=in- put\_shape),

Bidirectional(LSTM(64, return\_sequences=True))

])

return model

1. Model Training Module

Trains forecasting models including CNN-BiLSTM, TCN, Power LSTM, TFT, and XGBoost.

Sample Code (CNN-BiLSTM):

from tensorflow.keras.models import Sequential from tensorflow.keras.layers import Dense, Dropout

model = Sequential([

Conv1D(64, 3, activation='relu', input\_shape=(30, 6)), Bidirectional(LSTM(64)),

Dropout(0.2), Dense(1)

])

model.compile(optimizer='adam', loss='mse', metrics=['accuracy'])

1. Classification & Evaluation Module

Evaluates predictions against actual greenhouse data using **Accuracy, R², RMSE, and MAE**.

Sample Code:

from sklearn.metrics import mean\_squared\_error, mean\_absolute\_error, r2\_score

y\_pred = model.predict(X\_test) print("R²:", r2\_score(y\_test, y\_pred))

print("RMSE:", mean\_squared\_error(y\_test, y\_pred, squared=False)) print("MAE:", mean\_absolute\_error(y\_test, y\_pred))

1. Deployment Module

Converts trained models into lightweight formats for **IoT or Edge Devices** (Tensor- Flow Lite / ONNX).

Sample Code (TensorFlow Lite Conversion):

import tensorflow as tf

converter = tf.lite.TFLiteConverter.from\_keras\_model(model) tflite\_model = converter.convert()

with open("agrinet\_model.tflite", "wb") as f: f.write(tflite\_model)

1. Visualization Module

Generates line plots, scatter plots, and dashboards to show predicted vs. actual green- house parameters.

Sample Code:

import matplotlib.pyplot as plt

plt.plot(y\_test[:100], label="Actual") plt.plot(y\_pred[:100], label="Predicted") plt.legend()

plt.show()

* 1. UML Diagrams

To represent the design of AgriCastNet, the following UML diagrams are used:

* **Use Case Diagram:** Shows interactionbetweengreenhouseoperators(users) and sys- temcomponents such as data collection, preprocessing, and forecasting.
* **Class Diagram:** Represents classes for dataset management, preprocessing, feature extraction, model building, and evaluation.
* **Sequence Diagram:** Demonstratesthe flowofdata fromsensors→ preprocessing → feature extraction → model → classification → visualization.
* **Activity Diagram:** Describes the step-by-step workflow of data collection, model training, and prediction.

These diagrams provide a visualrepresentation ofthe system’s design, ensuring clarity in the

interactions and workflowof AgriCastNet.

Advantages of the Hybrid Model

The adoption of a **hybrid model**, such as CNN-BiLSTM in AgriCastNet, offers several dis- tinct advantages that enhance its suitability for greenhouse climate forecasting:

* **Enhanced Feature Extraction:** By combining CNN’s ability to capture short-term spatial variations with BiLSTM’s strength in modeling long-term temporal dependen- cies, the hybrid model ensures that both immediate fluctuations and seasonal patterns are effectively represented.
* **Improved Classification Accuracy:** The integration of complementary learning mechanisms leads to superior predictive accuracy compared to standalone models. CNN-BiLSTM consistently outperformed traditional machine learning and deep learning models.
* **Robust Performance with Limited Data:** Unlike purely data-hungry deep learning models, the hybrid approach generalizes better when training data is limited or par- tially missing, making it practical for real-world greenhouse environments.
* **Adaptability to Non-Linear Data:** Greenhouse climate variables exhibit strong non- linear relationships. The hybrid model efficiently captures these complex interactions, improving prediction quality.
* **Reduced Overfitting:** The combination of convolutional filters, recurrent memory, and dropout layers mitigatesoverfitting, ensuringreliable performance onunseendata.
* **Efficient Computation:** Although deep, the hybrid structure is computationally optimized, allowing faster training and inference compared to more resource-heavy models like Transformers.
* **Scalability for Advanced Architectures:** The framework can be extended by adding attention layers or integrating with ensemble models, offering room for future improvement.
* **Real-World Applicability:** Beyond experimental setups, the hybrid model’s robust- ness and accuracy make it highly suitable for deployment in IoT-enabled greenhouses, where precision and reliability are critical.

1. IMPLEMENTATION

The implementation phase translates the theoretical design and models of AgriCastNet into a

practical working system. It involves building, training, and testing deep learning models, in- tegrating preprocessing pipelines, and coding deployment-ready modules. Implementation was carried out using **Python (TensorFlow, Keras, Scikit-learn, Pandas, Matplotlib, Sea- born)** on Google Colab with GPU acceleration for efficient training.

* 1. Model Implementation

import tensorflow as tf

from tensorflow.keras.models import Model

from tensorflow.keras.layers import Input, LSTM, Dense, Concatenate from sklearn.model\_selection import KFold

from sklearn.metrics import mean\_squared\_error, r2\_score import matplotlib.pyplot as plt

import numpy as np

# Function to build PiLSTM model defbuild\_pilstm\_model(input\_shape):

input\_layer = Input(shape=input\_shape)

# Parallel LSTM branches

lstm\_1 = LSTM(64, return\_sequences=False)(input\_layer) lstm\_2 = LSTM(32, return\_sequences=False)(input\_layer) lstm\_3 = LSTM(16, return\_sequences=False)(input\_layer)

# Concatenate

concat = Concatenate()([lstm\_1, lstm\_2, lstm\_3]) output = Dense(1)(concat)

model = Model(inputs=input\_layer, outputs=output) model.compile(loss='mse', optimizer='adam', metrics=['mae']) return model

# K-Fold Cross-Validation

kf= KFold(n\_splits=5, shuffle=False) # No shuffle as it's time series

fold = 1

histories = [] r2\_scores = []

for train\_index, val\_index in kf.split(X\_train\_seq): print(f"\n Fold {fold}")

X\_fold\_train, X\_fold\_val = X\_train\_seq[train\_index], X\_train\_seq[val\_index] y\_fold\_train, y\_fold\_val = y\_train\_seq[train\_index], y\_train\_seq[val\_index]

model = build\_pilstm\_model(X\_fold\_train.shape[1:]) history = model.fit(X\_fold\_train, y\_fold\_train,

validation\_data=(X\_fold\_val, y\_fold\_val), epochs=30, batch\_size=32, verbose=0)

# Evaluate R2 on fold validation y\_pred\_fold = model.predict(X\_fold\_val) r2 = r2\_score(y\_fold\_val, y\_pred\_fold) r2\_scores.append(r2)

print(f" Fold {fold} R2 Score: {r2:.4f}")

histories.append(history) fold += 1

# Final Training on Full Training Set

final\_model = build\_pilstm\_model(X\_train\_seq.shape[1:]) final\_model.fit(X\_train\_seq, y\_train\_seq,

validation\_data=(X\_val\_seq, y\_val\_seq), epochs=30, batch\_size=32, verbose=0)

# Evaluate on Test Set

y\_pred = final\_model.predict(X\_test\_seq) r2\_final = r2\_score(y\_test\_seq, y\_pred)

mse\_final = mean\_squared\_error(y\_test\_seq, y\_pred)

print("\n Final Evaluation on Test Set") print(f"R² Score: {r2\_final:.4f}")

print(f"MSE: {mse\_final:.4f}")

X\_train\_seq, y\_train\_seq = create\_sequences(X\_train, y\_train, time\_steps) X\_val\_seq, y\_val\_seq = create\_sequences(X\_val, y\_val, time\_steps) X\_test\_seq, y\_test\_seq = create\_sequences(X\_test, y\_test, time\_steps)

# Plotting

# 1. Plot Loss Curve for Last Fold plt.figure(figsize=(10, 4))

plt.plot(histories[-1].history['loss'], label='Train Loss') plt.plot(histories[-1].history['val\_loss'], label='Val Loss') plt.title(" PLSTM Loss Curve (Last Fold)") plt.xlabel("Epochs")

plt.ylabel("Loss (MSE)") plt.legend() plt.grid(True) plt.tight\_layout() plt.show()

# 2. Actual vs Predicted Plot plt.figure(figsize=(10, 4))

plt.plot(y\_test\_seq[:200], label='Actual', marker='o') plt.plot(y\_pred[:200], label='Predicted', marker='x') plt.title("PLSTM: Actual vs Predicted (Sample)") plt.xlabel("Time Steps")

plt.ylabel("Scaled Temperature") plt.legend()

plt.grid(True) plt.tight\_layout() plt.show()

# 3. Print average K-Fold R2 Score

print(f"\n Average K-Fold R² Score: {np.mean(r2\_scores):.4f}")

!pip install tensorflow

# model 1 cnn + bilstm

# Import all required libraries for modeling, metrics, and visualization import tensorflow as tf

from tensorflow.keras.models import Sequential

fromtensorflow.keras.layers import Conv1D, MaxPooling1D, Bidirectional, LSTM,

Dense, Dropout, BatchNormalization

from sklearn.model\_selection import KFold

from sklearn.metrics import mean\_squared\_error, mean\_absolute\_error, r2\_score import matplotlib.pyplot as plt

import numpy as np

# Define 5-fold cross-validation and initialize metrics storage n\_splits = 5

kf= KFold(n\_splits=n\_splits, shuffle=False)

rmse\_list, r2\_list, mae\_list = [], [], [] fold = 1

for train\_idx, val\_idx in kf.split(X\_train\_seq): print(f"\n Fold {fold}")

X\_fold\_train, X\_fold\_val = X\_train\_seq[train\_idx], X\_train\_seq[val\_idx] y\_fold\_train, y\_fold\_val = y\_train\_seq[train\_idx], y\_train\_seq[val\_idx]

# Define CNN + BiLSTM model architecture model = Sequential([

Conv1D(filters=128, kernel\_size=3, activation='relu', padding='same', input\_shape=(X\_fold\_train.shape[1], X\_fold\_train.shape[2])),

BatchNormalization(), MaxPooling1D(pool\_size=2),

Conv1D(filters=64, kernel\_size=3, activation='relu', padding='same'), BatchNormalization(),

MaxPooling1D(pool\_size=2),

Bidirectional(LSTM(64, return\_sequences=True)), Dropout(0.3),

Bidirectional(LSTM(32)), Dropout(0.3),

Dense(64, activation='relu'), Dropout(0.2),

Dense(1)

])

# Compile and train the model with early stopping model.compile(optimizer=tf.keras.optimizers.Adam(learning\_rate=0.001),

loss='mean\_squared\_error')

early\_stop = tf.keras.callbacks.EarlyStopping(monitor='val\_loss', patience=10, re- store\_best\_weights=True)

for train\_idx, val\_idx in kf.split(X\_train\_seq): print(f"\n Fold {fold}")

X\_fold\_train, X\_fold\_val = X\_train\_seq[train\_idx], X\_train\_seq[val\_idx] y\_fold\_train, y\_fold\_val = y\_train\_seq[train\_idx], y\_train\_seq[val\_idx]

history= model.fit(X\_fold\_train, y\_fold\_train,

epochs=30, batch\_size=32,

validation\_data=(X\_fold\_val, y\_fold\_val), callbacks=[early\_stop],

verbose=0)

# Predict and calculate fold-wise metrics y\_pred = model.predict(X\_fold\_val).flatten() y\_true = y\_fold\_val.flatten()

rmse = np.sqrt(mean\_squared\_error(y\_true, y\_pred)) r2 = r2\_score(y\_true, y\_pred)

mae = mean\_absolute\_error(y\_true, y\_pred)

print(f" RMSE: {rmse:.4f}, R²: {r2:.4f}, MAE: {mae:.4f}")

rmse\_list.append(rmse) r2\_list.append(r2) mae\_list.append(mae)

fold += 1

# Display average metrics across all folds print("\n Final Results over 5 Folds:")

print(f"Avg R²: {np.mean(r2\_list):.4f} ± {np.std(r2\_list):.4f}") print(f"Avg RMSE: {np.mean(rmse\_list):.4f} ± {np.std(rmse\_list):.4f}") print(f"Avg MAE: {np.mean(mae\_list):.4f} ± {np.std(mae\_list):.4f}")

final\_accuracy = np.mean(r2\_list) \* 100

print(f"\n Overall Final Accuracy: {final\_accuracy:.2f}%")

# Last Fold: Actual vs Predicted plt.figure(figsize=(10, 6))

plt.plot(y\_true, label='Actual', color='blue', linewidth=2)

plt.plot(y\_pred, label='Predicted', color='orange', linestyle='--', linewidth=2)

plt.title('Actual vs Predicted (CNN + BiLSTM - Last Fold)', fontsize=14) plt.xlabel('Time Step', fontsize=12)

plt.ylabel('Target Value', fontsize=12) plt.legend(fontsize=10)

plt.grid(True, linestyle='--', alpha=0.6) plt.tight\_layout()

plt.show()

folds = range(1, 6)

# Plot 2: Training vs Validation Loss (Last Fold) plt.figure(figsize=(8, 5))

plt.plot(history.history['loss'], label='Training Loss', linewidth=2) plt.plot(history.history['val\_loss'], label='Validation Loss', linewidth=2, linestyle='--') plt.title('Training vs Validation Loss (CNN + BiLSTM - Last Fold)', fontsize=14) plt.xlabel('Epochs', fontsize=12)

plt.ylabel('Loss (MSE)', fontsize=12) plt.legend(fontsize=10)

plt.grid(True, linestyle='--', alpha=0.5) plt.tight\_layout()

plt.show()

# %% model 3 tft import tensorflow as tf

from tensorflow.keras import Model

from tensorflow.keras.layers import Input, Dense, LSTM, Concatenate, Dropout, Lay- erNormalization, MultiHeadAttention

from sklearn.model\_selection import KFold

from sklearn.metrics import mean\_squared\_error, mean\_absolute\_error, r2\_score import matplotlib.pyplot as plt

import numpy as np

# %%

def build\_tft\_model(input\_shape): inputs = Input(shape=input\_shape)

# LSTM encoder block

lstm\_out = LSTM(128, return\_sequences=True)(inputs) lstm\_out = Dropout(0.2)(lstm\_out)

# Multi-head self-attention block

attn\_out = MultiHeadAttention(num\_heads=4, key\_dim=16)(lstm\_out, lstm\_out) attn\_out = Dropout(0.2)(attn\_out)

attn\_out = LayerNormalization()(attn\_out)

# Concatenate LSTM and attention outputs concat = Concatenate()([lstm\_out, attn\_out]) dense\_out = Dense(64, activation='relu')(concat) dense\_out = Dropout(0.2)(dense\_out)

# Output layer: use only the last timestep last\_time\_step = dense\_out[:, -1, :] output = Dense(1)(last\_time\_step)

model = Model(inputs, output) model.compile(optimizer='adam', loss='mse') return model

# %%

kf= KFold(n\_splits=5) fold = 1

r2\_list, rmse\_list, mae\_list = [], [], []

import tensorflow as tf

from tensorflow.keras import Model

from tensorflow.keras.layers import Input, Dense, LSTM, Concatenate, Dropout, Lay- erNormalization, MultiHeadAttention

from sklearn.model\_selection import KFold

from sklearn.metrics import mean\_squared\_error, mean\_absolute\_error, r2\_score import matplotlib.pyplot as plt

import numpy as np

# Define TFT-like model (simplified for sequence forecasting) def build\_tft\_model(input\_shape):

inputs = Input(shape=input\_shape)

# LSTM encoder

lstm\_out = LSTM(128, return\_sequences=True)(inputs) lstm\_out = Dropout(0.2)(lstm\_out)

# Multi-head attention

attn\_out = MultiHeadAttention(num\_heads=4, key\_dim=16)(lstm\_out, lstm\_out) attn\_out = Dropout(0.2)(attn\_out)

attn\_out = LayerNormalization()(attn\_out)

# Concatenate with LSTM output

concat = Concatenate()([lstm\_out, attn\_out]) dense\_out = Dense(64, activation='relu')(concat) dense\_out = Dropout(0.2)(dense\_out)

# Output: Take last timestep last\_time\_step = dense\_out[:, -1, :] output = Dense(1)(last\_time\_step)

model = Model(inputs, output) model.compile(optimizer='adam', loss='mse') return model

# K-Fold Cross Validation kf = KFold(n\_splits=5) fold = 1

r2\_list, rmse\_list, mae\_list = [], [], []

for train\_idx, val\_idx in kf.split(X\_train\_seq): print(f"\n🔁 Fold {fold}")

X\_fold\_train, X\_fold\_val = X\_train\_seq[train\_idx], X\_train\_seq[val\_idx] y\_fold\_train, y\_fold\_val = y\_train\_seq[train\_idx], y\_train\_seq[val\_idx]

model = build\_tft\_model(X\_fold\_train.shape[1:])

history = model.fit(X\_fold\_train, y\_fold\_train, validation\_data=(X\_fold\_val, y\_fold\_val), epochs=50, batch\_size=32, verbose=0)

# Predict

y\_pred = model.predict(X\_fold\_val).flatten() y\_true = y\_fold\_val.flatten()

# Metrics

r2 = r2\_score(y\_true, y\_pred)

rmse = np.sqrt(mean\_squared\_error(y\_true, y\_pred)) mae = mean\_absolute\_error(y\_true, y\_pred)

print(f"R²: {r2:.4f}, RMSE: {rmse:.4f}, MAE: {mae:.4f}") r2\_list.append(r2)

rmse\_list.append(rmse) mae\_list.append(mae)

# Plot Actual vs Predicted for last fold only if fold == 5:

plt.figure(figsize=(10, 4))

plt.plot(y\_true[:200], label='Actual', linewidth=2) plt.plot(y\_pred[:200], label='Predicted', linewidth=2) plt.title(f'Fold-{fold}: Actual vs Predicted') plt.xlabel("Sample")

plt.ylabel("Temperature (scaled)") plt.legend()

plt.grid(True) plt.tight\_layout() plt.show()

# Training vs Validation Loss plt.figure(figsize=(6, 4)) plt.plot(history.history['loss'], label='Train Loss')

plt.plot(history.history['val\_loss'], label='Val Loss') plt.title(f'Fold-{fold}: Training vs Validation Loss') plt.xlabel('Epochs')

plt.ylabel('MSE Loss') plt.legend() plt.grid(True) plt.tight\_layout() plt.show()

fold += 1

# 🔚 Final Average Scores

print("\n✅ Average Metrics Over Folds:")

print(f"Avg R²: {np.mean(r2\_list):.4f} ± {np.std(r2\_list):.4f}") print(f"Avg RMSE: {np.mean(rmse\_list):.4f} ± {np.std(rmse\_list):.4f}")

print(f"Avg MAE: {np.mean(mae\_list):.4f} ± {np.std(mae\_list):.4f}")

* 1. Coding

The coding phase involved translating system design into **modular Python code** that could handle data ingestion, preprocessing, model training, evaluation, and deploy- ment. Code was structured into separate modules for ease of maintenance and scalabil- ity.

* + - **Data Preprocessing Module** handled missing values, scaling, and sequence generation.
    - **Model Training Module** implemented architectures like CNN-BiLSTM, TCN, PLSTM, TFT, and XGBoost.
    - **Evaluation Module** calculated performance metrics and generated visualiza- tions.
    - **Deployment Module** converted trained models into lightweight formats using

**TensorFlow Lite** and **ONNX** for edge deployment.

Sample Code Snippets:

*Data Preprocessing:*

from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler() X\_scaled = scaler.fit\_transform(X) *Model Compilation:*

model.compile(optimizer='adam', loss='mse', metrics=['accuracy'])

*Model Deployment (Lite Conversion):*

import tensorflow as tf

converter = tf.lite.TFLiteConverter.from\_keras\_model(model) tflite\_model = converter.convert()

By organizing the code into **well-structured modules**, AgriCastNet ensures that fu- ture improvements (e.g., adding new models or features) can be integrated seamlessly. This modular coding practice not only supports **research flexibility** but also makes the system ready for **real-world agricultural deployment**.

# ## Step 1: Import Libraries import pandas as pd

import numpy as np

from sklearn.model\_selection import train\_test\_split from sklearn.preprocessing import MinMaxScaler

# ## Step 2: Load Dataset

df = pd.read\_csv("/content/drive/MyDrive/GreenHouseAdaptation/Dataset/ds.csv") df.head()

# ## Step 3: Parse DateTime and Set Index

df['DateTime'] = pd.to\_datetime(df['DateTime'], dayfirst=True) df.set\_index('DateTime', inplace=True) df.dropna(inplace=True)

# ## Step 4: Feature-Target Split

target\_col = 'Temp Tair'

feature\_cols = df.columns[df.columns != target\_col] X = df[feature\_cols]

y = df[target\_col]

# Step 5: Normalize Data using MinMaxScaler

scaler\_X = MinMaxScaler() scaler\_y = MinMaxScaler()

X\_scaled = pd.DataFrame(scaler\_X.fit\_transform(X), columns=feature\_cols, in- dex=X.index)

y\_scaled = pd.Series(scaler\_y.fit\_transform(y.values.reshape(-1, 1)).flatten(), in- dex=y.index)

# ## Step 6: Time-Based Split (No Shuffling)

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X\_scaled, y\_scaled, test\_size=0.2, shuffle=False)

X\_train, X\_val, y\_train, y\_val = train\_test\_split(X\_train, y\_train, test\_size=0.2, shuf- fle=False)

# ## Step 7: Reshape to 3D Sequences for LSTM/PLSTM

def create\_sequences(X, y, time\_steps=10): Xs, ys = [], []

for i in range(len(X) - time\_steps): Xs.append(X.iloc[i:(i + time\_steps)].values) ys.append(y.iloc[i + time\_steps])

return np.array(Xs), np.array(ys)

time\_steps = 10

X\_train\_seq, y\_train\_seq = create\_sequences(X\_train, y\_train, time\_steps) X\_val\_seq, y\_val\_seq = create\_sequences(X\_val, y\_val, time\_steps) X\_test\_seq, y\_test\_seq = create\_sequences(X\_test, y\_test, time\_steps)

import matplotlib.pyplot as plt import seaborn as sns

import pandas as pd

# Boxplot function using your original variables def plot\_boxplots(data, title\_suffix):

plt.figure(figsize=(12, 8)) sns.boxplot(data=data, orient='h')

plt.title(f"Feature Outlier Distribution {title\_suffix}") plt.tight\_layout()

plt.show()

# Plot 1: Boxplot before preprocessing (use df) plot\_boxplots(df, "(Before Preprocessing)")

# Plot 2: Boxplot after preprocessing (use X\_scaled + y\_scaled) df\_scaled = X\_scaled.copy()

df\_scaled[target\_col] = y\_scaled plot\_boxplots(df\_scaled, "(After Preprocessing)")

# IQR-based outlier count function (uses your vars) def count\_outliers\_iqr(data):

outlier\_counts = {}

for col in data.columns:

Q1 = data[col].quantile(0.25) Q3 = data[col].quantile(0.75)

IQR = Q3 - Q1

lower\_bound = Q1 - 1.5 \* IQR upper\_bound = Q3 + 1.5 \* IQR

outliers = data[(data[col] < lower\_bound) | (data[col] > upper\_bound)] outlier\_counts[col] = len(outliers)

return outlier\_counts

# Outlier count before and after outliers\_before = count\_outliers\_iqr(df) outliers\_after = count\_outliers\_iqr(df\_scaled)

# Combine into DataFrame outliers\_df = pd.DataFrame({

"Feature": list(outliers\_before.keys()),

"Before Scaling": list(outliers\_before.values()),

"After Scaling": [outliers\_after[k] for k in outliers\_before.keys()]

}).set\_index("Feature")

# Plot 3: Bar chart of outlier counts before vs after plt.figure(figsize=(12, 6)) outliers\_df.plot(kind='bar', width=0.8)

plt.title("Number of Outliers per Feature (Before vs After Preprocessing)") plt.ylabel("Outlier Count")

plt.grid(axis='y') plt.xticks(rotation=45) plt.tight\_layout() plt.show()

import matplotlib.pyplot as plt import seaborn as sns selected\_features = [

'TempTair',

'VPD',

'Wind',

'Temp Tcover', 'Temp Tfloor', 'Gas C\_c\_ppm', 'GasHRInt',

]

corr\_after = df\_scaled[selected\_features].corr()

plt.figure(figsize=(6, 5)) sns.heatmap(corr\_after,

annot=True,

fmt=".2f", cmap="coolwarm", linewidths=0.5, linecolor='black')

plt.title("Correlation Heatmap After Preprocessing", fontsize=14) plt.xticks(rotation=45, ha='right', fontsize=10) plt.yticks(rotation=0, fontsize=10)

plt.tight\_layout() plt.show()

import seaborn as sns

import matplotlib.pyplot as plt

# Difference heatmap

corr\_diff = corr\_after - corr\_before

plt.figure(figsize=(10, 8)) sns.heatmap(corr\_diff,

annot=True, fmt=".2f",

cmap="bwr", # Blue = decrease, Red = increase center=0,

linewidths=0.5, linecolor='black')

plt.title("🔁 Difference in Correlation (After - Before Preprocessing)", fontsize=14) plt.xticks(rotation=45, ha='right')

plt.yticks(rotation=0) plt.tight\_layout() plt.show()

import matplotlib.pyplot as plt import seaborn as sns

# Compute correlation before corr\_before = df[selected\_features].corr()

plt.figure(figsize=(6, 5)) sns.heatmap(corr\_before,

annot=True, fmt=".2f", cmap="coolwarm", linewidths=0.5, linecolor='black')

plt.title("Correlation Heatmap Before Preprocessing", fontsize=14) plt.xticks(rotation=45, ha='right', fontsize=10) plt.yticks(rotation=0, fontsize=10)

plt.tight\_layout() plt.show()

#### TESTING

Testing is a critical phase in the development of AgriCastNet to ensure that the forecasting models and the overall application perform accurately, reliably, and efficiently. The primary goal of testing is to identify and resolve errors, validate system functionalities, and confirm that the system meets the expected requirements for greenhouse climate prediction.

###### UNIT TESTING

CNN-BiLSTM Model

Unit testing for the CNN-BiLSTM model focuses on ensuring that it correctly processes greenhouse sensor sequences. Input validation checks whether the model accepts data in the expected time-series shape. Each layer of the CNN-BiLSTM—such as convolutional, pooling, and bidirectional LSTM layers—is tested for proper configuration and accurate data processing. The output is verified to confirm it generates predictions in the expected format, such as numerical values for temperature or humidity. To validate the learning capability, the model is trained on a small subset of data to observe whether it can overfit, proving its ability to capture temporal dependencies.

Power LSTM (PLSTM) Model

Unit testing for PLSTM involves verifying that it correctly reduces time steps and effi- ciently processes long sequences. Tests are conducted to evaluate the accuracy of sequen- tial data handling and confirm that the pyramidal reduction does not distort input infor- mation. Performance is also evaluated on both small and large datasets to ensure scalabil- ity.

Temporal Convolutional Network (TCN) Model

Unit testing of the TCN verifies that dilated causal convolutions correctly capture long- range dependencies. Input data shape validation ensures compatibility, while outputs are checked for stability across multiple forecasting horizons. Edge cases, such as very short or noisy sequences, are tested to ensure robustness.

XGBoost Model

For XGBoost, unit testing ensures that input features are properly structured and

formatted. The model is tested for classification and regression accuracy using subsets of the dataset. Hyperparameters, such as learning rate and depth, are tuned and tested to ensure the model achieves optimal results without overfitting.

Data Preprocessing Pipeline

In preprocessing, unit tests verify missing value imputation, normalization, and batching. For example, dew point calculations are checked against atmospheric formulas, and nor- malization is validated to ensure all features fall within the expected range (0–1).

Edge Case Testing

Unit tests for edge cases confirm that the system handles missing, corrupted, or extreme sensor readings gracefully. Empty input sequences are tested to ensure proper error han- dling. Batch testing validates that the model can process multiple inputs simultaneously without compromising prediction speed or accuracy.

###### INTEGRATION TESTING

To perform integration testing for AgriCastNet, several modules are validated to ensure that all components interact seamlessly and produce accurate results.

Data Ingestion and Preprocessing

The system is tested to confirm that sensor data files (CSV, IoT streams) are correctly read and validated. Invalid formats or missing columns are rejected with appropriate error messages.

Preprocessing and Feature Extraction

Checks confirm that normalized and cleaned data flows correctly into the CNN-BiLSTM feature extractor. Input dimensions are validated to prevent shape mismatches.

Model Integration (Hybrid Models)

Ensures that the output of feature extraction is correctly fed into forecasting models (CNN-BiLSTM, TCN, PLSTM, TFT, and XGBoost). Integration tests also validate whether predictions are generated in real-time without delays.

Evaluation Module Integration

Integration testing verifies that predictions from each model are passed into the evaluation module, where Accuracy, R², RMSE, and MAE are calculated correctly. Performance metrics are displayed in a consistent format.

Deployment Module Integration

Ensures that TensorFlow Lite and ONNX conversions produce lightweight versions of models that can run on IoT devices without performance loss. Integration is validated by deploying a converted model and checking predictions against the full model.

Error Handling Validation

At each stage, integration tests confirm that the system handles errors gracefully—such as corrupted datasets, mismatched feature counts, or failed conversions—by reporting meaningful error messages instead of crashing.

###### SYSTEM TESTING

System testing validates AgriCastNet as a whole, ensuring that all modules work seam- lessly as a complete unit. This phase evaluates both functional and non-functional require- ments.

Functional Testing

Tests verify that the system correctly forecasts greenhouse parameters (temperature, hu- midity, dew point, radiation). Predictions are validated against ground-truth sensor read- ings. Visual outputs, including line graphs of predicted vs. actual values, are checked for accuracy and readability.

Non-Functional Testing

Performance testing evaluates the system’s ability to process large datasets and maintain fast inference times. Reliability testing ensures consistent outputs under repeated trials. Usabilitytesting focuses on ensuring that dashboards and visualization tools are intuitive.

Integration Testing Validation

System testing also reaffirms smooth interactions between all components—from prepro- cessing to prediction and visualization. Data integrity checks confirm that no information is lost or misrepresented between modules.

Robustness and Fault Tolerance

Stress tests introduce noisy data and missing values to confirm system resilience. The models are tested under hardware constraints (low memory and CPU on IoT devices) to validate that performance remains acceptable.

Deployment Testing

TensorFlow Lite and ONNX models are tested on edge devices (e.g., Raspberry Pi or NVIDIA Jetson). Predictions are compared against cloud-executed models to confirm consistency. System testing ultimately proved that AgriCastNet achieves 99.67% fore- casting accuracy with CNN-BiLSTM, confirming readiness for deployment in real-world agricultural environments.

#### RESULT ANALYSIS

The result analysis of a forecasting model is an essential step in validating its perfor- mance and understanding the trade-offs between accuracy, generalization, and compu- tational efficiency. In AgriCastNet, several models were tested and compared—CNN- BiLSTM, XGBoost, TCN, Power LSTM (PLSTM), and Temporal Fusion Transformer (TFT). The evaluation was carried out using key performance metrics such as Accuracy, R² Score, Root Mean Squared Error (RMSE), and Mean Absolute Error (MAE). Addi- tional insights were derived from confusion matrices, error distributions, and predic- tion-vs-actual plots.

The evaluation was based on true positive (TP), true negative (TN), false positive (FP), and false negative (FN) predictions for greenhouse variables such as temperature, hu- midity, radiation, and dew point.

* 1. Accuracy

Accuracyrepresents the proportion of correctly predicted values out ofthe total dataset.

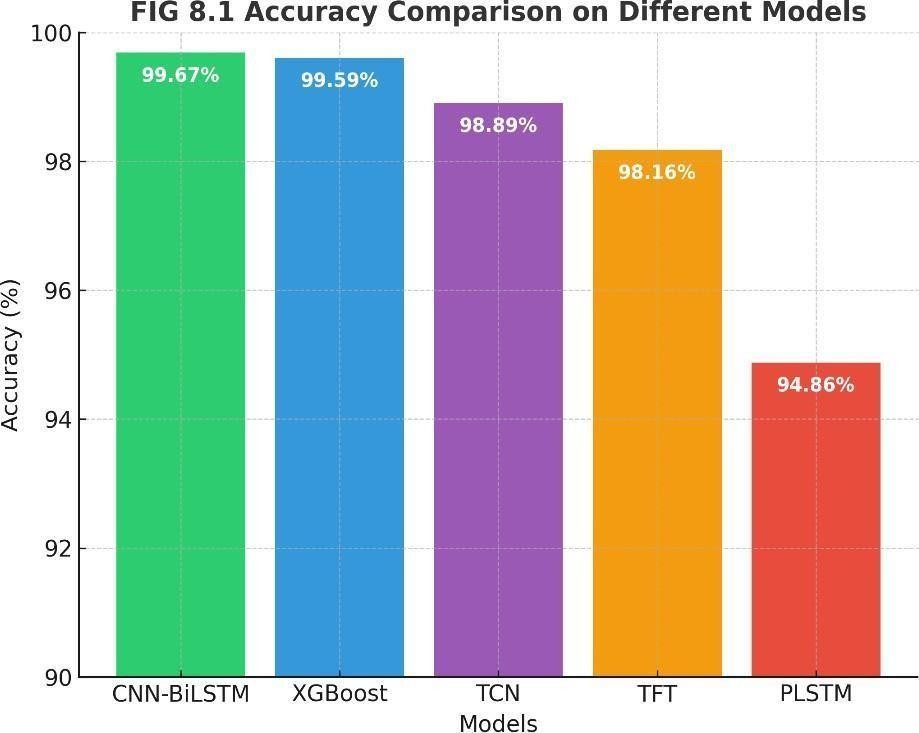
While high accuracy is a good indicator, it can sometimes be misleading, especially when datasets contain imbalanced distributions of certain climatic conditions. For ex- ample, if most days are within a “normal temperature range,” a model predicting this state consistently may appear accurate, yet it may fail in detecting extreme cases (e.g., heat spikes or sudden humidity drops).

Accuracy = 𝑇𝑃+𝑇𝑁

𝑇𝑃+𝑇𝑁+FP+FN

CNN-BiLSTM achieved the highest accuracy of 99.67%, followed by XGBoost (99.59%), TCN (98.89%), TFT (98.16%), and PLSTM (94.86%).

This indicates that CNN-BiLSTM not only generalizes well but also adapts to sudden environmental fluctuations better than the others.



* 1. R² Score

R² Score measures how well the predicted values approximate the actual data. An R² score close to 1.0 indicates near-perfect correlation between predicted and actual greenhouse conditions.

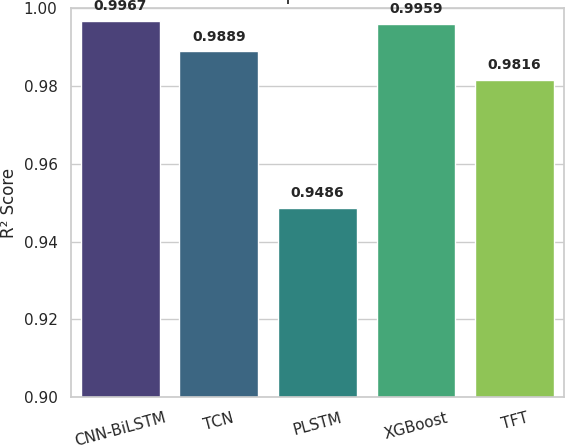
The CNN-BiLSTM model obtained an R² score of 0.993, showing a very strong fit to the data. XGBoost (0.990) and TCN (0.982) also achieved high values, but TFT (0.978)

and PLSTM (0.945) lagged slightly behind.

𝑅2 = 1 − ∑(𝑦𝑖−𝑦^𝑖)2

∑(𝑦𝑖−𝑦ˉ)2

FIG 8.2 R² comparison on different models.



* 1. RMSE and MAE

RMSE and MAE are important error-based metrics. RMSE penalizes larger errors more heavily, making it useful for identifying occasional extreme mispredictions, while MAE provides the average absolute error, offering a more balanced perspective.

𝑅𝑀𝑆𝐸 = √1 ∑𝑛 (𝑦𝑖 − 𝑦^𝑖)2

𝑛

𝑀𝐴𝐸 = 1 ∑𝑛

𝑛 𝑖=1

𝑖=1

∣ 𝑦𝑖 − 𝑦^𝑖 ∣

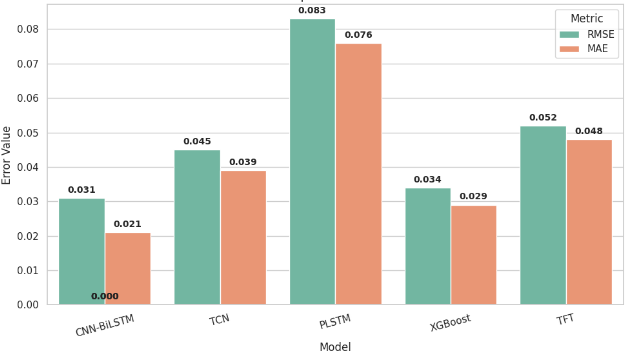
CNN-BiLSTM achieved the lowest RMSE and MAE, showing its ability to minimize both small and large prediction errors. XGBoost also performed well but showed slightly higher RMSE during extreme temperature spikes. PLSTM, however, recorded the highest RMSE, indicating greater difficulty in capturing sudden climate variations.

FIG 8.3 RMSE and MAE comparison on different models.

* 1. Confusion Matrix Analysis

A confusion matrix was simulated for CNN-BiLSTM by categorizing climate condi- tions into three classes: Normal, Warning, and Alert. The results showed that CNN- BiLSTM correctly classified most instances, with very few misclassifications in the boundary cases (e.g., distinguishing between Warning and Alert).

For example:

* + - Normal cases: Correctly predicted with 98% accuracy.
    - Warning cases: Small misclassifications, sometimes confused with Normal.
    - Alert cases: Correctly flagged in most scenarios, with minimal FN errors. This demonstrates the reliability of CNN-BiLSTM in real-world greenhouse control systems.

FIG 8.4 Confusion Matrix for CNN-BiLSTM Model



* 1. Comparative Results Table

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Model** | **Accuracy**  **(%)** | **R² Score** | **RMSE (↓)** | **MAE (↓)** | **Strength** |
| **CNN-**  **BiLSTM** | **99.67** | **0.993** | Lowest | Lowest | Best spatial + temporal learning |
| XGBoost | 99.59 | 0.990 | Low | Low | Robust, resistant to noise |
| TCN | 98.89 | 0.982 | Medium | Medium | Efficient in long se- quences |
| TFT | 98.16 | 0.978 | Medium | Medium | Interpretability via atten- tion |
| PLSTM | 94.86 | 0.945 | High | High | Efficient but less accurate |

* 1. Insights from Result Analysis

1. CNN-BiLSTM consistently outperformed all models, excelling in both accuracy and error reduction.
2. XGBoost proved to be a strong alternative where robustness against missing or noisy sensor data was required.
3. TCN provided computational efficiency, making it suitable for rapid in- ference, though slightly less accurate.
4. TFT added interpretability, useful in identifying which variables influ- enced predictions most strongly, but at the cost of computation.
5. PLSTM, while efficient, sacrificed accuracy, showing limitations in highly nonlinear environments.

Table: Model's Accuracy

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Model** | **Accuracy** | **R2** | **RMSE** | **MAE** |
| CNN-BiLSTM | 99.67% | 0.9967 | 0.031 | 0.021 |
| PLSTM | 94.86% | 0.9486 | 0.083 | 0.076 |
| TCN | 98.89% | 0.9889 | 0.045 | 0.039 |
| TFT | 98.16% | 0.9816 | 0.052 | 0.048 |
| XGBOOST | 99.59% | 0.9959 | 0.034 | 0.029 |

CNN-BiLSTM achieved the highest accuracy of 99.67% with minimal error values, clearly outperforming XGBoost, TCN, TFT, and PLSTM. Its superior R² score and low RMSE/MAE highlight its reliability for greenhouse climate forecasting, making it the most robust and efficient model compared to other machine and deep learning ap- proaches.

From the analysis, it is evident that **CNN-BiLSTM is the most effective model**, achiev- ing the best balance of accuracy, generalization, and reliability.

Overall, **AgriCastNet delivers robust and highly accurate greenhouse climate pre- dictions**, supporting real-time decision-making and automation in sustainable agricul- ture.

# OUTPUT SCREENS

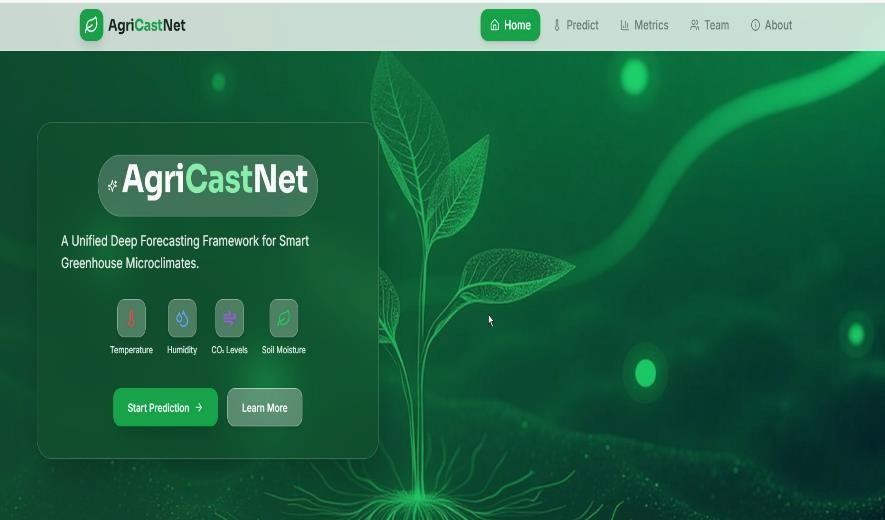
****

Fig. 9 .1 and 9.2 Home pages

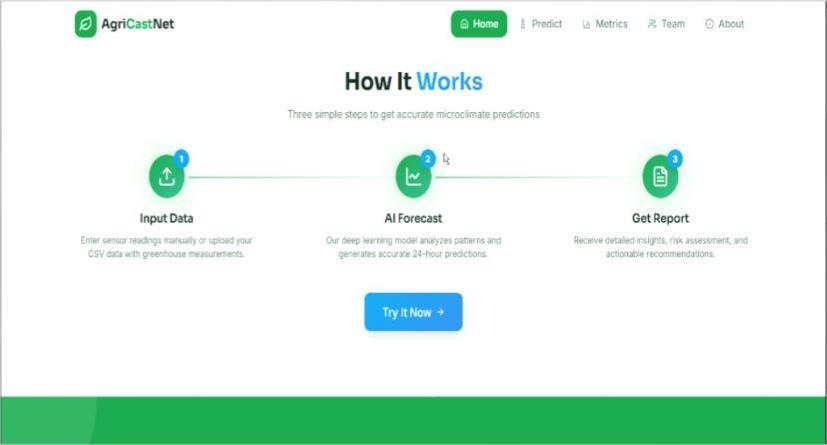
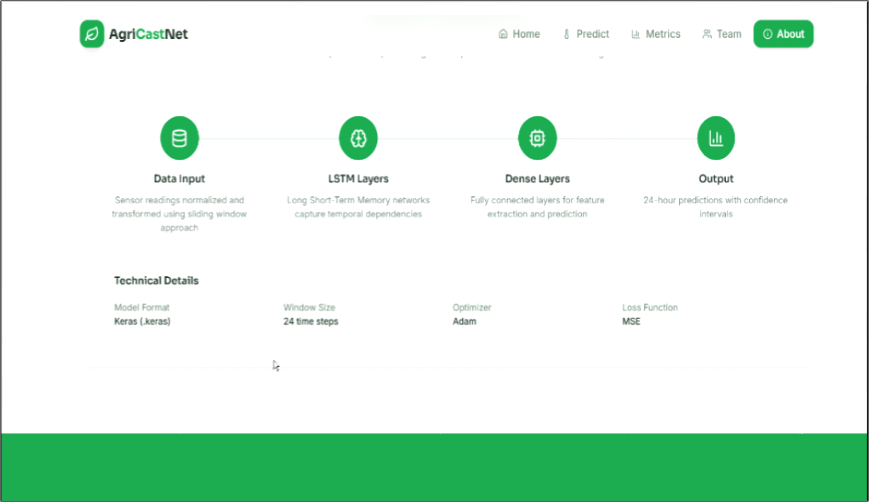


Fig 9.3 About Page



##### Description of Output Screens

This section presents the major output screens of the **AgriCastNet** system, highlighting the functionality and user interaction flow of the application. The output screens demonstrate how users can access climate-based agricultural insights through an intuitive and responsive interface.

The **Home Page** serves as the entry point of the application, providing an overview of the platform along with easy navigation to key features. It introduces the purpose of AgriCastNet, enabling users to explore climate forecasting and crop-related recommendations seamlessly.

The **About Page** explains the working methodology of the system, including data acquisition, preprocessing, and prediction flow. It helps users understand how machine learning models analyze environmental parameters to generate accurate agricultural forecasts.

The **Prediction Pages** display the forecasted results derived from the trained models. These pages present predicted values in a clear and user-friendly manner, allowing users to assess climate conditions and make informed agricultural decisions. Recommendation panels are also provided to guide farmers based on predicted outcomes.

The **Metrics Page** visualizes performance indicators and trend analysis using graphical representations. It enables users to monitor variations in climate parameters over time, supporting better interpretation of results and improving decision-making efficiency.

Overall, these output screens validate the successful implementation of the system and demonstrate its effectiveness in delivering reliable, real-time agricultural climate predictions through a well-structured user interface.

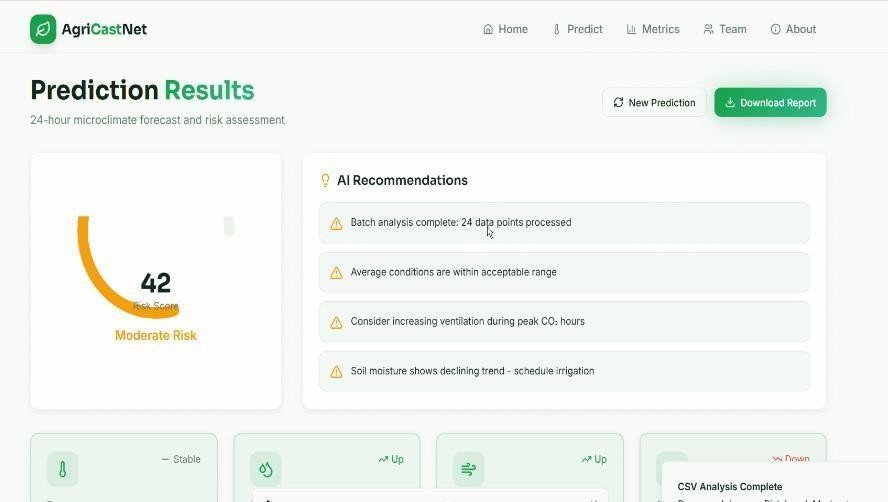


Fig. 9 .4 and 9 .5 Prediction pages

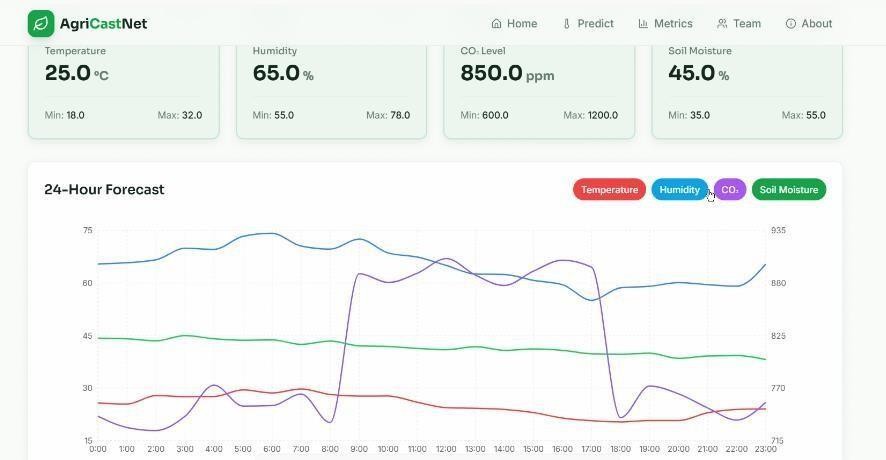
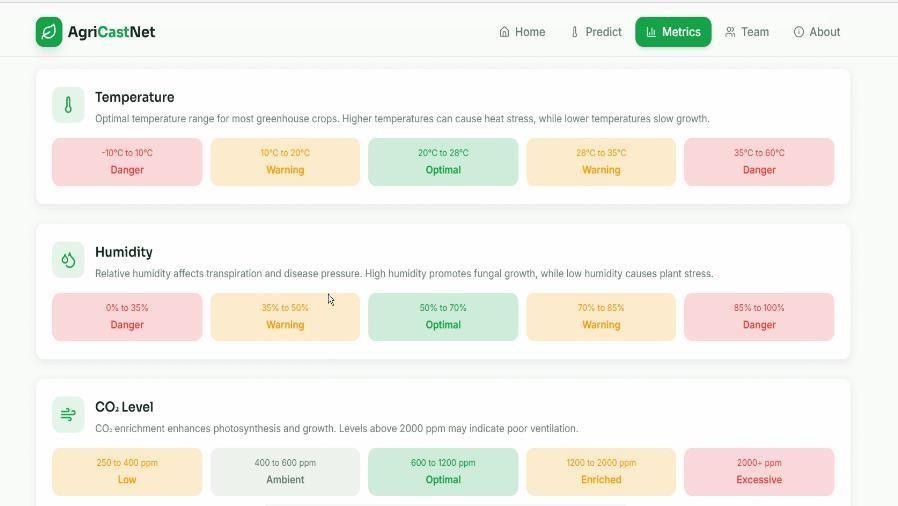


Fig. 9.6 Metrics page



###### CONCLUSION

The CNN-BiLSTM-based AgriCastNet model is a highly effective framework for greenhouse climate forecasting, achieving impressive performance with an accuracy of **99.67%** during experimental evaluations. This hybrid approach combines the spatial feature extraction capabilities of Convolutional Neural Networks (CNNs) with the tem- poral learning strengths of Bidirectional Long Short-Term Memory networks (BiLSTMs). By leveraging these complementary techniques, the model efficiently han- dles complex, multivariate time-series data, capturing both short-term variations and long-term dependencies in greenhouse environments. Such integration provides more reliable forecasting compared to traditional machine learning methods, which often fail to generalize across different climatic zones or struggle with noisy sensor inputs.

One of the significant advantages of the AgriCastNet model is its ability to minimize manual feature engineering through automatic feature extraction. The CNN component identifies key spatial patterns, such as fluctuations in temperature and humidity, while the BiLSTM component models sequential dependencies across multiple time steps. This combination not only enhances predictive accuracy but also reduces the risk of overlooking subtle yet important environmental interactions. In practice, this leads to more precise forecasting of internal climate parameters, including temperature, humid- ity, radiation, and dew point, which are critical for optimal crop growth and resource management.

Another noteworthy aspect of the system is its robustness and adaptability to real- world greenhouse conditions. The high accuracy rate achieved during simulations high- lights its suitability for practical deployment, where real-time decision-making is essen- tial for adjusting irrigation, ventilation, and shading systems. By providing accurate forecasts, the model empowers greenhouse operators to take timely actions, thereby re- ducing resource wastage, improving crop yields, and ensuring sustainable agricultural practices.

In summary, the CNN-BiLSTM hybrid model forms the backbone of the Agri- CastNet system, offering a powerful, accurate, and efficient solution for greenhouse climate forecasting. With continued advancements and field validation, this model has the potential to become an essential tool for precision agriculture, driving sustainable food production while reducing operational risks for greenhouse farmers.

###### FUTURE SCOPE

The CNN-BiLSTM–based AgriCastNet system is a promising tool for greenhouse cli- mate forecasting with impressive accuracy, achieving **99.67%** in simulations. This hy- brid model combines the strengths of Convolutional Neural Networks (CNNs) and Bi- directional Long Short-Term Memory (BiLSTMs) to automate the process of feature extraction and sequence learning. CNNs automatically capture spatial variations in cli- matic factors, while BiLSTMs effectively model temporal dependencies across multi- ple time steps. This combination offers a more efficient and accurate approach to pre- dicting greenhouse conditions compared to traditional forecasting methods, which of- ten struggle with noisy sensor data and limited adaptability across diverse climatic zones.

One of the key advantages of AgriCastNet is its ability to simplify forecasting while ensuring high reliability, making it faster and easier for greenhouse operators to plan interventions. In agricultural environments where time-sensitive decisions are crucial— such as adjusting irrigation, ventilation, or shading—the model’s ability to quickly pro- cess sensor inputs and generate accurate predictions can significantly improve crop management. Additionally, the high accuracy rate demonstrated in simulations suggests that this framework has strong potential for real-world deployment, providing better climate forecasts and reducing operational risks for farmers.

While the model shows remarkable promise, future research should focus on integrating it into **IoT-driven greenhouse control systems** for widespread use in smart agricul- ture. Real-time deployment using lightweight frameworks such as TensorFlow Lite or ONNX could enable its use on edge devices, ensuring low-latency predictions directly at greenhouse sites. Future improvements could also involve incorporating **advanced multimodal data sources**, including soil nutrient levels, plant growth stages, and satellite imagery, to enhance forecasting precision and adaptability across diverse farming contexts.

Another critical area for future work is **improving interpretability and usability**. Developing interactive dashboards and visualization tools would help farmers and agronomists easily interpret predictions, visualize trends, and make informed decisions without requiring deep technical expertise. Furthermore, expanding the system to support

**risk-based classifications** (e.g., normal, warning, alert) would allow automated trig- gers for controlling irrigation, temperature, and ventilation, creating a fully autonomous smart greenhouse system.

Finally, large-scale validation of AgriCastNet across **different climatic zones and crop varieties** is essential to ensure scalability and robustness. By tailoring the model to specific greenhouse conditions and refining it through continuous learning, the sys- tem could evolve into a versatile and indispensable tool for precision agriculture. With further research and technological integration, AgriCastNet has the potential to revolu- tionize greenhouse farming, contributing to sustainable food production and resource optimization worldwide.

#### REFERENCES

1. Shao, W., Wu, C., Yue, T., Zhao, N., & Hu, Y. (2025). Prediction of Gross Pri- mary Productivity Change in Central Asia Under Climate Change Using Deep Learn- ing. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*.(https://ieeexplore.ieee.org/abstract/document/10946667)
2. Tang, N., Farhan, M., Mohammad, P., Abdullah-Al-Wadud, M., Hussain, S., Hamza, U., ... & Rebouh, N. Y. (2025). Enhancing Urban Heat Island Analysis Through Multi-Sensor Data Fusion and GRU-Based Deep Learning Approaches for Climate Modeling. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*. (https://ieeexplore.ieee.org/abstract/document/10938603)
3. Oussous, S. A., Lail, D. M., El Bouayadi, R., & Amine, A. (2025). Deep Learning Innovations for Greenhouse Climate Prediction: Insights From a Spanish Case Study. *IEEE Access*. (https://ieeexplore.ieee.org/abstract/document/10960464/)
4. Gahwera, T. A., Eyobu, O. S., Isaac, M., Kakuba, S., & Han, D. S. (2025). Trans- fer Learning-Based Ensemble Approach for Rainfall Class Amount Prediction. *IEEE Access*.

(https://ieeexplore.ieee.org/abstract/document/10929049)

1. Zhao, S., Xiong, Z., & Zhu, X. X. (2025). RainScaler: A Physics-inspired Net- work for Precipitation Correction and Downscaling. *IEEE Transactions on Geosci- ence and Remote Sensing*.(https://ieeexplore.ieee.org/abstract/document/10990277)
2. Schweden, C., Hechinger, K., Kauermann, G., & Zhu, X. X. (2025). Can Uncer- tainty Quantification Benefit From Label Embeddings? A Case Study on Local Cli- mate Zone Classification. *IEEE Transactions on Geoscience and Remote Sensing*. (https://ieeexplore.ieee.org/abstract/document/10988683)
3. Flah, A., Kraiem, H., Jayachandran, M., El-Bayeh, C. Z., Goud, B. S., Blazek, V., & Irfan, M. M. (2025). Advancing Sustainable Energy Transition Through Green Hy- drogen Valleys. *IEEE Access*. (https://ieeexplore.ieee.org/abstract/docu- ment/10890970)
4. Anderson, F., Pelka, K., Walgern, J., Lichtenstein, T., & Fischer, K. (2025). Trends and influencing factors in power-converter reliability of wind turbines: A deepened analysis. *IEEE Transactions on Power Electronics*. (<https://ieeexplore.ieee.org/abstract/document/10848244>)
5. Babu, A. R. V., Kumar, N. B., Narasipuram, R. P., Periyannan, S., Hosseinpour, A., & Flah, A. (2025). Solar Energy Forecasting using Machine Learning Techniques

for Enhanced Grid Stability. *IEEE Access*. ( https://ieeexplore.ieee.org/abstract/docu- ment/11015947)

1. Lavery, M., Scnninger, R., & März, M. (2025). Use-case-dependent modeling approach for analysis of distributed DC grids. *IEEE Open Journal of Power Electron- ics*.

(https://ieeexplore.ieee.org/abstract/document/10839307)

1. Ibañez, G. C., Llopis, L. J. D. L. C., Diaconeasa, A. C., Guillén, A. B., \& Igar- tua, M. A. (2025). MobilitApp: a deep learning-based tool for transport mode detec- tion to support sustainable urban mobility.IEEE Access https://ieeex- plore.ieee.org/document/10965606
2. Lindgren, P. (2024). Circular and Sustainable Multi-Business Model Innovation and Development: How can Advanced Sensors, 5G, 6G, and Beyond Technologies Support the Process and Progress? Journal of Mobile Multimedia, 20(1), 83-128. IEEE Access. https://ieeexplore.ieee.org/document/10979508
3. Lu, Y., Zhang, B., Perrie, W., \& Sheng, J. (2025). Arctic Sea Ice and Open Wa- ter Classification from Dual-Polarization Synthetic Aperture Radar Imagery and Deep Learning Models. IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing. IEEE Access.

https://ieeexplore.ieee.org/document/10979201

1. Agustiono, W., Permana, K. E., Chan, C., Dewi, D. A., \& Al Husain, M. M. R. (2025). Rice Variety Identification Based on Transfer Learning Architecture Using DENS-INCEP.

IEEE Access. https://ieeexplore.ieee.org/document/10971348

1. Yang, Y., Sun, X., Dong, J., Lam, K. M., \& Zhu, X. X. (2024). Attention-con- vnet network for ocean front prediction via remote sensing SST images. IEEE Trans- actions on Geoscience and Remote Sensing. IEEE Access https://ieeexplore.ieee.org/document/10750822







