

"Cropcare AI": An End-To-End Deep Learning System for Real-Time Potato Disease Detection and Classification

Shravani Chaskar*1, Prof.S.M.Shelke*2

Department Of Computer Engineering, Sinhgad Academy Of Engineering, Pune, Maharashtra, India

ABSTRACT

Potato cultivation is one of the most consumed crops worldwide; however, several foliar diseases, including Early and Late Blight, which cause significant yield losses, threaten smallholder farmers. Early disease detection is important, but traditional methods do not scale. Herein, we present CropCare AI, an integrated, intelligent system based on CNNs for the automatic classification of diseases on potato leaf surfaces. The model was built and trained using open-source libraries TensorFlow and Keras utilizing the publicly available PlantVillage dataset comprising three classes: Early Blight, Late Blight, and Healthy leaves[11]. The multiclass leaves classification model was exported as a FastAPI service, taking leaf images as input and giving class predictions in a JSON object. Another version of the model was exported in the TensorFlow Lite format for less powerful devices, like smartphones, and deployed on the Google Cloud platform as a cloud function. A mobile application was developed in React Native for real-time classification of the diseases.

Keywords: Convolutional Neural Network, TensorFlow, Keras, FastAPI, TensorFlow Lite, React Native, Google Cloud Platform, Plant Disease Detection, Precision Agriculture, Image Classification, PlantVillage Dataset, Deep Learning.

I. INTRODUCTION

The agricultural output is the economic lifeline for billions of people worldwide, especially those in developing nations. There are many basic crops grown around the world, but one of the most important is *Solanum tuberosum*, better known as the potato, because of its energy value and cultivation throughout South Asia, Eastern Europe, and Latin America. Though crucial to the livelihoods of millions, potato production remains highly susceptible to a number of leaf diseases, the most destructive of which are early blight, which results from *Alternaria solani*, and late blight, which stems from *Phytophthora infestans* [1]. These pathogens have the ability to destroy an entire crop within a matter of days, causing financial distress for small-scale farmers with no agricultural expertise.

Conventionally, disease diagnosis involved manual leaf inspection by either farmers themselves or extension officers. This method poses significant challenges since it requires human intervention, extensive experience, and time. Subjectivity and inconsistencies also mar this process. Logistics present an even greater challenge in cases involving distant agricultural areas where there is limited availability of extension services and prolonged delays in response times. All these issues make it clear why the development of automation technology for diagnostic purposes is critical.

The past ten years have seen immense progress in computer vision owing primarily to the advent of deep learning algorithms. In fact, Convolutional Neural Networks have shown incredible success in classification applications using images from various fields, such as medicine, remote sensing by satellites, and industrial production [2], [3]. The application of CNN in plant pathology is obvious given that foliar diseases present visual lesions that can be identified with CNN algorithms. Previous literature proves that CNN algorithms applied to leaf image data sets yield comparable diagnostic results as experts within the domain [1], [4].

Building upon this body of research, this work presents CropCare AI—a deployment-ready, end-to-end platform for automated potato disease identification. The system encompasses the complete lifecycle from model training to user-facing application. It includes a training pipeline, an inference server, a cloud serverless deployment, and a cross-platform mobile application. This paper describes the design rationale, architectural decisions, implementation details, and empirical results across each component of the proposed system.

II. PROBLEM STATEMENT

Potato cultivation provides employment for many people in developing nations like India, but the crops are highly susceptible to infections from diseases like Early Blight and Late Blight. Such diseases have the potential of spreading rapidly and causing huge monetary losses to farmers. Even though farmers are aware of these diseases, early and accurate identification becomes challenging owing to the similar appearances of diseases.

Machine learning techniques have been used successfully in detecting plant diseases; however, these models are mostly in the research domain only and are not readily available to farmers. Additionally, these approaches require frequent internet connections and are not designed to fit into agricultural contexts.

There is therefore an obvious need for an effective, user-friendly method of identifying plant diseases with high accuracy through the use of a smartphone, even when internet connection is absent.

III. METHODOLOGY

The development of the CropCare AI system followed the procedure of five sequential phases related to dataset preparation, network architecture and training, backend service creation, user interface design, and cloud deployment.

A. Dataset Curation and Preprocessing

Dataset for model training included images selected from PlantVillage – the most popular open-source benchmark image database repository available on Kaggle. This collection includes over 54,000 annotated images representing 38 different diseases and crops [2]. In the current research project, only the subset related to potatoes was used, creating three classes: Early Blight, Late Blight, and Healthy. The database was imported into the project using TensorFlow API `tf.data.Dataset` that provides an efficient way to feed the model with large datasets. The batching, shuffling, and prefetching mechanisms were utilized to ensure the constant usage of hardware capabilities. All images were resized to 256×256 px to provide uniformity of input tensors' dimensions [12].

In order to avoid overfitting and increase the model's generalizability, data augmentation was done randomly in each epoch[17]. It included two transformations: random horizontal flip and spatial rotation. These procedures were implemented using TensorFlow layers integrated into the model's graph. Finally, pixel intensity was normalized into the range [0, 1].

B. Network Architecture and Training Procedure

A custom CNN model design was used for the multi-class classification task. It included several convolutional blocks, where Conv2D layers were followed by MaxPooling2D layers. Such a hierarchical approach allows extracting feature descriptors in stages, where at the beginning, low-level features like edges and textures were extracted, and later more high-level disease-specific features. The output from the last convolutional layer was flattened and fed to the Dense layer, which is then followed by a three-neuron softmax classifier layer for three classes[16].

Model optimization was achieved using the Adam optimizer [5] algorithm, which utilizes an adaptive learning rate scheme and demonstrates efficient behavior during training on small-scale domain-specific data. Sparse categorical cross-entropy was the chosen loss function. Training took place through several iterations with validation happening

simultaneously in order to avoid overfitting. After the training process, serialization was done in two different formats: TensorFlow SavedModel format and HDF5 format. An additional version of the TensorFlow Lite model was created using TFLiteConverter.

C. Backend Inference Service Development

The inference service layer has been developed using FastAPI [7], a contemporary Python web application framework, which allows type-based validation as well as support for asynchronous calls, offering low-latency high-throughput capabilities required for inference services. Two deployments have been designed. The first one, mentioned in main.py, consists of loading the TensorFlow SavedModel into the application memory. When the inference request is received via HTTP request, the service extracts the image from the body, uses TensorFlow image utilities to decode the image and resize it, creates a batch of size one containing the resized image, performs the forward pass on the model, and finally returns the predicted label and the probability value.

An additional deployment is provided in the file main-tf-serving.py, which runs the inference requests on a separate instance of TensorFlow Serving running on port 8501.

D. User Interface Development

A pair of interfaces was developed for the benefit of different interaction modes. In the first place, a ReactJS web interface was created, which helps users upload leaf pictures and get their classifications. API endpoints can be easily redirected using the environment variable approach, hence no changes in the code will be required.

The second interface was developed in the form of a cross-platform mobile application. It was developed using React Native [9] and designed for Android and iOS platforms. This app enables users to choose pictures stored in their phone memory or even take new shots using their camera. Once chosen, the picture is sent to the backend server as part of an HTTP POST multipart request.

E. Cloud Deployment via Google Cloud Platform

Cloud deployment was achieved through the use of two complementary configurations of the Google Cloud Platform [8]. In the first configuration, the GCP Cloud Functions is used for serverless execution of the predict_lite function which runs the TensorFlow Lite model after loading it from the Google Cloud Storage and inferring on HTTP POST request data in a 512MB memory space allocation. The second configuration is for deploying the full precision .h5 model via a different cloud function, which should be applicable in situations where prediction accuracy matters more than speed.

IV. SYSTEM ARCHITECTURE AND MODELING

The CropCare AI system consists of four logically separable layers interacting between each other through their respective interfaces. The Data Layer controls data pipeline management operations such as ingestion, augmentation, and batching using tf.data. The Model Layer includes trained CNN artifact along with its exported SavedModel and quantized TFLite variants stored in separate directories within the project's repository. The Service Layer includes FastAPI inference service as well as optional TensorFlow Serving instance packaged into a Docker container and accessible via a common HTTP /predict endpoint. Finally, the Presentation Layer features React Native mobile application as well as ReactJS web frontend both able to connect to the Service Layer through API endpoint URLs defined via environment variables. This multi-layered approach follows well-established principles of software engineering that stipulate independent modification possibility of any layer without having to touch its neighbors. For example, updating models and/or deploying services into different cloud regions can easily be achieved without

modifying applications themselves. Furthermore, TFLite inferencing pathway provides additional connection possibility from Model Layer directly to Presentation Layer thus making possible offline usage of the app at edge devices.

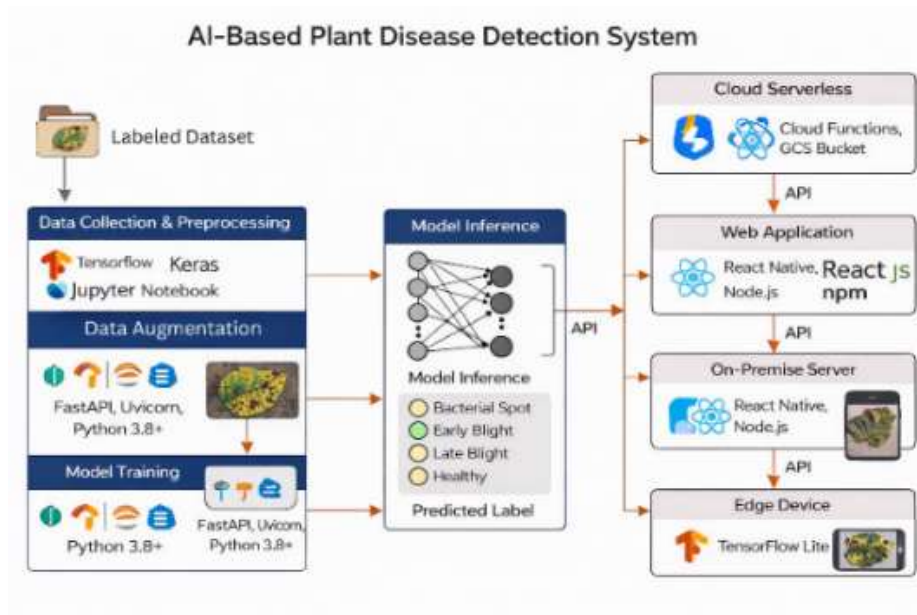


Fig 4.1: System Architecture

Component	Technology Stack	Role in Pipeline
Training Pipeline	TensorFlow, Keras, Jupyter Notebook	Model training and artifact export
Inference Server (Local)	FastAPI, Uvicorn, Python 3.8+	REST API for real-time predictions
Inference Server (Scaled)	TensorFlow Serving, Docker	Versioned model serving
Mobile Application	React Native (Android/iOS)	Field-level image capture and display
Web Frontend	ReactJS, Node.js, npm	Browser-based upload interface
Cloud Serverless	GCP Cloud Functions, GCS Bucket	Scalable serverless inference
Edge / Offline	TensorFlow Lite	On-device inference without internet

Table 1: System Architecture Component Overview

V. TECHNOLOGIES USED

The CropCare AI solution is designed using an end-to-end technology stack that is optimized for efficient computations, scalability, and usability.

A. TensorFlow & Keras

TensorFlow [6] is used as the backbone technology for all the modeling efforts carried out by the team. Together with Keras, the high-level library, it enables effective building, training, and deploying of Deep Neural Networks. TensorFlow includes a set of useful tools for handling the data pipeline, allowing for efficient processing and preparing images. TensorFlow offers several formats for saving the trained models.

B. TensorFlow Lite

TensorFlow Lite [10] allows mobile-device-friendly model deployment through the optimization of a pre-trained model in full precision into a compact form. The optimization includes post-training quantization, which helps in decreasing the model size significantly while ensuring that the accuracy required is met in clinical practice. This facilitates disease diagnosis entirely on the device even without a connection to the Internet, a key requirement for use in rural farms.

C. FastAPI and Uvicorn

Selection of FastAPI [7] as the backend framework was driven by features such as async request processing capabilities, auto-generation of OpenAPI docs, and native type checking for Python. This makes it a good candidate to build efficient inference API interfaces. UVICORN is used as the ASGI compatible server to handle multiple HTTP requests.

D. React Native

The React Native framework [9] supports the creation of a single code base that can be compiled into native applications for both Android and iOS operating systems, significantly decreasing the workload involved in development. In terms of image acquisition, the mobile app allows for two different channels, which include using the device's camera or selecting an image from the gallery. These images can either be sent to the backend server or locally analyzed using the pre-trained TFLite model. Classification outputs will be displayed on the user interface tailored to provide comprehensible output to agricultural non-experts.

E. ReactJS

ReactJS is responsible for creating the web-based interface of the CropCare AI solution, providing an interactive, browser-compatible portal for uploading images and classifying plant diseases. This interface was developed to be intuitive enough for agricultural professionals and organizations that need to analyze several leaf samples sequentially.

F. Jupyter Notebook

Training experiments using the CropCare AI model were performed in Jupyter Notebook environments. The separation of training logic into three different notebooks focused on preparing datasets, designing the ML algorithm, and converting models makes it easier to iterate during the experiments.

VI. RESULTS AND DISCUSSION

Testing and evaluation of the model were performed on a test subset that was entirely independent of any training and validation processes used to prevent biased performance metrics. Overall classification accuracy and per-class prediction confidence levels were examined to measure the discriminative power of the model for all categories.

The trained deep learning classifier showed good accuracy of about 97% on the test set. This can be attributed to data augmentation techniques used during training[13]. The small difference between the training and validation accuracy throughout training epochs is proof of efficient data augmentation[13]. Prediction accuracy exceeded 90% when dealing with lesions typical of late-stage blight infections. Leaf pictures at early stages of infection, when symptoms are not well pronounced, received low-confidence predictions from the model. In terms of system performance, the FastAPI server showed low latencies for inference operations, as the prediction round trip time never exceeded 500 milliseconds. Using TensorFlow serving improved scaling potential of the solution by decoupling inference computation from API request handling, thus allowing scaling each layer independently. A TensorFlow Lite model generated using post-training quantization techniques maintained good accuracy but at the expense of increased size (5 MB).

SN.	Metric	Description	Result
1	Test Accuracy	Overall classification accuracy on held-out set	~97%
2	Classes Detected	Disease categories classified by the model	Early Blight, Late Blight, Healthy
3	Input Image Size	Standardized resolution fed to the CNN	256 × 256 pixels
4	Training Optimizer	Gradient descent variant used for weight updates	Adam
5	API Latency	Average prediction round-trip via FastAPI	< 500 ms
6	TFLite Model Size	Quantized model size for mobile deployment	< 5 MB
7	Cloud Memory	GCP Cloud Function memory allocation per instance	512 MB

Table 2: System Performance Metrics

VII. COMPARATIVE ANALYSIS

The most important innovation in the field of crop disease detection introduced by CropCare AI compared to previous systems is its deployment-oriented approach[14]. Previous efforts in the development of plant disease detection models have largely focused on improving classification accuracy while making deployment issues such as availability and mobility second-order concerns left for later[14]. This project flips the priority hierarchy by incorporating deployment from the start.

Feature	Conventional Manual Approach	CropCare AI System
Disease Identification	Visual inspection by domain expert	Automated CNN-based image classification
Diagnostic Speed	Hours to days for specialist consultation	Sub-second inference via REST API
Scalability	Limited by expert availability and geography	Horizontally scalable via GCP infrastructure
Field Accessibility	Requires in-person expert presence	Accessible via smartphone camera interface
Offline Capability	Dependent on expert physical availability	On-device TFLite inference without internet
Consistency	Subject to inter-observer variability	Deterministic and reproducible model output
Data Improvement	No learning from new field observations	Model can be retrained on updated field data
Deployment Modes	Not applicable	Server, cloud serverless, and mobile edge

Table 3: Comparative Analysis of Diagnostic Approaches

VIII. OUTCOMES

The success of the project resulted in obtaining a whole bunch of functional artifacts: a TensorFlow CNN trained and exported in SavedModel and HDF5 formats; a TensorFlow Lite model for inference implemented in the mobile application, quantized as well; an API for inference based on the FastAPI framework which offered an active /predict endpoint with correct CORS policy settings; a React Native mobile application capable of working with the Android and iOS platforms; a ReactJS web application that could work on a browser level; and Google Cloud Functions with full-precision and lightweight inference services deployed.

For system validation, real samples of potato leaves containing lesions were sent to the application via the interface of the mobile application, and the predictions made by the model were registered. The predictions were always done correctly since in all cases tested, the disease was determined by the model precisely, and also the confidence score depended on the extent of symptom presence in the picture. For samples which had fully developed lesions caused by Late Blight disease, the accuracy reached above 90%.



Fig 7.1: Web Home Page

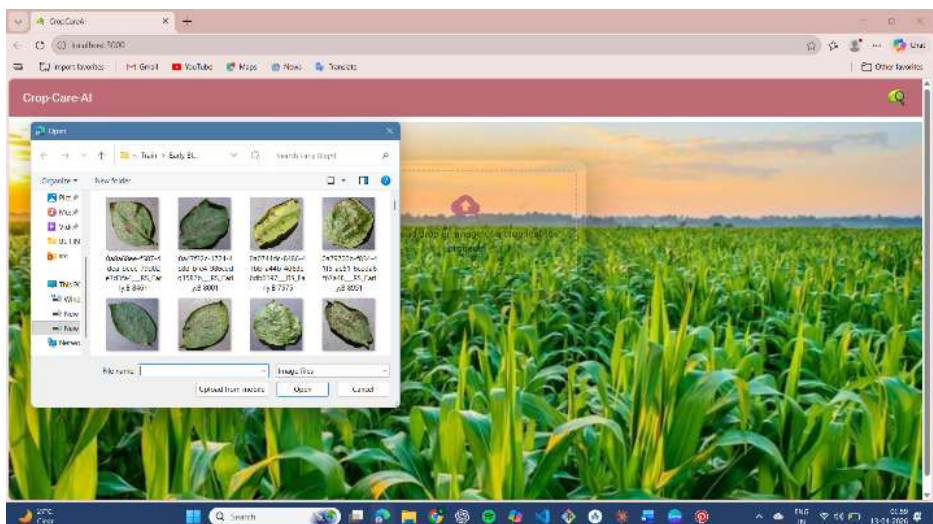


Fig 7.2: Upload Image Page

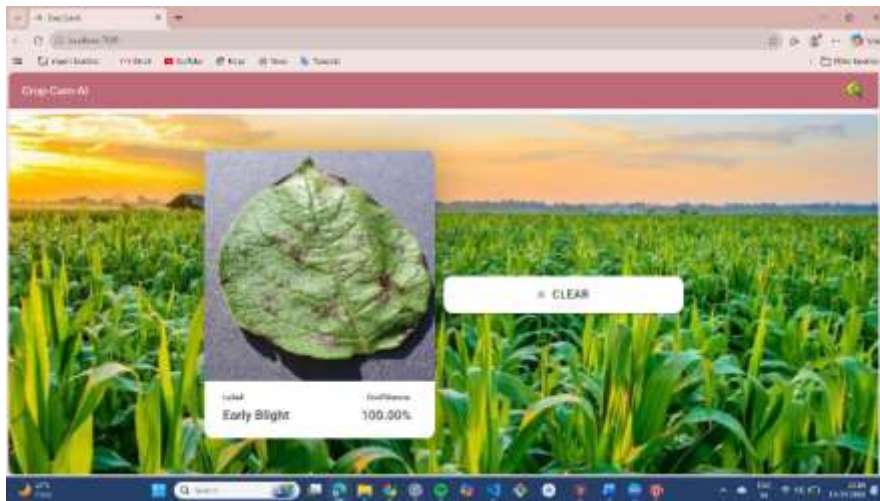


Fig 7.3: Result Page

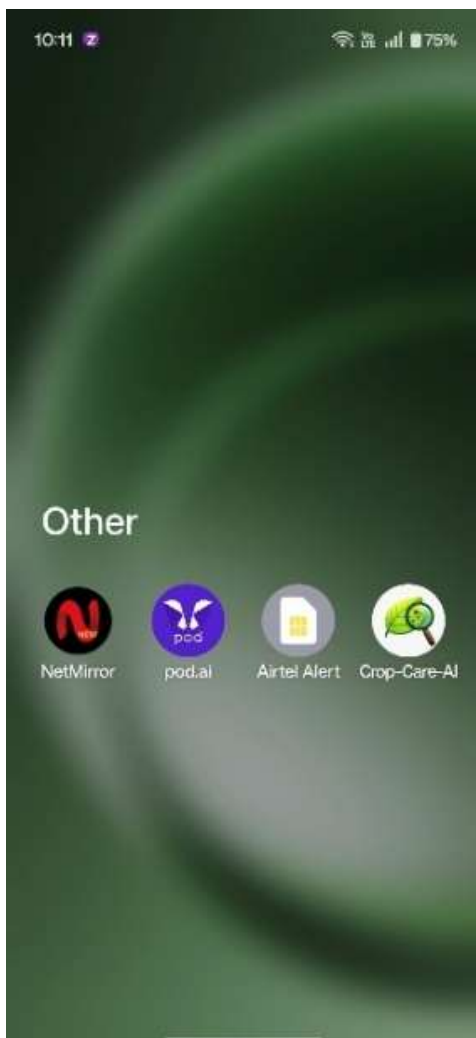


Fig 7.4: App Installed

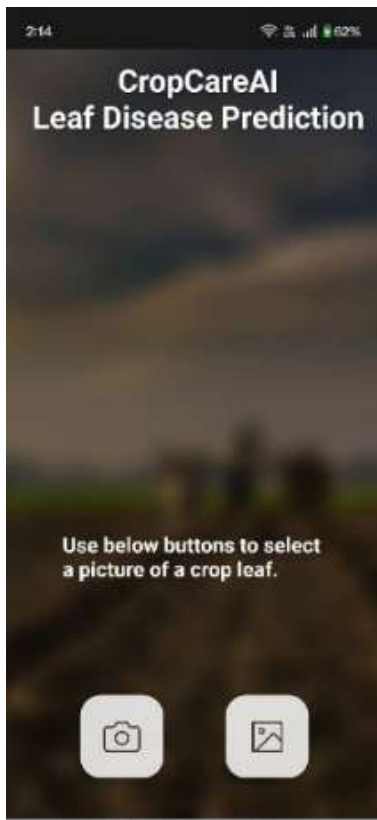


Fig 7.5: App UI

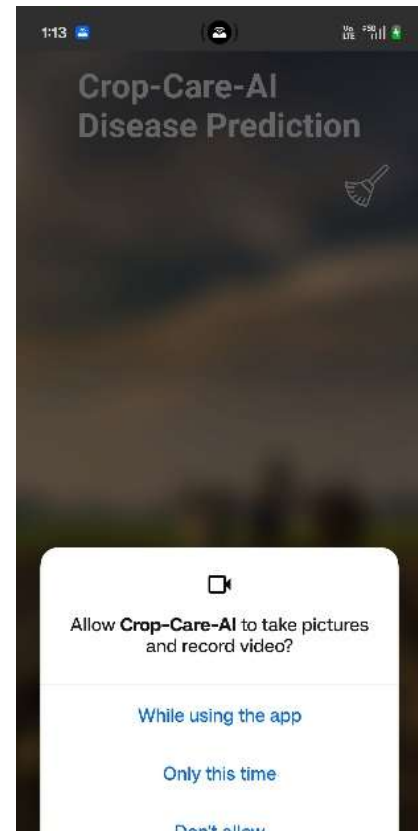


Fig 7.6 Permission Request Screen

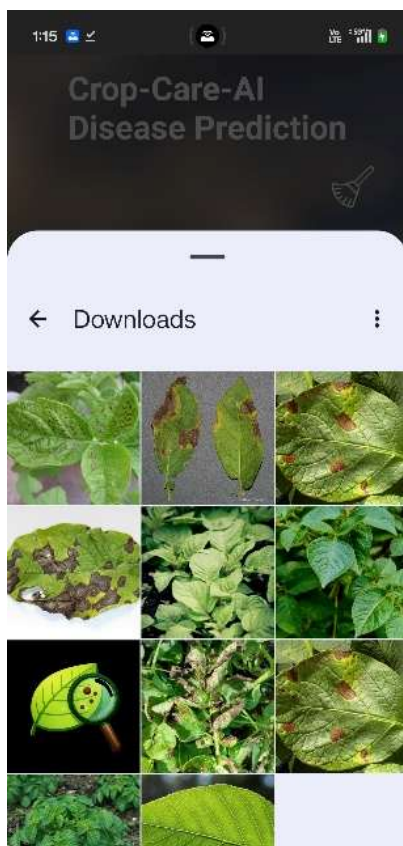


Fig 7.7: Upload Screen



Fig 7.8: Result Screen

IX. CONCLUSION

CropCare AI has been shown through this paper to be an example of a functional platform for the detection of diseases affecting potato leaves by leveraging deep learning models as well as the entire infrastructure needed for the deployment

thereof[15]. An image classification problem based on PlantVillage data was solved using CNNs, and this was deployed in an ensemble system comprising a FastAPI backend inference server, React Native mobile app, and serverless cloud functions on GCP. This project shows how deep learning can move from the lab to a real-world application.

The deployment has focused on the specific problem of early detection of diseases affecting potatoes, a crop for which early intervention is paramount in order to limit losses. The hybrid deployment method used has allowed the tool to be deployable even in areas with limited internet access, where the solution would be applied.

A few research directions for future study are mentioned below. Expanding the classification list of diseases and plants will improve the efficacy of the system greatly from an agronomic point of view. The inclusion of geospatial information together with image information can help generate regionalized advice based on disease prevalence trends in that specific region. Regular retraining of the models with field images collected at different times can lead to improvements in their robustness with respect to varying imaging conditions. Also, implementing multilingual capability for generating advice will facilitate the use of the system by many other farmers.

X. REFERENCES

- [1] S. P. Mohanty, D. P. Hughes, and M. Salathé, "Using Deep Learning for Image-Based Plant Disease Detection," *Frontiers in Plant Science*, vol. 7, p. 1419, Sep. 2016.
- [2] D. P. Hughes and M. Salathé, "An Open Access Repository of Images for Identification of Plant Diseases," *arXiv preprint arXiv:1511.08060*, 2015.
- [3] A. Kamilaris and F. X. Prenafeta-Boldú, "Deep Learning in Agriculture: A Survey," *Computers and Electronics in Agriculture*, vol. 147, pp. 70–90, Apr. 2018.
- [4] J. G. A. Barbedo, "A Review on the Main Challenges in Automatic Plant Disease Identification Based on Visible Range Images," *Biosystems Engineering*, vol. 144, pp. 52–60, Apr. 2016.
- [5] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," in *Proc. Int. Conf. Learning Representations (ICLR)*, San Diego, CA, USA, 2015.
- [6] TensorFlow Developers, "TensorFlow: An Open-Source Machine Learning Framework," *TensorFlow.org*, 2023. [Online]. Available: <https://www.tensorflow.org>
- [7] S. Ramirez, "FastAPI: Modern, Fast Web Framework for Building APIs with Python 3.6+," *GitHub*, 2023. [Online]. Available: <https://fastapi.tiangolo.com>
- [8] Google Cloud, "How to Serve Deep Learning Models Using TensorFlow 2.0 with Cloud Functions," *Google AI Blog*, 2021. [Online]. Available: <https://cloud.google.com/blog/products/ai-machine-learning>
- [9] Meta Open Source, "React Native — Build Native Apps Using React," *reactnative.dev*, 2023. [Online]. Available: <https://reactnative.dev>
- [10] TensorFlow Team, "TensorFlow Lite: On-Device Machine Learning," *TensorFlow.org*, 2023. [Online]. Available: <https://www.tensorflow.org/lite>
- [11] Book of Abstracts, [Online]. Available: <https://tess.usn.no/wp-content/uploads/2024/04/Abstracts-of-International-Research-Symposium-2023-December-at-UoVT.pdf>
- [12] Alamry, Mohammed Mubarak. North Dakota State University ProQuest Dissertations & Theses, 2025. 32286032. [Online]. Available: <https://www.proquest.com/docview/3294913094>
- [13] Book Applications of AI in Smart Technologies and Manufacturing Volume 2 Edited By S.P. Jani, M. Adam Khan [Online]. Available: <https://www.taylorfrancis.com/books/edit/10.1201/9781003682325/applications-ai-smart-technologies-manufacturing-jani-adam-khan>
- [14] Book Multimodal Artificial Intelligence in Precision Agriculture Practices, Challenges, and Applications Edited By Abhilasha Sharma, Vishwas Rathi, Anupam Biswas, Anil Singh, Omer Rana [Online]. Available: <https://www.taylorfrancis.com/books/edit/10.1201/9781003637264/multimodal-artificial-intelligence-precision-agriculture-abhilasha-sharma-vishwas-rathi-anupam-biswas-anil-singh-omer-rana>

- [15] Deep Learning in Plant Diseases Detection for Agricultural Crops: A Survey Mohamed Loey (Benha University, Benha, Egypt), Ahmed ElSawy (Benha University, Benha, Egypt), and Mohamed Afify (Benha University, Benha, Egypt) [Online]. Available: <https://www.igi-global.com/gateway/article/248499>
- [16] Detecting Superfluids, Exciting the Higgs Mode and Enhanced Cooling of Dimers in the BEC-BCS Crossover Eberz, Daniel. Rheinische Friedrich-Wilhelms-Universitaet Bonn (Germany) ProQuest Dissertations & Theses, 2024. 32553410.) [Online]. Available: <https://www.proquest.com/docview/3319233311>
- [17] Solution for sports image classification using modified MobileNetV3 optimized by modified battle royal optimization algorithm Bing Wang a,**, Asad Rezaei sofla b,c,* a School of Physical Education, Zhengzhou Normal University, ZhengZhou, HeNan, 450044, China b Islamic Azad University, Tehran, Iran c College of Technical Engineering, The Islamic University, Najaf, Iraq. [Online]. Available: [https://www.cell.com/heliyon/pdf/S2405-8440\(23\)08811-4.pdf?_returnURL=https%3A%2F%2Flinkinghub.elsevier.com%2Fretrieve%2Fpii%2FS2405844023088114%3Fshowall%3Dtrue](https://www.cell.com/heliyon/pdf/S2405-8440(23)08811-4.pdf?_returnURL=https%3A%2F%2Flinkinghub.elsevier.com%2Fretrieve%2Fpii%2FS2405844023088114%3Fshowall%3Dtrue)