Research Article

Cotton Disease Detection Using TensorFlow Machine Learning Technique

Sandeep Kumar, Rajeev Ratan, and J. V. Desai

MVN University, Palwal, India

Correspondence should be addressed to Sandeep Kumar; yadav.san20061987@gmail.com

Received 8 June 2022; Revised 6 July 2022; Accepted 25 July 2022; Published 24 August 2022

Agriculture is a main source of income for farmers in India. Farmers produce many seasonal local crops based on their location. Cotton is the most produced crop across India. Cotton is a commercial crop, and farmers get good capital from cotton. This will increase the income of the farmer. However, one of the basic problems with cotton is that it is easily exposed to many diseases. These diseases need to be identified as early as possible to avoid production loss. In this paper, the CNN algorithm is used to create the prediction model by leveraging the TensorFlow’s Keras API. This model is further used in mobile app development which helps the farmers identify cotton disease and recommend the pesticides which can be used to overcome the disease. The TensorFlow open-source platform was used to prepare the ML model. The TensorFlow TFLite model is created, and after that, the model is converted into the Core ML model, which is used in iOS app to make the disease predication. Google’s core API is used to convert the TensorFlow model into the Core ML model. The label dataset was used to create the model. The Swift language is used in app development. The model accuracy was around 90%. Currently, boll rot and fungal leafspot disease are detected in this app. However, the app can be further extended for other cotton diseases too.

1. Introduction

In India, the agriculture is done traditionally. The use of technology in agriculture is far distant yet. The innovation in the technology sector, specifically in IT-based sector, is far beyond the limit. In the last decade, the AI, IoT [1] etc., have shown tremendous growth, and it is changing the world. The usage of IoT and AI can also be extended to the agriculture domain for smart e-agriculture. In the field of IoT, the drones are doing very well. In the field of agriculture, drones can be used for multipurpose. The drones are used to spray the pesticide in the crop that helps farmers not to be exposed to any diseases. Additionally, with the help of drones, the damage to crops can be minimised. The drones can also be used for surveillance of the crop from wild animals. In the field of water management, the IoT is plays significant role. The SWAMP [2] and EEWMP [3] projects are an example of water efficient agriculture. A lot of water is saved using these models, which helps in keeping the water level at maximum. These models also help in measuring the soil moisture and then providing the required water to the crop based on the soil moisture of that crop area.

Similarly, the role of AI and ML in agriculture is essential. ML algorithm–based applications or tools are used in different areas of agriculture. AI-based robots are widely used to collect fruits or any other products from the crop field. This saves a lot of workforce, money, and time. In precision agriculture, the AI and IoT [4, 5] helps the farmers to improve the overall harvest quality and accuracy. With the help of sensors, weeds can be targeted and detected and then decide which herbicide to apply to that region. In the region of the supply chain, many AI applications help in the smooth operation.

The major challenge is in the broad adoption of AI applications due to the lack of simple solutions and seamless operation of the tools and applications. This problem restricts the use of technology in the agriculture domain. The motivation behind this research is to use the AI and technology in the agriculture field because AI and technology have huge
potentials to bring revolution in agriculture sector like usage of drones, etc. This research paper is a step towards solving the farmer’s problem of doing the farming manually, which takes more time and workforce. The outcome of this research paper will solve the cotton disease detection problem with higher accuracy in less time. This will help in higher production of cotton. In this paper, a mobile app is developed using the ML-based model to detect the cotton plant disease. The mobile application is very simple to use and does not have complex UI/UX. The farmers can easily use the app to detect the cotton plant disease [6]. The model is prepared offline using the TensorFlow open-source Google framework, and then this model is embedded in the project. The app can work offline without an Internet connection in the device. For the model preparation, CNN-based image classification APIs are used. The advantage of the TensorFlow framework over the other frameworks used in the literature work is that it has a very simple implementation using the Keras library API’s.

The use of this application is beneficial for farmers to early detect the disease. This will also help the farmers to get the proactive action if the crop is exposed to the disease, then, based on the app recommendations, the farmers use that pesticides to overcome the disease. Now, with this, the farmers can get the better crop production and post-harvesting the crop, and the farmers can also get the good capital from the good quality crop. Table 1 shows the comparative study in a literature review.

2. Related Work

Good quality fruits are produced using the trading technique. Apple fruits are being traded to get the best variety and is exported at high price [12]. Similarly, bacterial wilt has also been studied [13] for eggplant and its impact on the crop. It helps in understanding the importance of the disease-free crop and to get the maximum output. In similar objectives, the crop rotation policy also helps [14]. The benefit of crop rotation is that the soil is not getting exhausted with similar types of nutrients in the soil, and it helps to maintain a good balance of the different nutrients in the soil.

3. Methodology: Algorithm and ML Workflow

A comparative study has been conducted on various frameworks, which can be used for model creation, and this model further can be used in the iOS project [15].

3.1. CNN: Image Classification Algorithm. TensorFlow is a well-known deep learning framework, and Keras is its official high-level API. TensorFlow compiles different algorithms and models together, enabling us to implement a deep learning neural network for the use in task like image recognition/classification and natural language processing.

Keras was designed with modularity and user-friendliness. Keras make implementing the many powerful but complex functions of the TensorFlow as simple as possible. It is designed to work with the Python language.

The advantage of the TensorFlow framework over the other frameworks used in the literature work is that it has a very simple implementation using the Keras library API’s. The TensorFlow framework supports many programming

---

**Table 1: Comparative study in a literature review.**

| Authors/year | Work detail | Algorithm/technique | Platform | Findings |
|--------------|-------------|---------------------|----------|----------|
| K.P. Ferentinos [8]—2018 | Using an opensource dataset, the deep learning model is prepared. Model is used to detect the diseases Python language is used | CNN | Android/Web | Android and web-based apps to identify the disease Server-based implementation The server set up is costly and need maintenance |
| A. Shah, P. Gupta, and Y.M Aigar [9]—2018 | Automatic solution for nutrient deficiency identification Data set of deficient and healthy leaves is used RGB colour pattern is extracted | OpenCV Supervised ML | Web | Image feature extraction logic Serve-based implementation The server set up is costly and need maintenance |
| J. Shirahatti, P. Patil, and P. Akulwar [10]—2018 | Studied different kinds of ML techniques Accuracy result comparison is performed Image data set is used for comparison | SVM, PNN, ANN, GA | Web | Comparative study with results |
| K.K. Singh [11]—2018 | System to track, diagnose and forecast the disease information Local expert advice can be taken Geo tagging on images Image dataset is used | CNN | Web | Cloud-based integration Interface to upload the image and see the result High accuracy ~ 95% Local expert opinion Cloud-based implementation is costly and need maintenance |
languages, which makes the framework popular. Also, Google developed this framework, so it is maintained with the latest ML algorithms and libraries.

3.1.1. Image Classification. Image classification is supervised learning techniques. It defines the set of target classes and trains a model to recognize them using the labelled images, the label that the network output corresponds to a predefined class. If there is a single class, then it is called image recognition, whereas a multiclass recognition task is called image classification.

3.1.2. Feature Extraction. Features are the elements of the data which will be fed through the network. In some special image recognition, the features are the group of pixels, like edges and points, of an object that the network will analyse for the pattern.

Feature recognition is the process of pulling the relevant features from the input image so that these features can be analysed. The process of extracting features from the image is accomplished with a convolutional layer, and this layer makes the representational part of image. The result of all these calculations is a feature map. This process is typically performed with more than one filter, which helps preserve the complexity of the image.

3.1.3. Activation Function. After the feature map of the image is created, the values of that represent the image are passed through the activation layer. This layer makes the nonlinearity of the images since they images are already nonlinear. Mostly, rectified linear unit (ReLU) activation function was used.

3.1.4. Pooling Layers. This layer makes the image smaller by taking the information that represents and compresses it. The pooling process makes the network more flexible and more adept at recognizing the images based on the relevant features. This layer in a CNN will abstract away the necessary parts of the image, keeping the part of image, which is more relevant. This helps in overfitting the model where the network learns aspects of the training case too well and fails to generalize to new data.

3.1.5. Flattening. This is the final layer of the CNN, the densely connected layers, which requires the data in the form of a vector to be compressed. The values are compressed into a long vector or column of sequentially ordered numbers.

3.2. Machine Learning Workflow. We identified the image classification algorithm which is used in the machine learning workflow to create and train the model. The process is divided into four different phases.

3.2.1. Data Set Preparation. Cotton disease images were collected from an actual cotton crop field under different weather conditions. It adds the variety to the dataset, which is good for better model creation. The images are labelled as per their disease characteristics and placed in the respective folder. The operations and data were prepared manually.

Figure 1 shows the structure of the labelled data, which is used in model preparation. Three label classes are used, i.e., healthy, fungal, leafspot, and boll rot. The number of images in each image class are kept almost same to achieve better model accuracy.

Totally, 825 images were used to prepare the model. The ratio of the training data set and testing data set is kept 80:20.

3.2.2. TensorFlow Model Creation. TensorFlow lite is an open-source machine learning framework to perform on the device. Some key features of the TensorFlow lite are listed as follows [16]:

(i) Optimized for on-device performance:
   - No round trip to server—no latency
   - Internet connectivity not required
   - Reduced model size
   - Low device power consumption
   - Privacy—no personal data leaves the device

(ii) Multiplatform support like iOS, Android, Linux, and microcontroller

(iii) Multiprogramming language support like Java, Swift, Objective C, C++, and Python

(iv) High performance with device hardware

For app development, an image classification type model is prepared. The TensorFlow lite model maker library was used to prepare the Tflite model with custom dataset. A custom dataset was used for model creation. The model maker library makes the process simple with a custom dataset to create a new model. It uses the transfer learning technique to reduce the training time and the amount of training data required. Google’s Colab environment [17] is used to write the code to make the model. A PIP installation package manager is used to install the required dependencies and libraries.

Figure 2 shows the details of model maker dependency installation. The PIP command [18] installs all required dependencies and libraries, which are required to create the new TensorFlow lite model with a custom dataset. It leverages the advantage of Google’s Colab online editor and saves the code on the Google drive, and the same code can be accessed any time later when required. Once the above installation command is successfully executed, then the dependent libraries or module can be imported in script.

Figure 3 has the details of the submodule which needs to be imported to create the model. The open-cv dependency is required to check the valid image format in the dataset, and if any of the image is in not in allowed format, then that image is deleted from the dataset. This helps in preventing the getting the error if the format of any of the image is invalid and model creation is failed.

Figure 4 shows the code snippet of TensorFlow model creation. The Tflite extension model is created.
For this model creation, the dataset zip file is uploaded to the Google Drive, and permission is given to access the dataset by the Colab editor. The zip file contains the labelled training data. The Keras utils module extracts the zip dataset file from the given file path and stores the dataset in the temp folder for further processing. Now, the exact dataset root path was prepared by joining the dataset folder name.

To remove the invalid images from the dataset, the allowed image extension names are taken, i.e., jpg, jpeg, and png. The “check_images” function checks the all-dataset images, and whenever any invalid image is found, then that image will be deleted. Thus, after this function call, the remaining images in the dataset are valid. Then, the data loader module loads the data from the given folder path. These data are now split into train_data and test_data.

The “image_classifier” object calls the “create” function to create the TensorFlow model on “train_data.” This process creates the Tflite extension model.

### 3.2.3. Model Training

Once we have the model instance, we must fit the data. The complete process takes some time as the dataset is getting trained. Generally, the longer we train the model, the performance is improved, but too many training epochs risk overfitting the model. Therefore, we should choose the value of the training epochs very wisely.

Figure 5 shows the TensorFlow model parameters which is internally used while training the model. As shown in the above figure, three label classes are used in the model, i.e., boll rot, fungal leafspot, and healthy. It also shows the trainable and nontrainable parameters.

### 3.2.4. Model Evaluation

Now, the model is further evaluated with test_data, and it will analyse the accuracy and loss. Once all these done, then the model is exported using the “export” function, and the Tflite extension model is saved in the given directory path. The accuracy metrics of the model are analysed and based on these metrics some tweaks can be done to improve the metric data. Figure 6 shows the model characteristics while training the model. The model was trained for five iterations. The model is lightweight, and size is 4 MB. This model is used for cross platform development.

### 3.2.5. CoreML Model Conversion

However, the Tflite extension model cannot be used directly in the iOS project to develop the app. This model must further convert into CoreML model using the TensorFlow convertor.

Figure 7 shows the code snippet for converting the TensorFlow Tflite model into a CoreML model [19]. To convert the model into the CoreML model, the tfscoreml module is required. All parameters are prepared, and then the parameters are passed in “convert” function, which takes all the required parameters and converts the model into a CoreML model. This ML model is further used in the development of the iOS app to detect cotton diseases. The TensorFlow library internally leverages many advantages from other dependent libraries such as Keras, NumPy, and Matplotlib, while creating a new model or retraining the existing model with a custom dataset. This makes the library more flexible and can be used for different purposes. It also gives an option to use other algorithms for better performance and accuracy.

Now, finally, the converted CoreML model is used in the iOS project for app development. CoreML framework APIs are used to interact with the model to detect the cotton leaf disease. The result of the disease is shown to the farmer in the app. This model is lightweight and easily integrated into the app.
4. App Development

4.1. Project Setup. The iOS mobile app development is done on the Mac environment using the XCode IDE. The development was done in Swift programming language. Swift is an open-source programming language for the iOS platform powered by Apple Inc. The app is designed specifically for iPhone devices, but the app can be used on the iPad too. For local storage of the disease data, Core Data [20] local database is used in the app.

4.2. App Architecture: VIPER. The app is based on the VIPER [21] mobile design patterns, which has some advantages compared to other design patterns. This design pattern is based on five layers named as interactor, view, presenter, router, and entity. These five layers have different individual responsibilities, which follows the principle of separation of concerns. According to the separation of concern principle, each layer has a separate responsibility, and no layer may have a common responsibility. This makes the layers easily mockable while writing the test cases. These layers are loosely coupled, and wherever required, any layer can be removed or modified at any time, which makes the app more scalable.

According to this architecture, the responsibility of the view layer is to render the UI components and make the screen ready for the user. The presenter handles all the logic related to the view, which is visible to the user like by receiving the touch, button click etc., and passes it to the next layer if required. The interactor layer handles all business logic, data manipulations, algorithms, etc., in the app. The router layer handles the animation in the app during the navigation. The last entity layer binds the data either from the server or from the local database and prepares the model which is used further in the app.

4.3. App Modules

4.3.1. Inspect Disease. This is the home screen or landing screen in the app. When an app is launched, then this screen is visible to the farmers. Figure 8 shows the inspect disease screen, where the farmer selects the cotton disease image, and with the selected disease images, the farmer can check the disease. When an image is unselected and the “Check Disease” button is clicked, then the app will show the pop-up message, asking the farmer to select the cotton image. When an image is selected and then clicked on the “Check Disease” button, then the app redirects the farmer to the next screen. On the next screen, the farmer can see the

```python
image_path = tf.keras.utils.get_file('CottonDiseasesTraining.zip', 'file:///content/drive/MyDrive/CottonDiseasesTraining.zip', extract=True)

source_dir = image_path

good_exts=['.jpg', '.jpeg', '.png', '.PNG'] # list of acceptable extensions
bad_file_list, bad_ext_list=check_images(source_dir, good_exts)

if len(bad_file_list) >0:
    print('Improper image files are listed below')

print(bad_file_list)
print(bad_ext_list)

model = image_classifier.create(train_data)
loss, accuracy = model.evaluate(test_data)
model.export(export_dir='')
```

Figure 4: Model creation code snippet.
disease details, if any, or the app shows that the cotton image is healthy.

Figure 9 is the disease detail screen. Machine learning model prediction is displayed on this screen. The higher confidence label is shown on the top and then followed by the lower confidence label. The recommendations about the disease are also shown on the same screen. This will help the farmer to take the proactive action against the disease. With the help of this, the crop production is increased, and farmers will get good income from the cotton production.

The “Save” button is on the top right of the navigation bar. When clicked on this button, all crop details such as image and disease detail are saved in the local database on the device. This will help the farmer to check the disease history.

4.3.2. History Module. This module shows the saved disease details. The data are obtained from the device’s local database.

4.3.2. History Module. This module shows the saved disease details. The data are obtained from the device’s local database.

Figure 10 shows the history screen. All the historical disease data are displayed on this screen. The farmer can check the disease detail when clicking on any row on this screen. Such results will help farmers to understand what types of disease are happening at what time. By looking at the history of the disease, the farmers can use the correct pesticide in advance to prevent the disease again in the field. This would save the time and money of the farmer.

The farmer can delete the record at any time by swiping right to the left side of the row. A red delete button appears, which deletes the record when clicked.

5. Results

The different types of diseases are [22–33] detected using different machine learning techniques, but these approaches are mostly server-based where logic is implemented on the server side. We have written a program, which leverages the existing algorithms to prepare the ML
Classification:
Healthy : 82.12
Boll rot: 9.87

Recommendation:
Fields having long history of disease should be avoided for sowing. Field should be deeply ploughed and left for solarization. After harvesting, either plant debris should be completely buried or be removed. Early sowing and harvesting is recommended to avoid extreme temperatures. Sowing in April or June instead of May reduces disease incidence. Crop should be rotated. Intercropping using Vigna acontifolia reduces the incidence quite significantly. Seed treatment with Carboxin 37.5% + Thiram 37.5% DS @ 3.5 g/kg of seeds effectively reduces the root rot incidence.

Figure 9: Disease detail screen.

Figure 10: History screen.
model and embedded them on the mobile apps. First, the TensorFlow lite model was prepared using the TensorFlow open-source library. Now, this model is further converted into the CoreML model. The CoreML model is compatible with the iOS apps and is used in the development of the app.

The ML model is successfully created using the TensorFlow framework. The accuracy of the model was 90%. The model detects the boll rot and fungal leafspot diseases. The model also predicted healthy leaves. Using the converted ML model, an iOS app was successfully developed which detects the cotton plant diseases. The TensorFlow library internally used the neural network algorithm to train the image classification model.

Figure 11 shows the training and validation accuracy are close to each other in the above graph, which indicates that the model is balanced and works more accurately.

6. Conclusion

The TensorFlow model was successfully created and then converted into a CoreML model for iOS app development. An iOS mobile app is successfully developed, and the app detects the cotton leaf disease. As of now, the app supports only two diseases such as boll rot and fungal leafspot along with healthy leaves. The app complies with the iOS programming [34] and human interface [35] guideline. This app does not require an Internet connection of the device to detect the cotton disease. The ML model is embedded inside the project and works offline.

The accuracy of the model can be further improved by using some different approaches such as changing the training dataset, by adding more diverse images, by changing some input parameters, and by using other algorithms.

Based on this app architecture or reference, more disease detection can be added to this app, or a new app can be developed. Additionally, using this app architecture, more iOS apps can be developed for other domains or sectors based on the demand of the market.

In this app, server-side integration can be performed using the APIs, which communicates with the server. The model can be deployed on the server and app, communicate to the server using the APIs, and can display the result. The advantage of this approach is that the model can be changed at any time without affecting the app, while the disadvantage is that it requires the server need to be purchased and increases the overall cost of the system.
Data Availability
The cotton disease image set data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest
The authors declare that they have no conflicts of interest.

Acknowledgments
The authors would like to thank Prof. Dr. Rajeev Ratan and Senior Prof. Dr. J.V. Desai, MVN University, Palwal, India, for their guidance and help in the research. Also, they would like to thank Dr. Devvrat Akheriya for the support throughout this work.

References
[1] Agriculture, “Internet of Things in agriculture,” 2020, https://agriculture.vic.gov.au/farm-management/digital-agriculture/ internet-of-things-in-agriculture.
[2] C. Kamienski, J.-P. Soininen, M. Taumberger et al., “SWAMP: an IoT-based smart water management platform for precision Irrigation in agriculture,” in Proceedings of the 2018 Global Internet of Things Summit (GIoTS), pp. 1–6, Article ID IEEE, Bilbao, Spain, 2018.
[3] R. Ullah, A. W. Abbas, M. Ullah et al., “EEWMP: an IoT-based Energy-efficient water management platform for smart Irrigation,” Scientific Programming, vol. 2021, pp. 1–9, Article ID 5536884, 2021.
[4] D. L. Donoho, “Compressed sensing,” IEEE Transactions on Information Theory, vol. 52, no. 4, pp. 1289–1306, 2006.
[5] P. Sethi and S. R. Sarangi, “Internet of Things: Architectures, protocols, and applications,” Journal of Electrical and Computer Engineering, vol. 2017, pp. 1–25, 2017.
[6] Cottonic Cotton, “Incorporated, identification, and management of bacterial Blight of cotton,” 2020, https://www.cottonic.com/cotton-production/ag-research/plant-pathology/management-bacterial-blight-cotton/.
[7] TensorFlow, “TensorFlow for iOS guide,” 2022, https://www.tensorflow.org/lite/guide/ios.
[8] K. Ferentinos, “Deep learning models for plant disease detection and diagnosis,” Computers and Electronics in Agriculture, vol. 145, pp. 311–318, 2018.
[9] A. Shah, P. Gupta, and Y. M Ajgar, “Macro-Nutrient deficiency identification in plants using image processing and machine learning,” in Proceedings of the International Conference for Convergence in Technology, pp. 1–4, IEEE, Pune, India, April 2018.
[10] J. Shirahatti, R. Patil, and P. Akulwar, “A survey paper on plant disease identification using machine learning approach,” in Proceedings of the 2018 International Conference on Communication and Electronics Systems (ICCES-2018), pp. 1–4, IEEE, Coimbatore, India, October 2018.
[11] K. K. Singh, “An artificial intelligence and cloud based collaborative platform for plant disease identification, tracking and forecasting for farmers,” in Proceedings of the 2018, IEEE International Conference on Cloud Computing in Emerging Markets (CCEM), pp. 49–56, IEEE, Bangalore, India, November 2018.
[12] S. Militante, “Fruit Grading of Garcinia Binucao (Batuan) using image processing,” International Journal of Recent Technology and Engineering, vol. 8, no. 2, pp. 1829–1832, 2019.
[13] R. Ramesh, M. D’Souza, T. Asolkar, G. Achari, and J. Mathala, “Gupta, Rootstocks for the management of bacterial wilt in eggplant (Solanum melongena L.) and Tomato (Solanum lycopersicum L.) in the Coastal regions of India,” Advances in Agriculture, vol. 2022, Article ID 8594080, 2022.
[14] K. K. Shah, B. Modi, H. P. Pandey et al., “Diversified crop rotation: an approach for Sustainable agriculture production,” Advances in Agriculture, vol. 2021, Article ID 8924087, 2021.
[15] S. Kumar, R. Ratan, and J. V. Desai, “A study of iOS machine learning and artificial intelligence frameworks and libraries for cotton plant disease detection,” in Machine Learning, Advances in Computing, Renewable Energy and Communication, A. Tomar, H. Malik, P. Kumar, and A. Iqbal, Eds., Springer, Singapore. https://doi.org/10.1007/978-981-16-2354-7_24 Lecture Notes in Electrical Engineering, vol 768, 2022.
[16] TensorFlow, “TensorFlow guide,” 2022, https://www.tensorflow.org/lite/guide.
[17] Google, “Google Colab Notebook,” 2022, https://colab.research.google.com/?utm_source=scs-index.
[18] Pip, “Pip installation package,” 2022, https://pip.pypi.io/en/ stable/.
[19] Apple, “Framework core ML,” 2020, https://developer.apple. com/documentation/coreml.
[20] Apple, “Documentation core data,” 2021, https://developer.apple. com/documentation/coredata.
[21] S. Kumar, R. Ratan, and J. V. Desai, “Design of an iOS app architecture for cotton plant disease detection using artificial intelligence and machine learning techniques,” in Proceedings of Data Analytics and Management, D. Gupta, Z. Polkowski, A. Khanna, S. Bhattacharyya, and O. Castillo, Eds., Springer, Singapore. https://doi.org/10.1007/978-981-16-6289-8_55 Lecture Notes on Data Engineering and Communications Technologies, vol 90, 2022.
[22] R. Dwivedi, S. Dey, and S. Tiwari, “Notice of Violation of IEEE Publication principles: Grape disease detection network based on multi-task learning and Attention features,” IEEE Sensors Journal, vol. 21, no. 16, pp. 17573–17580, 2021.
[23] M. Bhanage and H. Hingoliwal/Hingoliwala, “Smart farming: Pomegranate disease detection using image processing,” Procedia Computer Science, vol. 58, pp. 280–288, 2015.
[24] B. ChandraKarmokar, M. Samawat Ullah, M. Kibria Siddique, and K. Rokibul Alam, “Tea leaf diseases recognition using neural network Ensemble,” International Journal of Computer Applications, vol. 114, no. 17, pp. 27–30, 2015.
[25] V. Pooja, R. Das, and V. Kanchana, “Identification of plant leaf diseases using image processing techniques,” in Proceedings of the Technological Innovations in ICT for Agriculture and Rural Development (TIAR), pp. 130–133, IEEE, Chennai, India, April 2017.
[26] K. Elangovan and S. Nalini, “Plant disease classification using image segmentation and SVM techniques,” International Journal of Computational Intelligence Research, vol. 13, no. 7, pp. 1821–1828, 2017.
[27] V. Singh and A. K. Misra, “Detection of plant leaf diseases using image segmentation and Soft Computing techniques,” Information Processing in Agriculture, vol. 4, no. 1, pp. 41–49, 2017.
[28] M. Brahim, K. Boukhalfa, and A. Moussaoui, “Deep learning for Tomato diseases: classification and Symptoms
Visualization,” Applied Artificial Intelligence, vol. 31, no. 4, pp. 299–315, 2017.

[29] S. Aasha Nandhini, R. Hemalatha, S. Radha, and K. Indumathi, “Web enabled plant disease detection system for agricultural applications using WMSN,” Wireless Personal Communications, vol. 102, no. 2, pp. 725–740, 2018.

[30] X. Zhang, Y. Qiao, F. Meng, C. Fan, and M. Zhang, “Iden-
tification of maize leaf diseases using improved deep con-
volutional neural networks,” IEEE Access, vol. 6, pp. 30370–30377, 2018.

[31] J. Ma, K. Du, F. Zheng, L. Zhang, and Z. Sun, “A recognition method for cucumber diseases using leaf symptom images based on deep convolutional neural network,” Computers and Electronics in Agriculture, vol. 154, pp. 18–24, 2018.

[32] R. R. Atole and D. Park, “A Multiclass deep convolutional neural network classifier for detection of common Rice plant Anomalies,” International Journal of Advanced Computer Science and Applications, vol. 9, no. 1, pp. 67–70, 2018.

[33] X. E. Pantazi, D. Moshou, and A. A. Tamouridou, “Automated leaf disease detection in different crop species through image features analysis and one class classifiers,” Computers and Electronics in Agriculture, vol. 156, pp. 96–104, 2019.

[34] Apple, “Apple app store guideline,” 2021, https://developer.
aple.com/app-store/review/guidelines/.

[35] Apple, “Apple human interface guidelines,” 2021, https://
developer.apple.com/design/human-interface-guidelines/.