An Image-based Plant Weed Detector using Machine Learning

Ahmed Abdelmoamen Ahmed*
Department of Computer Science
Prairie View A&M University
Prairie View, TX, USA
*Corresponding Author: amahmed@pvamu.edu

Jamil Ahmed
Department of Computer Science
Prairie View A&M University
Prairie View, TX, USA
jahmed1@PVAMU.EDU

***Abstract*—Plant diseases, pest infestation, weed pressure, and nutrient deficiencies are some of the grand challenges for the agricultural sector worldwide, which can result in substantial crop yield losses. To limit these losses, farmers must promptly identify the different types of plant weeds to stop their spread within agricultural fields. Farmers try to recognize plant weeds through color and multi-spectral imaging, and optical observation, which incorporates a significantly high degree of complexity, especially for large-scale farms. This paper presents an Artificial intelligence (AI)–powered system to automate the plant weeds identification process. The developed system uses the Convolutional Neural network (CNN) model as an underlying Machine Learning (ML) engine for classifying eight weed categories. The user interface is developed as an Android mobile app, allowing farmers to capture a photo of the suspected weed plants conveniently. It then displays the weed category along with the confidence percentage and classification time. The system is evaluated using different performance metrics, such as classification accuracy and processing time.**

***Keywords*-Weed Detector; Agriculture; AI; Machine Learning (ML); Mobile Computing; Communication; Edge Computing.**

I. Introduction

The agriculture sector worldwide grapples with significant challenges such as pest infestations, nutrient deficiencies, weed pressure, and plant diseases, which can result in substantial crop yield losses [1]. In particular, weed pressure represents an imposing threat to agricultural crop yield gains. Different weeds can compete voraciously with crops for sunlight, water, and nutrients. Despite the importance of such weed challenges, farmers use traditional ways to detect the types of weeds in their agriculture fields, including multi-spectral imaging and even optical observations, which are often time-consuming and dependent on specific data collection conditions. However, these conventional ways need to be more practical in the era of large-scale farms and precision agriculture [2].

In response, precision weed identification and control have become increasingly pertinent [3]. Concurrently, the success of Machine Learning (ML) [1] and Computer Vision [4] in agriculture have paved the way for developing efficacious image-based weed detection systems. This paper presents an ML-powered system designed to automate the process of plant weed identification and diagnosis. This system harnesses the computational prowess of Convolutional Neural Networks (CNN) [5], trained, validated, and tested using an imagery dataset encompassing 10,896 images, facilitating the classification of the most common eight weed categories. Figure 1 shows examples of various types of weeds from our dataset.

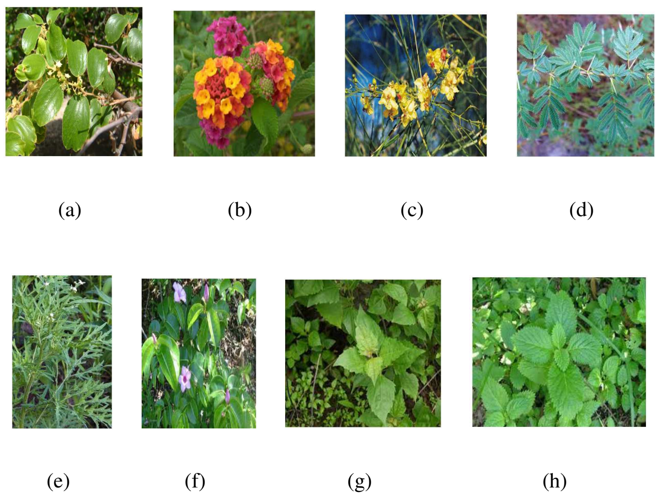


Fig. 1. Sample Examples from our Imagery Dataset: 1(a) Chinee Apple; 1(b) Lantana; 1(c) Parkinsonia; 1(d) Prickly Acacia; 1(e) Parthenium; 1(f) Rubber Vine; 1(g) Siam; 1(h) Snake.

We built a user-friendly interface using an Android mobile app that allows farmers to capture images of suspected weed leaves and rapidly receive the classified weed category, alongside a confidence percentage and the classification time. This functionality promises to enhance decision-making processes concerning the appropriate use of fertilizers, ultimately aiding in effective weed management and healthier crop growth [6]. Upon evaluation, the system exhibits an impressive average classification accuracy of 92% and an average classification time of a mere 0.88 seconds, demonstrating the system's efficiency and reliability.

II. Related Work

Global efforts to mitigate yield losses and enhance food security continue, given that plant diseases and weeds account for over 20% of crop losses worldwide [1]. Recent advancements in AI/ML in the agricultural domain [4], [7]–[9], coupled with the widespread use of mobile phones [2], [10],

[11], allow farmers to spot and identify the different types of weeds to prevent crop yield and quality losses [6].

DeepWeeds [9] is a multi-class weed species image dataset from the Australian rangelands. The dataset consists of 17,509 labeled images of eight nationally significant weed species native to eight locations across northern Australia. The authors used this dataset to train two machine learning models, Inception-v3 and ResNet-50, which achieved an average accuracy of 95.1% in classifying these eight weed classes.

Sapkota et al. in [6] collected an imagery dataset using an Unmanned Aerial System (UAS) to conduct site-specific weed control in a corn field by mapping the spatial distribution information of weeds in the field. The resulting prescription map was used to spray the field using a commercial-size sprayer, which showed an effective reduction in chemical usage for weed control scenarios. In particular, the study showed that 26.2% of the acreage from being sprayed with herbicide was saved, compared to the existing methods.

Another study was conducted in [7] to provide an effective solution for detecting multiple diseases in several plant varieties. The developed system was able to detect different plant diseases in various species, including apple, corn, grapes, potato, sugarcane, and tomato. The authors collected a dataset of 35k images of six species. The system achieved an average accuracy of 96.5% in identifying the studied plant diseases.

In summary, the review of plant weed detection using machine learning [7] and computer vision [4] shows that most of these approaches focus on particular weed classes [1], crop species [7], geographical regions or countries [9]. Moreover, most ML-based models are designed to work offline, which is inappropriate for real-time weed detection. Furthermore, to the best of our knowledge, none of the current ML-based approaches can be deployed on mobile devices due to their limited computational capabilities, which precludes minimizing communication delays and enhancing the farmer experience in using the system.

III. System Design

Figure 2 shows the distributed run-time system for the weed detector, which is organized with parts executing on mobile devices on the user side and centralized servers on the cloud side. Layer 1 describes the ML model used in the system (i.e., CNN) and the Intermediate Representation (IR) model that runs on the mobile device. Layer 2 illustrates the user interface, developed as an Android app to enable system users (shown in layer 3) to interact with the system conveniently.

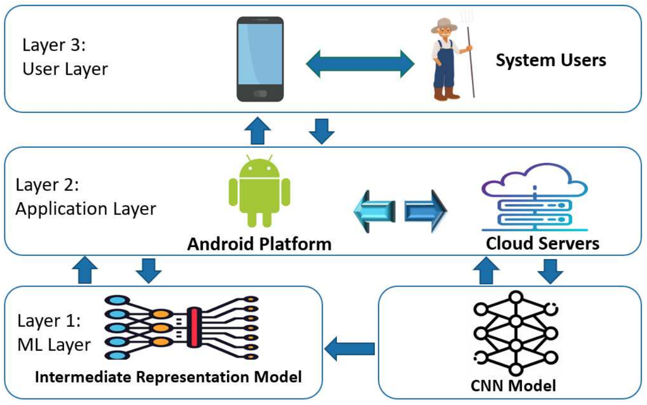


Fig. 2. System Architecture

TABLE I
The Number of Images used in the Training, Validation, and Testing Phases Across the Eight Weed Classes

Class	Training	Validation	Testing	Total
Chinee Apple	975	235	246	1,456
Lantana	937	233	223	1,393
Parkinsonia	918	226	227	1,371
Parthenium	913	224	225	1,362
Prickly Acacia	937	232	233	1,402
Rubber Vine	905	232	222	1,359
Siam	944	235	235	1,204
Snake	902	223	224	1,349

A. Dataset

Although standard object detection datasets (e.g., Microsoft COCO [12]) exhibit volume and variety of examples, they are unsuitable for plant disease and weed detection as they annotate object categories that do not include plant weeds. Therefore, we collected more than labeled 10,896 images of different weed leaves for training the CNN model from various sources such as Kaggle [13], Plant Village [14] and Google Web Scraper [15]. Most of the images in our dataset are in their natural environments because object detection is highly dependent on contextual information.

Our dataset is divided into three parts: training, validation and testing. Table I shows the number of images used in the three phases across the eight weed species. The number of images in each phase is determined based on the fine-tuned hyperparameters and structure of the CNN model.

B. CNN Structure

We trained a CNN model with four convolutional layers, one input layer, and one output layer. $I = [i_1, i_2, .., i_r]$ and $O = [o_1, o_2, ..., o_h]$ represent the input and output vectors, respectively, where r means the number of elements in the input feature set and h is the number of classes. The network's main objective is to learn a compressed dataset representation. In other words, it tries to approximately learn the identity function F, which is defined as:

$$F_{W,B}(I) \simeq I \tag{1}$$

where W and B are the network weights and biases vectors.

The training algorithm involves two phases: forward and backward phases. During the forward phase, the network's weights are kept fixed, and the input data is propagated through the network layer by layer. In the backward phase, the error signal e_i is propagated through the network in the backward direction. During this phase, error adjustments are applied to the CNN network's weights for minimizing e_i. We used the gradient descent first-order iterative optimization algorithm to calculate the change of each neuron weight $\Delta\omega_{i,j}$, which is defined as follows:

$$\Delta\omega_{i,j} = -\eta \frac{\delta\varepsilon(n)}{\delta e_j(n)} y_i(n) \tag{2}$$

where $y_i(n)$ is the intermediate output of the previous neuron n, η is the learning rate, and $\varepsilon(n)$ is the error signal in the entire output. $\varepsilon(n)$ is calculated as follows:

$$\varepsilon(n) = \frac{1}{2}\sum_j e_j^2(n) \tag{3}$$

We further customized the CNN model by implementing object classification using focal loss, Υ, which is defined as:

$$\Upsilon = -\sigma_\tau (1-p_\tau)^\gamma \; log(p_t) \tag{4}$$

where γ is the modulating factor that balances the training dataset, σ_τ is a factor used to balance the number of true positive and false negatives samples, p_τ is the estimated probability of the ground truth, and $log(p_\tau)$ is the cross entropy for the model's binary classification.

Before moving the trained CNN model to the mobile device, we had to convert it into an optimized Intermediate Representation (IR) model based on the trained network topology, weights and biases values. We used the Intel OpenVINO toolkit [16] to generate the IR model, which is the only format that the inference engine at NCS2 accepts and understands. The conversion process involved removing the convolution and pooling layers that are not relevant to the inference engine at the stick. In particular, OpenVINO splits the trained model into two types of files: XML and Bin extension. The XML files contain the network topology, while the BIN files contain the weights and biases binary data.

IV. Implementation

A. CNN Training and Implementation

We had to normalize the range of pixel intensity values of leaf images in the dataset before training the CNN model. This step was necessary because all dimensions of feature vectors extracted from input images should be in the same intensity range. This made the convergence of our CNN model faster during the training phase. Image normalization was implemented by subtracting the input image's mean value μ from each pixel's value $I(i,j)$, and then dividing the result by the standard deviation σ of the input image. The distribution of the output pixel intensity values would resemble a Gaussian curve centered at zero. We used the following formula to normalize each image in our training set:

$$O(i,j) = \frac{I(i,j)-\mu}{\sigma} \tag{5}$$

where I and O are the input and output images, respectively; and i and j are the current pixel indices to be normalized.

The training images must have the same size before feeding them as input to the model. Our model was trained with colored (RGB) images with resized dimensions of 400×400 pixels. We set the batch size and number of epochs to be 50 images and 100 epochs, respectively. The model training was carried out using a server computer equipped with a 4.50GHz Intel Core^{TM} i7-16MB CPU processor, 16GB of RAM, and CUDA GPU capability. The training phase took approximately 45 hours to run 100 epochs. We took a snapshot of the trained weights every 5 epochs to monitor the progress. The training error and loss are calculated using this equation:

$$M = \frac{1}{n}\sum_{i=1}^{n}(y_i - x_i)^2 \tag{6}$$

where M is the mean square error of the model, y is the value calculated by the model, and x is the actual value. M represents the error in object detection.

We measured these performance metrics during the training process: training and validation accuracies, training and validation losses, and classification confidence. Figure 3 illustrates the efficiency of the training process across 100 epochs. As shown in the figure, both loss and accuracy curves plateaued after the 85th epoch, indicating that even though training continued, little was learned. This shows us that our data worked very well, given the type of data and the parameters set. Figure 4 shows the CNN classification confidence for the eight weed classes, with an average confidence rate of 93.4%.

The CNN model is implemented using Keras development environment 2.4 [17]. Keras is an open-source neural network library written in Python using TensorFlow 02 as a back-end engine. Keras libraries running on top of TensorFlow make it easier for developers to build ML models written in Python.

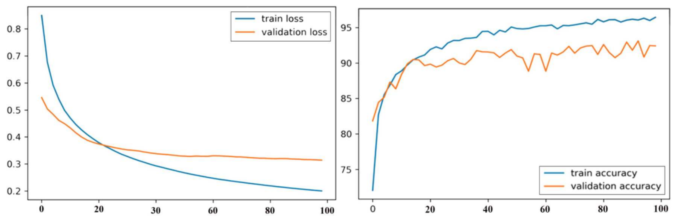


Fig. 3. The Training and Validation Accuracy vs. The Training and Validation Losses of the CNN Model

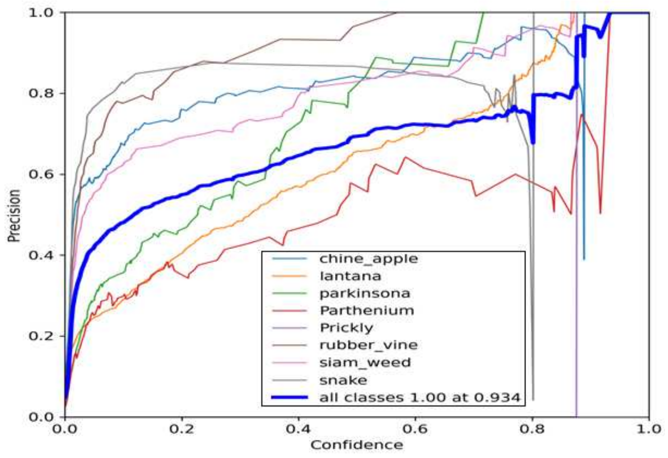


Fig. 4. The CNN Classification Confidence

TABLE II
THE CONFUSION MATRIX FOR THE CNN MODEL: CHINEE APPLE (CA), LANTANA (LA), PARKINSONIA (PS), PARTHENIUM (PT), PRICKLY ACACIA (PA), RUBBER VINE (RV), SIAM (SI) AND SNAKE (SN)

Class	CA	LA	PS	PT	PA	RV	SI	SN
CA	230	8	0	3	0	1	0	4
LA	2	216	0	0	0	0	2	3
PS	1	0	221	0	4	0	0	0
PT	1	4	3	206	0	6	0	5
PA	1	0	0	4	227	0	0	1
RV	4	4	0	0	1	211	1	1
SI	0	2	0	0	0	0	233	0
SN	29	9	0	1	1	0	0	184

TABLE III
THE PRECISION VS. RECALL VS. F1-SCORE VALUES OF THE CNN MODEL FOR ALL WEED CLASSES

Class	Precision	Recall	F1-Score
Chinee Apple	0.78	0.87	0.83
Lantana	0.83	0.94	0.88
Parkinsonia	0.95	0.96	0.95
Parthenium	0.93	0.85	0.89
Prickly Acacia	0.89	0.92	0.90
Rubber Vine	0.95	0.83	0.89
Siam	0.92	0.94	0.93
Snake	0.88	0.74	0.80

B. Mobile App

The plant weed detector's user interface is implemented as a self-contained mobile app developed using Android Development Environment ADT bundle (64-bit). The app uses different technologies and tools, including the Android SDK and XML, to create the front-end interface. We used Python's Flask Web Framework as a middleware between the app and the database server on the cloud and SQLite for the database.

The mobile app allows farmers to photograph the suspected weed plant or upload an existing image on the phone with proper alignment and orientation. The orientation handler, which runs as a background service thread in the mobile app, is responsible for correcting the tilt and camera angle of capturing the plant photo. Once the right image is captured, the app uploads it to a cloud server to detect the weed class(es) by applying our CNN model. The captured image is transferred to the cloud side via a REST (Representational State Transfer) service in the form of a JSON image object.

V. EXPERIMENTAL EVALUATION

Figure 5 illustrates some examples of the inference result of the weed detector system. Most notably, the system classified the Prickly Acacia, Lantana, and Siam classes correctly with a confidence score of 95%, 96%, and 97%, respectively. The operations of class prediction and displaying results took around 0.88 seconds, including the communication overheads. This shows that our system can be used as a plant weed detector in real-time at the network edge.

Our system, in most cases, delivers good results in natural conditions even when the plant images are captured from different angles from the camera, orientations, and illumination conditions. However, sometimes, the system fails to achieve such high confidence levels for some classes. For instance, Figure 5(c) illustrates an example of an 82% confidence ratio for detecting the snake weed class. This may be justified by the confusion between Snake and Chinee Apple classes because they have similar leaf phenology, as shown in Table II.

Table II shows the confusion matrix for the CNN model that gives a detailed analysis of how the model performance changes for different weed classes. The matrix rows represent the actual (true) disease classes, and the columns correspond to the predicted classes. The diagonal cells show the proportion of the correct predictions of our CNN model, whereas the off-diagonal cells illustrate the error rate of our model. The confusion matrix demonstrates that our model, in most cases, can differentiate between the weed classes and achieve high prediction accuracy. For the three most common types of weed classes, Parkinsonia, Parthenium, and Rubber Vine, the model achieves accuracies above 95%, 93%, and 95%, respectively.

The precision, recall and F1-score ratios, shown in Table III, summarizes the trade-off between the true-positive rate and the positive predictive value for our CNN model using different probability thresholds. Precision represents the positive predictive value of our model, while recall is a measure of how many true positives are identified correctly, and F1-score takes into account the number of false positives and false negatives. As shown in the table, most of the precision vs. recall values tilts towards 1.0, which means that our CNN model achieves high accuracy while minimizing the number of false negatives.

The precision ratio describes the performance of our model at predicting the positive class. It is calculated by dividing the number of true positives by the sum of the true positives and false positives, as follows:

$$Precision = \frac{TruePositives}{TruePositives + FalsePositives} \tag{7}$$

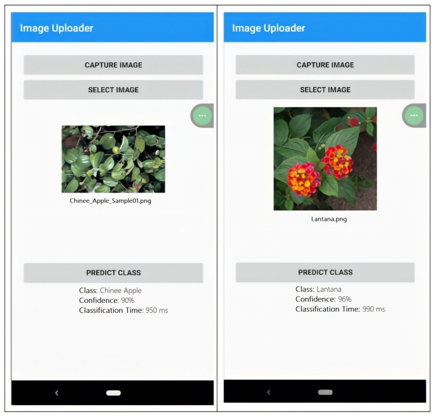
The recall ratio is calculated as the ratio of the number of true positives divided by the sum of the true positives and the false negatives, as follows:

$$Recall = \frac{TruePositives}{TruePositives + FalseNegatives} \tag{8}$$

F1-score ratio is calculated by a weighted average of both precision and recall, as follows:

$$Recall = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{9}$$

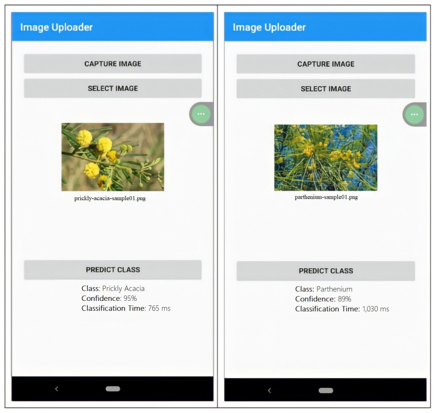
Table IV shows the classification accuracy and prediction time across the eight weed classes. The CNN model achieved an overall average classification accuracy of 92%. Class prediction and displaying results via the mobile app took an average 0.88 seconds. This is evident that farmers can diagnose any plant weeds in their agricultural fields using a handy mobile app in less than one second. The table shows that our model is robust and can operate in real-time inference in the agricultural fields with high accuracy.



Class: Chinee Apple
Confidence: 90%
Classification Time: 950 ms

Class: Lantana
Confidence: 96%
Classification Time: 990 ms

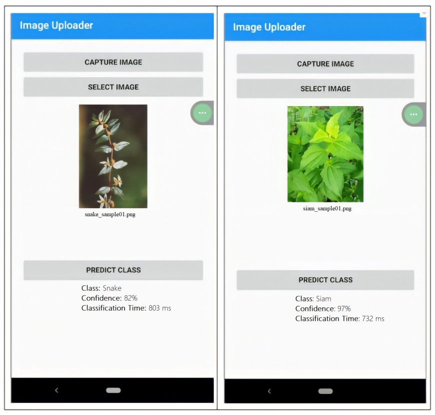
(a) Chinee Apple and Lantana



Class: Prickly Acacia
Confidence: 95%
Classification Time: 765 ms

Class: Parthenium
Confidence: 89%
Classification Time: 1,030 ms

(b) Prickly Acacia and Parkinsonia



Class: Snake
Confidence: 82%
Classification Time: 803 ms

Class: Siam
Confidence: 97%
Classification Time: 732 ms

(c) Snake and Siam

Fig. 5. Examples of Successful Recognition of Different Weed Classes in Natural Conditions

TABLE IV
THE AVERAGE CLASSIFICATION ACCURACY AND PREDICTION TIME OF THE CNN MODEL

Class	Validation Precision	Testing Precision	Time (s)
Chinee Apple	78%	87%	0.96
Lantana	83%	94%	1.03
Parkinsonia	95%	96%	0.78
Parthenium	93%	85%	1.13
Prickly Acacia	89%	92%	0.76
Rubber Vine	95%	83%	0.79
Siam	92%	94%	0.75
Snake	88%	74%	0.83

VI. CONCLUSIONS

This paper presented an ML-powered plant weed detector that enables farmers to diagnose the most common eight weed species. We trained a CNN model using an imagery dataset consisting of 10,896 photos of different weed leaves, where crowded backgrounds, low contrast, and diverse illumination condition images are considered. To increase the system's usability, we developed a mobile app that would create a better opportunity for limited-resources farmers to detect plant weeds in their early stages and eliminate the use of incorrect fertilizers that can hurt the health of both the plants and soil. This system is expected to create a better opportunity for farmers to keep their crops healthy and eliminate the use of wrong fertilizers that could stress the plants. We found that our system achieved a classification accuracy of 92 percent. In ongoing work, we are looking into opportunities for generalizing our approach to be deployed locally at UAS, which farmers can use to monitor their crops from the sky.

ACKNOWLEDGMENTS

This research work is supported in part by the National Science Foundation (NSF) under grants # 2011330, 2200377 and 2302469. Any opinions, findings, and conclusions expressed in this paper are those of the authors and do not necessarily reflect NSF's views.

REFERENCES

[1] K. Hu, Z. Wang, and G. e. a. Coleman, "Deep learning techniques for in-crop weed recognition in large-scale grain production systems: a review," *Precision Agriculture*, vol. Hu2023, pp. 1573–1618, 2023.
[2] A. A. Ahmed and G. H. Reddy, "A mobile-based system for detecting plant leaf diseases using deep learning," *AgriEngineering*, vol. 3, no. 3, pp. 478–493, 2021.
[3] A. A. Ahmed, S. A. Omari, R. Awal, and A. F. et al., "A distributed system for supporting smart irrigation using iot technology," *Engineering Reports*, vol. 3, pp. 1–13, 2020.
[4] Z. Wu, Y. Chen, B. Zhao, and X. e. a. Kang, "Review of weed detection methods based on computer vision," *Sensors*, vol. 21, no. 11, 2021.
[5] S. Ren, K. He, and R. G. et al., "Faster r-cnn: Towards real-time object detection with region proposal networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 6, pp. 1137–1149, 2017.
[6] R. Sapkota, J. Stenger, and M. e. a. Ostlie, "Towards reducing chemical usage for weed control in agriculture using uas imagery analysis and computer vision techniques," *Scientific Reports*, vol. 13, pp. 1–14, 2023.
[7] S. V. Militante, B. D. Gerardo, and N. V. Dionisio, "Plant leaf detection and disease recognition using deep learning," in *IEEE Eurasia Conference on IOT, Communication and Engineering*, 2019, pp. 579–582.
[8] A. M. Hasan, F. Sohel, and D. e. a. Diepeveen, "A survey of deep learning techniques for weed detection from images," *Computers and Electronics in Agriculture*, vol. 184, pp. 0168–1699, 2021.
[9] A. Olsen, D. Konovalov, and B. e. a. Philippa, "Deepweeds: A multiclass weed species image dataset for deep learning," *Scientific Reports*, vol. 9, no. 1, pp. 2045–2322, 2019.
[10] A. A. Ahmed and M. Echi, "Hawk-eye: An ai-powered threat detector for intelligent surveillance cameras," *IEEE Access*, vol. 9, pp. 283–293, 2021.
[11] A. A. Ahmed, "A privacy-preserving mobile location-based advertising system for small businesses," *Engineering Reports*, vol. e12416, pp. 1–15, 2021.
[12] T. Lin, M. Maire, and S. J. B. et al., "Microsoft COCO: common objects in context," *Computer Vision*, vol. 1405.0312, no. 1, 2014.
[13] "Kaggle: Machine learning and data science community," accessed December 4, 2023. [Online]. Available: https://www.kaggle.com/
[14] D. P. Hughes and M. Salathe, "An open access repository of images on plant health to enable the development of mobile disease diagnostics," *Computers and Society*, pp. 1–13, 2016.
[15] "Google web scraper," accessed December 4, 2023. [Online]. Available: https://chrome.google.com/webstore/detail/web-scraper/jnhgnonknehpejjnehehllkliplmbmhn?hl=en
[16] "Openvino toolkit," accessed December 4, 2023. [Online]. Available: https://software.intel.com/content/www/us/en/develop/tools/openvino-toolkit.html
[17] "Keras: A python deep learning api," accessed December 4, 2023. [Online]. Available: https://keras.io/