Enhancing Machine Learning Training Performance in Smart Agriculture Datasets Using a Mobile App

Temirlan Zarymkanov*, Amin Kargar†, Cristina M. Pinotti‡, Brendan O'Flynn†, Dimitrios Zorbas*
*Nazarbayev University, School of Engineering & Digital Sciences, Astana, Kazakhstan
†Tyndall National Institute, University College Cork, Cork, Ireland
‡Department of Computer Science and Mathematics, University of Perugia, Italy
Authors' email: {firstname.lastname}@nu.edu.kz, {firstname.lastname}@tyndall.ie, cristina.pinotti@unipg.it

***Abstract*—The agricultural sector faces significant challenges due to invasion of pests that damage crops and cause significant loss of production. Traditional methods to detect these insects are cost ineffective, thus, automated vision systems based on machine learning (ML) have recently been proposed in the literature. However, a significant issue is the lack of a prior dataset to build the ML model on. To mitigate this problem, we propose a new approach to train a model using a small initial dataset and continually improve the accuracy process by retraining it on new images labeled by a mobile application. Retraining is performed on new data, which comes from a mobile application that displays pictures of insects and prompts expert users to label them. The users' input is used to retrain the model on new coming images. Specifically, our method trains the model on 100 initial images, and retrains it with every 100 new images. The IP102 large-scale dataset for pest recognition was used to demonstrate the effectiveness of the approach. The results show an improvement of accuracy of up to 50 percentage units for the built Convolutional Neural Network (CNN) model.**

***Index Terms*—Machine Learning, Image Classification, Pest Detection, Model retraining, Mobile application**

I. Introduction

Agriculture is an important aspect of our life as the population of the earth is ever-growing and so is the demand for food. Annually, 40% of the crops are destroyed by pests creating gigantic losses for farmers[1]. If the issue of crop damage caused by insect pests is addressed correctly, it could result in economic benefits for the entire agricultural sector and lower resource usage to produce a large amount of food [1].

The initial step to prevent such damage is to identify and categorize insects accurately, distinguishing between the harmless and the harmful ones. However, traditional monitoring processes require particular training of humans to recognize the insects and are time-costly and inefficient [2]. In that regard, there has been a growing interest in using smart Internet of Things (IoT) systems that employ artificial intelligence and computer vision to detect pests, such as [3] and [4].

These IoT systems periodically collect data (pictures) from the insects and transmit this data to a server (cloud). When a considerable amount of data is collected, an ML model is built and then it is transferred to the IoT device to autonomously detect and classify the insects. It is important for this process to be done on the edge to avoid the time and energy-consuming task of transmitting the pictures to the cloud but also to improve the detection response time. However, the process of collecting data to build the model may be considerably high, while building a model using a few only data may lead to low accuracy and false conclusions.

To mitigate such a problem, we propose a solution where new pictures coming from IoT devices are fed to a mobile application where expert users can identify and automatically label the insects in the pictures. The model is then retrained once the number of new labelled images exceeds a certain threshold. Once the accuracy is above a certain level, the model is then transferred to the IoT devices. We show through a use case scenario on the IP102 large-scale dataset [5] that the accuracy of the model can rapidly be improved.

The rest of the paper is structured as follows. The related work is discussed in Section II. Section III presents the methodology of the current work. Evaluation results of the use-case scenario are presented and discussed in Section IV. Finally, conclusions and ideas for future work are drawn in Section V.

II. Related Work

The related work is divided into two parts. The first part is related to autonomous IoT systems with or without ML where their focus is to autonomously detect and classify insects, while the second part, deals with works about dataset construction, augmentation, and transfer. Recent works of these two categories are described below.

There are several projects dealing with the problem of autonomous pest identification and classification. In one of them, Miranda et al. [6] create a system for extracting insects from images captured with digital cameras on the field. Then, they apply background modeling to captured images to detect the presence of pests, while they use a median filter to reduce the noise in the image. In a similar work, Chen et al. [7] investigate the use of sensors in conjunction with computer vision using IoT data for insect classification. The authors use the deep learning model YOLOv3 to detect pests and Long Short-Term Memory to confirm its performance. They reach an accuracy of 90% on their sample dataset. In another work, using CNN-based models with multi-branch and multi-scale

[1]https://www.fao.org/news/story/en/item/1402920/icode

attention network Ung et al. [8] achieve an accuracy of 74% on the IP102 dataset.

Moreover, there have been several studies that combine automated IoT systems with ML models. Guo et al. [4] propose an automatic method for tracking flying vegetable insects. The system makes use of a camera to periodically take pictures and upload them to a cloud server through a wireless network. The cloud server contains the YOLO for Small Insect Pests (YOLO-SIP) model which is used for insect classification. Zhao et al. [9] developed an automatic system for pest detection based on a camera and a Raspberry Pi board as an information collector. Images captured by the information collector device are transferred to a cloud server for processing, and then the results are displayed on the end user's device. The processing part benefited from a deep learning model named DPeNet for insect detection and classification. However, such systems use cloud-based servers that are focused on improving the accuracy of insect detection and pay little attention to computing capacity, memory requirements, or power consumption.

In some other works, ML computing and image analysis are performed on the edge device. A few-shot recognition technique is created by Li et al. [10] to recognize cotton pests, which uses triplet loss to categorize insect species and a CNN to extract features. Their technique is put into practice on an embedded system, which combines an ARM processor for program control with a specially designed FPGA and achieves 95.4% accuracy on a dataset from the National Bureau of Agricultural Insect Resources. Brunelli et al. [11] present an ultra-low power smart pest detecting system for grayscale images with a resolution of 244x324 pixels. This system uses the GAP8 microcontroller [12] to create a machine learning model and extract some areas of interest in the acquired image for classification. It processes images and classifies objects with an accuracy of 93%. Albanese et al. [13] propose an edge device using Raspberry Pi to monitor Codling Moths. This study investigates the performance of three off-the-shelf deep neural network models including MobileNetV2, LeNet, and VGG16 for image classification in terms of accuracy and power consumption. Similarly, Suto [14] works on an embedded insect monitoring system to detect Codling Moths based on a Raspberry Pi board. This study also uses MobileNetV2 as a base model for insect classification.

The above studies worked on specific insects, and if the system is to be used for other insects, the image processing part must be trained for new insects, which means a new data set is needed. But the problem is that the variety of insect species is very large and there is no suitable dataset for each of them. Therefore, the leading challenge is to collect the appropriate dataset. A proper dataset should contain thousands of images of each class which is a time-consuming process.

Hence, insect detection models generally suffer because of the insufficient number of images of targeted insects. In small datasets, the lack of diversity of classes leads to overfitting and poor generalization which negatively affect the accuracy. For example, Kargar et al. [15] proposed an edge-based device to monitor and detect the Halyomorpha Halys (HH) in orchards using a CNN-based model. This system was trained using a small dataset and achieved 70% accuracy when it was deployed in an orchard. This obviously highlights the impact of a small dataset on accuracy in real-world scenarios.

There are several methods that try to resolve issues related to small datasets. One of the proposed solutions is artificial data generation. Li et al. [16] introduce mega-trend-diffusion (MTD) that is commonly used to generate synthetic training samples. The MTD method employs a shared diffusion function to disperse a group of data and applies information diffusion through fuzzy possibility distribution to establish potential coverage of a dataset across an attribute domain. Another approach is data augmentation, which applies data transformations such as rotation, translations, and flips. Han et al. [17] propose a novel two-phase method combining CNN transfer learning and web data augmentation. They use a pre-trained network, while the original small dataset can be augmented with images, and therefore expand the training dataset and achieve a better performance. Sorbelli et al [18] worked on the detection of Halyomorpha halys (HH) in orchards. In this study, drones were used to capture images from the target insect in orchards. Then several augmentation techniques were used to increase the dataset, and a YOLO-based model was utilized to detect HH on the collected dataset. Although the augmentation technique is commonly used to increase the dataset size, mitigate overfitting and improve model generalization, it has its own limitations as well. This technique uses predefined transformations to create new images from existing images, so it may not represent diversity in real-world data. In fact, since there are lots of insect species, this technique could not properly reflect real-world diversity in such datasets.

Reading the literature review, it is straightforward that despite the research effort on implementing efficient ML solutions on the cloud/edge, the problem of low accuracy due to the initial small size of samples still exists.

III. Methodology

This section presents the methodology followed in the proposed approach along with the dataset and information about the mobile application.

A. Phases

As shown in Fig. 1, the methodology of the proposed solution is divided into two phases. The first process is responsible for the model training using a small dataset (training set) while the second phase is responsible for retraining the data using the labeled information from the app. Data from the training set is preprocessed and used to train the initial version of the model. The initial model is then saved on a cloud platform. In the second phase, images from the retraining set are used in a mobile application to get user feedback. The new labeled images are stored in the cloud platform, and once the number of labeled images is enough, the model retraining is initiated and its accuracy is recorded.

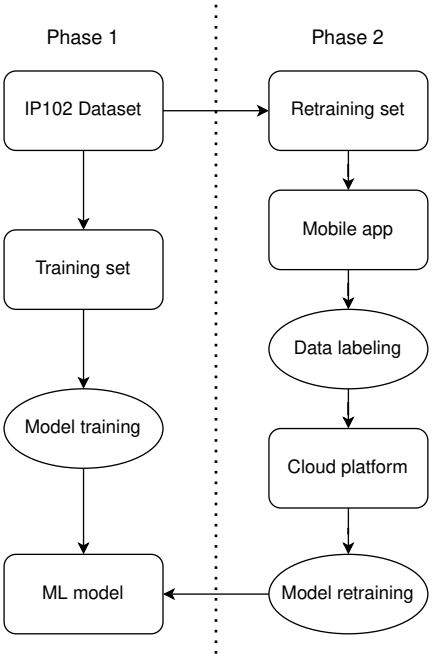

Fig. 1. The methodology followed in the approach.

Fig. 2. Two chosen insects from IP102 dataset, Left: Cicadellidae, Right: Lycorma delicatula.

B. Dataset

For the needs of our study, the IP102 dataset is employed. It is the largest publicly available pest detection dataset containing 75,222 images of insects belonging to 102 categories. For convenience and presentation purposes, we have reduced the size of our dataset images to 7600 images by choosing only two classes with the largest number of images: "Lycorma delicatula" with 3716 images and "Cicadellidae" with 4017 images. Some samples of these two insects are shown in Fig. 2. Binary classification was chosen in order to simplify the training process and reduce the training time. The reduced dataset is split into two sets: the training and the retraining parts. As it was mentioned in the previous subsection, the training set contains labeled images and is used for initial training of the model. The retraining set consists of unlabeled images that will appear in the mobile app where expert users can label these images.

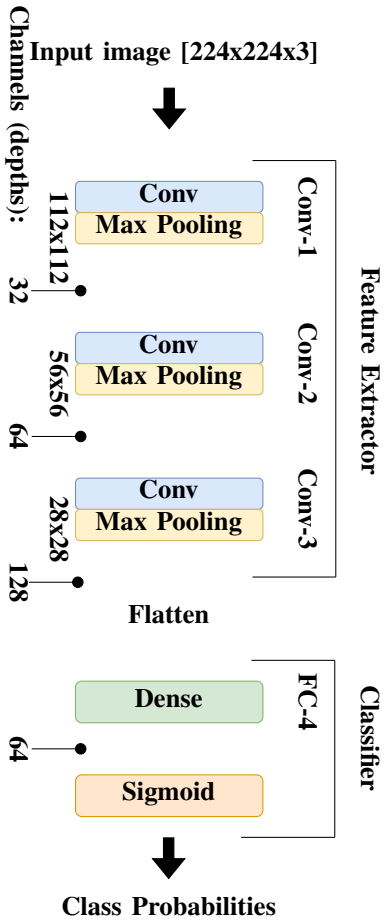

Fig. 3. The adopted Machine Learning model architecture.

C. Machine Learning Model

In order to correctly identify and classify insects, a CNN model is built using the Keras API in TensorFlow library. The proposed network model consists of several layers as shown in Fig. 3. The convolutional layer (Conv) is used to detect and extract important features such as edges, textures and patterns. A max-pooling layer is applied after each convolutional layer to reduce the feature map dimensions while keeping important information. This combination of layers is repeated three times to capture and extract important features from the input image. Then, a fully connected (FC) layer is used to perform the classification task. The output layer is a single-neuron layer with a sigmoid activation function that represents the likelihood of the input image belonging to one of the two classes (Cicadellidae or Lycorma Delicatula).

The model has 5,631,169 parameters in total and it is trained to minimize the binary cross-entropy loss and optimized with the Adam optimizer. The model input shape is 224x224x3, which represents the dimensions of the input images.

Fig. 4. A screenshot of the mobile application where the user indicates whether the inspected insect is present in the sample picture or not. The app works for one insect per time. (An image from the Andoird emulator is shown for presentation purposes.)

D. Mobile application

Fig. 4 shows a sample screenshot of the mobile application which prompts the users to answer the question "Is Cicadellidae there?" by choosing one of two buttons. If the user taps "Yes", the Cicadellidae label will be assigned to the image. After each choice, a new image is shown.

The mobile application was created using the Dart programming language and the Flutter[2] framework which was chosen due to its cross-platform feature that gives the ability to launch it on both iOS and Android phones. Moreover, Python and Django[3] were used to create an API that handles requests from mobile applications. The Django application was hosted on the Heroku[4] cloud server.

A requirement of the system that employs the mobile application is that expert users with sufficient knowledge about the inspected insects must be available. If the labeling of data is done incorrectly, then more time to collect more images will be required to adapt to a better accuracy.

IV. Evaluation & Discussion of the Results

The evaluation methodology along with evaluation results run on the IP-102 dataset are presented and discussed in this section.

[2]https://flutter.dev
[3]https://www.djangoproject.com
[4]https://www.heroku.com

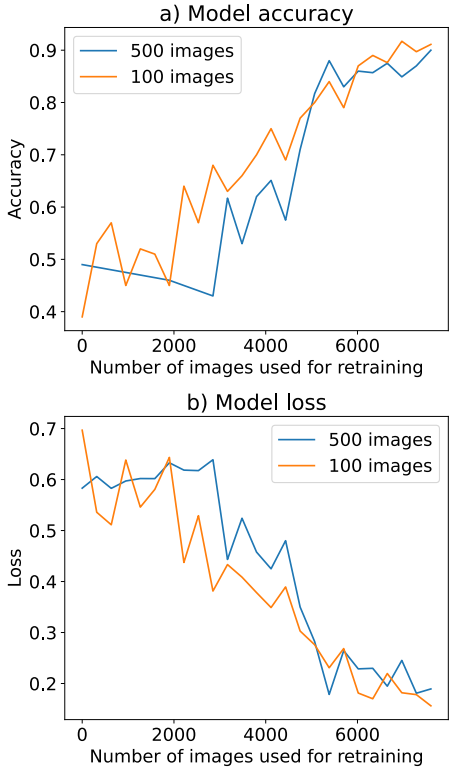

Fig. 5. Model accuracy and loss for different numbers of images per retraining.

A. Evaluation metrics

There are several metrics and methods available to evaluate the proposed approach. First, a number of metrics to assess the model performance were used. The accuracy of the model is the most important one, as it shows the percentage of correctly predicted classes. Other known model evaluation metrics such as the Binary Cross-Entropy Loss, the recall, and the F1-score were also employed.

Second, the rate of accuracy improvement was measured. To do so, a different number of images per retraining process is selected to see how much the model performance changes and at what rate. Finally, for the needs of the experiments (not for a real-world scenario), the best ratio of splitting the dataset into training and retraining sets was identified.

B. ML Model Performance

The initial model training was done using 100 images out of 3317 images. While the training dataset size increases with each retraining iteration, the test dataset size remains the same. Fig. 5a shows the model performance for 100 and 500 images per retraining. The accuracy increases from 40% to 90% in both cases. Additionally, Binary Cross-Entropy Loss, a metric which is commonly used in binary classification, is shown in Fig. 5b. Lower values of the Binary Cross-Entropy Loss indicate that the model assigns higher probabilities to the correct class and lower probabilities to the incorrect class.

Fig. 6. ML evaluation metrics per retraining step (every 100 images).

Thus, the results indicate that the model performance is improving with retraining new images.

Fig. 6 shows the precision, the recall, and the F1-score for the model that was retrained with 100 images and each retraining iteration score was recorded. Recall measures the proportion of correctly predicted positive instances out of all actual positive instances. It increases from around 45% to 97%, which means that the model can identify positive instances more often. Precision shows how many of the positive predictions made are correct (true positives). It also increases with each retraining, and when the model predicts a positive result, it is likely to be accurate. F1-score is a harmonic mean of precision and recall, and it is useful when dealing with imbalanced data, which is often the case with small datasets. The F1-score increases with the retraining and we can conclude that the model correctly identifies both classes.

C. Initial training set size

During the initial training of the dataset consisting of 100 images, the accuracy was around 40%. We saw in the previous subsection that after retraining the model with the app feedback, the accuracy increased to 90%. In this subsection, we assessed the impact of the initial dataset size on the model performance by selecting a larger initial set.

Fig. 7a depicts the model's accuracy results, which was trained on 30% of the whole dataset and retrained on 70% of it. The difference was only in the base accuracy, which was 70% at the beginning, but increased to 90%. In addition, as depicted in Fig. 7b, Binary Cross-Entropy Loss decreased in both cases and, as with accuracy, the difference was in the initial loss value. These results confirm the usefulness of the methodology using the mobile app even with large initial sets.

V. Conclusion & Future Work

In this paper, we proposed an approach to improve the accuracy of an insect classification model trained on a small initial dataset. This approach involves retraining the model

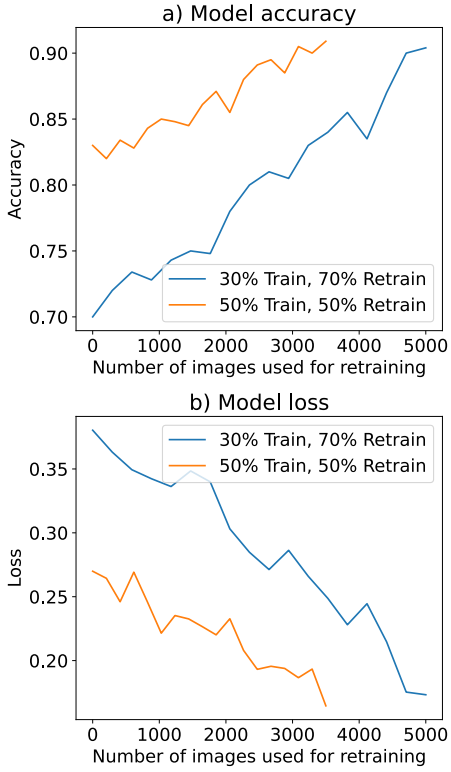

Fig. 7. Model accuracy and loss for different retraining and training set sizes.

with newly gathered images that are labelled using a mobile application handled by experts. To evaluate the effectiveness of the approach, we trained the model on 100 pictures of the insects as a small initial dataset, and then continuously kept adding new images to retrain the model. This way, our model performance improved from 40% to 90.2% with the help of the mobile application used for labeling and retraining. The proposed methodology can be used in cases where there is not a lot of data available to train a model of high accuracy, for example, in presence of new discovered insects.

In the future, we are planning to work on the feasibility of implementing the training and retraining approaches on an IoT device.

Acknowledgement

This publication has emanated from research conducted with the financial support of the Haly.ID project – 2020EN508 funded by Ireland's Department of Agriculture, Food and the Marine under Grant: 2020 Trans National ERA-NET and Nazarbayev University grant No. 11022021FD2916 for the project "DELITMENT: DEterministic Long-range IoT MEsh NeTworks". The second author is supported by a Walsh Scholarship funded by Teagasc, The Irish Food and Agriculture Authority.

References

[1] N. Ullah, J. A. Khan, L. A. Alharbi, A. Raza, W. Khan, and I. Ahmad, "An efficient approach for crops pests recognition and classification based on novel deeppestnet deep learning model," *IEEE Access*, vol. 10, pp. 73019–73032, 2022.

[2] A. Gutierrez, A. Ansuategi, L. Susperregi, C. Tubío, I. Rankić, and L. Lenža, "A benchmarking of learning strategies for pest detection and identification on tomato plants for autonomous scouting robots using internal databases," *Journal of Sensors*, vol. 2019, pp. 1–15, 2019.

[3] H. Nagar and R. Sharma, "Pest detection on leaf using image processing," in *2021 International Conference on Computer Communication and Informatics (ICCCI)*, pp. 1–5, 2021.

[4] Q. Guo, C. Wang, D. Xiao, and Q. Huang, "Automatic monitoring of flying vegetable insect pests using an RGB camera and YOLO-SIP detector," *Precision Agriculture*, vol. 24, no. 2, pp. 436–457, 2023.

[5] X. Wu, C. Zhan, Y.-K. Lai, M.-M. Cheng, and J. Yang, "Ip102: A large-scale benchmark dataset for insect pest recognition," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 8787–8796, 2019.

[6] J. L. Miranda, B. D. Gerardo, and B. T. Tanguilig III, "Pest detection and extraction using image processing techniques," *International Journal of Computer and Communication Engineering*, vol. 3, no. 3, p. 189, 2014.

[7] C.-J. Chen, Y.-Y. Huang, Y.-S. Li, C.-Y. Chang, and Y.-M. Huang, "An aiot based smart agricultural system for pests detection," *IEEE Access*, vol. 8, pp. 180750–180761, 2020.

[8] H. T. Ung, H. Q. Ung, and B. T. Nguyen, "An efficient insect pest classification using multiple convolutional neural network based models," *arXiv preprint arXiv:2107.12189*, 2021.

[9] N. Zhao, L. Zhou, T. Huang, M. F. Taha, Y. He, and Z. Qiu, "Development of an automatic pest monitoring system using a deep learning model of dpenet," *Measurement*, vol. 203, p. 111970, 2022.

[10] Y. Li and J. Yang, "Few-shot cotton pest recognition and terminal realization," *Computers and Electronics in Agriculture*, vol. 169, p. 105240, 2020.

[11] D. Brunelli, T. Polonelli, and L. Benini, "Ultra-low energy pest detection for smart agriculture," in *2020 IEEE SENSORS*, pp. 1–4, IEEE, 2020.

[12] E. Flamand, D. Rossi, F. Conti, I. Loi, A. Pullini, F. Rotenberg, and L. Benini, "Gap-8: A risc-v soc for ai at the edge of the iot," in *2018 IEEE 29th International Conference on Application-specific Systems, Architectures and Processors (ASAP)*, pp. 1–4, IEEE, 2018.

[13] A. Albanese, M. Nardello, and D. Brunelli, "Automated pest detection with dnn on the edge for precision agriculture," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 11, no. 3, pp. 458–467, 2021.

[14] J. Suto, "A Novel Plug-in Board for Remote Insect Monitoring," *Agriculture 2022, Vol. 12, Page 1897*, vol. 12, p. 1897, nov 2022.

[15] A. Kargar, M. P. Wilk, D. Zorbas, M. T. Gaffney, and B. Q'Flynn, "A novel resource-constrained insect monitoring system based on machine vision with edge ai," in *2022 IEEE 5th International Conference on Image Processing Applications and Systems (IPAS)*, vol. Five, pp. 1–6, 2022.

[16] D.-C. Li, C.-S. Wu, T.-I. Tsai, and Y.-S. Lina, "Using mega-trend-diffusion and artificial samples in small data set learning for early flexible manufacturing system scheduling knowledge," *Computers & Operations Research*, vol. 34, no. 4, pp. 966–982, 2007.

[17] D. Han, Q. Liu, and W. Fan, "A new image classification method using CNN transfer learning and web data augmentation," *Expert Systems with Applications*, vol. 95, pp. 43–56, 2018.

[18] F. Betti Sorbelli, L. Palazzetti, and C. M. Pinotti, "Yolo-based detection of halyomorpha halys in orchards using rgb cameras and drones," *Computers and Electronics in Agriculture*, vol. 213, p. 108228, 2023.