MATHEMATICAL MODELING AND COMPUTING, Vol. 11, No. 3, pp. 870–882 (2024)

A data-driven fusion of deep learning and transfer learning for orange disease classification

Sghir A.[1], Ziani M.[1], El Handri K.[1,2,3]

[1] *LMSA Laboratory, Department of Mathematics, Faculty of Sciences, Mohammed V University in Rabat, Morocco*
[2] *Aivancity School of AI & Data for Business & Society, France*
[3] *Faculty of Medicine and Pharmacy, Mohammed V University in Rabat, Medical Biotechnology (MedBiotech) Laboratory*

(Received 15 December 2023; Revised 12 September 2024; Accepted 17 September 2024)

In agriculture, early detection of crop diseases is imperative for sustainability and maximizing yields. Rooted in Agriculture 4.0, our innovative approach combines pre-trained Convolutional Neural Networks (CNNs) models with data-driven solutions to address global challenges related to water scarcity. By integrating the combined L_1/L_2 regularization technique to our model layers, we enhance their flexibility, reducing the risk of the overfitting effect of the model. In the orange dataset used in our experiments, we have 1790 orange images, including a class of fresh oranges and three disease categories. Applied on this dataset for classification, our model exhibits notable performance, namely 92.17% for CNN and 97.28% for ResNet-50 model. Evaluated across metrics like accuracy, precision, recall, F1-score, confusion matrix, and cross validation, our approach surpasses traditional classifiers, significantly contributing to smart agricultural and global food resilience amidst mounting water scarcity pressures.

Keywords: *multi-class classification; orange disease; water scarcity; Agriculture 4.0; data-driven; deep learning; convolutional neural networks; transfer learning; ResNet-50.*

2010 MSC: 68T10, 68Uxx, 92Cxx, 68Pxx, 68T42 **DOI:** 10.23939/mmc2024.03.870

1. Introduction and background

Agriculture, as the foundation of human sustenance, faces a multitude of challenges exacerbated by the global environmental crisis and dwindling water resources [1]. Among the crucial agricultural commodities, oranges hold a significant place, contributing to nutrition, livelihoods, and economic growth worldwide. However, the efficient management of orange crop health is under duress, especially in regions grappling with the scarcity of water, a challenge that threatens sustainable production [2]. In response to this pressing issue, innovative technologies and data-driven methodologies have emerged as indispensable tools for sustaining crop health in water-scarce environments [3]. At the forefront of this transformation lie deep learning techniques, particularly Convolutional Neural Networks (CNNs), which have revolutionized image analysis, including disease detection in plants [4]. Our article aims to harness the power of CNNs, renowned for their ability to extract intricate patterns from images, and complement this capability with transfer learning, allowing the adaptation of pre-existing knowledge to the specific context of orange disease classification. By integrating these techniques, the article seeks to provide a robust tool empowering stakeholders in the agricultural sector to effectively manage diseases in water-scarce environments. This fusion approach has the potential to revolutionize disease detection and contribute to sustainable agricultural practices for citrus cultivation. Figure 1 illustrates the used process.

In the context of our article, we will provide an in-depth exploration of this critical issue. In the upcoming sections, we resolve to delve into the heart of our study on enhancing the management of orange crop health in environments constrained by limited water resources. Our analysis starts with a 'Review of Literature', where we will delve into existing knowledge regarding the classification of

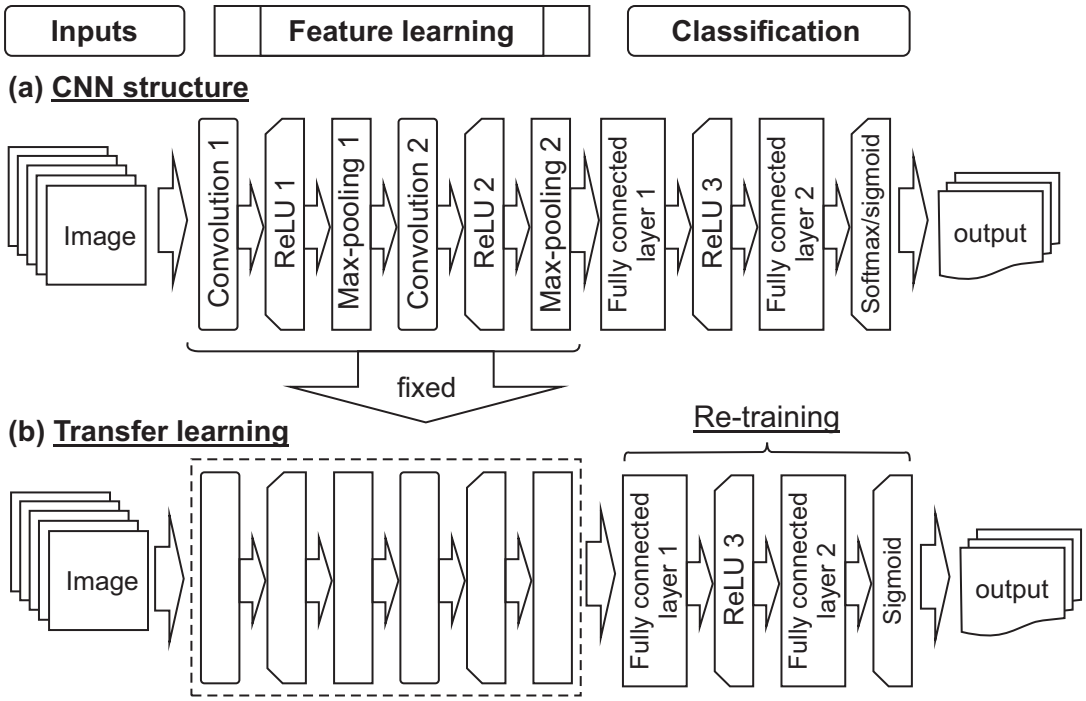


Fig. 1. Convolutional Neural Network Coupled with a Transfer-Learning.

diseases in plant crops. This review will establish a solid foundation for our research. We will then move on to the 'Methodology', detailing the tools, techniques, and data used in our analysis. The following 'Results' will provide insights into the findings of our study, and the 'Discussion' section will unveil the implications of these results. At last, the 'Conclusion' will bring together all these elements to offer a comprehensive perspective on how our approach, based on deep learning and transfer learning, can transform the management of orange crop health in arid contexts.

2. Literature review

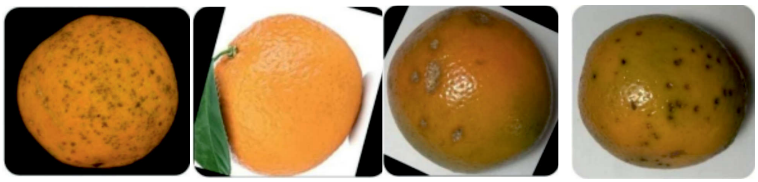
Plant disease detection using Convolutional Neural Networks (CNNs) and transfer learning has shown remarkable accuracy in various studies. Researchers have achieved high levels of accuracy, often exceeding 90%, in identifying plant diseases through image analysis. For instance, in a study by Rupali Saha et al. [5], a CNN-based model achieved an accuracy of approximately 93.21% in identifying orange fruit disease, showcasing the potential of deep learning in this domain. Similarly, a research project by Poonam Dhiman et al. reported an accuracy rate of around 98.25% in classifying plant diseases using deep learning models CNN-LSTM (CNN Long Short-Term Memory Network) with edge computing [6].

One notable advantage of the transfer learning approach is its ability to achieve high accuracy even with limited datasets, which is often the case in agriculture due to the complexity and diversity of diseases. For instance, a study by Junde Chen et al. [7] used transfer learning with a pre-trained VGGNet model and achieved an accuracy of over 92% in identifying multiple rice plant images. Similarly, a research project by Zhang et al. [8] reported an accuracy rate of around 96.90% in classifying orange diseases using the DenseNet model. This study compares two Convolutional Neural Network (CNN) architectures, MobileNet and Self-Structured (SSCNN), for classifying citrus leaf diseases, offering a cost-effective method for detection based on smartphone images. The results indicate that SSCNN achieved higher accuracy, with a 99% validation accuracy at epoch 12, compared to MobileNet, making it a more accurate and efficient choice for classifying citrus leaf diseases from smartphone images [9]. Another research assesses the application of convolutional neural networks (CNNs) for fruit counting in agriculture. Two common CNN architectures, Faster R-CNN with Inception V2 and SSD with MobileNet, achieve fruit counting performance up to 93% and 90%, respectively, which could enhance decision-making in agriculture [10].

3. Methodology

3.1. Data collection and preprocessing

Table 1 and Figure 2 illustrate the dataset used in our paper. The dataset contains 1790 images, of which 1164 are used to train the model, and 626 are used to test it. In this dataset, there is the class with fresh oranges and three other diseases, namely, citrus canker, black spot, and greening citrus. Each image has a height of 28 pixels and a width of 28 pixels, for approximately 784 pixels. Each pixel is associated with a single pixel value, indicating the brightness of that pixel. This pixel value is an integer ranging from 0 to 255.



a ***b*** ***c*** ***d***

Fig. 2. Sample images of dataset (***a***) Grenning, (***b***) Fresh, (***c***) Canker, (***d***) Blackspot.

Table 1. Orange diseases dataset.

Class	Count	Group
0	344	Blackspot
1	349	Canker
2	552	Fresh
3	545	Grenning

With a preliminary understanding of our data, our next crucial step is data preparation for our neural network. We will reformat all image data into a 128×128 format. Our dataset is clean and requires minimal preprocessing. This ensures a reliable foundation for deep learning, allowing us to concentrate on model development without extensive data cleaning.

3.2. CNN model implementation

3.2.1. Convolutional layers

The convolutional layer serves as the initial component in the architecture of CNNs. Its primary function is to extract features from input images. The feature map results from the convolution operation between a filter and the input image. This feature map encapsulates comprehensive information about image structures, encompassing interesting points, textures, edges, ridges, and more. Subsequent layers depend on this feature map to perform operations like dimensionality reduction and reduction of the number of features [11–13].

Definition 1. *Let X be an input matrix of dimensions $m \times n$, representing pixel values of an image. Let W be a weight matrix (filter) of dimensions $p \times q$. The convolution of X by the filter W is defined as*

$$(X * W)_{ij} = \sum_{a=1}^{p} \sum_{b=1}^{q} X_{i+a-1,j+b-1} \cdot W_{ab}. \tag{1}$$

By adding the bias term b, the output Y of the convolutional layer is obtained by applying an activation function f element-wise:

$$Y_{ij} = f\big((X * W + b)_{ij}\big). \tag{2}$$

The choice of the activation function depends on the model, but Rectified Linear Unit (ReLU) is commonly used. In terms of dimensions, if X is $m \times n$ and W is $p \times q$, the output Y is $(m - p + 1) \times (n - q + 1)$.

Remark 1. The definition 1 above holds for convolutional layer parameters set to $K = 1$ (using a single filter), $S = 1$ (stride of 1 for complete coverage), $P = 0$ (no zero-padding), and $D = 1$ (no dilation applied). The layer utilizes a single filter to extract specific features from the input image. The stride (S) is set to 1, ensuring complete coverage of the image without skipping pixels, and zero pixels are added around the image ($P = 0$). Additionally, no dilation is applied to the filter ($D = 1$), preserving proximity between the elements of the convolution kernel.

3.2.2. Activation function (ReLU)

To ensure the non-linearity of the convolutional layer, to improve the effectiveness of treatment and to overcome the problem of saturation, we use ReLU as activation function. The ReLU (Rectified Linear Unit) activation function [14] is formulated as follows.

Definition 2. *For an input value x, the ReLU (Rectified Linear Unit) function is defined as*

$$\mathrm{ReLU}(x) = \begin{cases} 0 & \text{if} \quad x < 0 \\ x & \text{if} \quad x \geqslant 0 \end{cases} = \max\{0, x\}. \tag{3}$$

One of its advantages is its low computational cost, as it involves taking the maximum between the value 0 and the weighted sum value. Another advantage is that it does not suffer from saturation, as it has no upper limit in the positive region and all negative values in input are replaced by zeros.

3.2.3. Pooling layers

The pooling layer is strategically positioned between the convolutional layers in a neural network architecture. Its primary purpose is to reduce the size of images while retaining crucial features, achieved by selecting either the maximum or average values within rectangular regions. The input image is partitioned into a series of rectangles, each containing n pixels. MaxPooling identifies the maximum element in each region, AveragePooling calculates the average output of the region, and Sumpooling computes the sum of the elements within the region.

Definition 3. *Let X be an input matrix of dimensions $m \times n$, representing an activation map. Consider a pooling window of size $p \times q$. The output matrix Z after MaxPooling, AveragePooling, and SumPooling operations is defined, respectively, as:*

$$\begin{aligned} Z(i,j) &= \max_{p',q'} X(i \times p + p', j \times q + q'), \\ Z(i,j) &= \frac{1}{p \times q} \sum_{p',q'} X(i \times p + p', j \times q + q'), \\ Z(i,j) &= \sum_{p',q'} X(i \times p + p', j \times q + q'). \end{aligned} \tag{4}$$

The output matrix will have dimensions $\frac{m}{p} \times \frac{n}{q}$.

3.2.4. Flatten layers

"Flatten" is a vital step in Convolutional Neural Networks (CNNs), converting the multi-dimensional output into a one-dimensional vector. This process streamlines feature maps into a linear array, serving as input for fully connected layers. It bridges convolutional and fully connected layers, enabling the network to understand complex data relationships.

Definition 4. *If X is an input matrix of dimensions $m \times n$, then Y is defined as $Y = [X_{1,1}, X_{1,2}, \ldots, X_{m,n}]$, where $X_{i,j}$ represents the element located at the i-th row and j-th column of the matrix X.*

3.2.5. Fully connected layers

The final layer of the CNN takes as input our image that has been entirely transformed by convolutions and other operations (in vector form), and produces the prediction.

3.2.6. Activation function (SoftMax)

The SoftMax activation function that assigns a probability to each of these distinct classes while ensuring that the sum of these probabilities equals 1. This type of activation function aligns perfectly with our objectives, and it is commonly employed in multi-layer neural networks and for multi-class classifications. When applied to the output layer, Softmax transforms the raw scores (also known as logits) produced by the previous layers into probabilities that correspond to different classes. These probabilities represent the model's confidence in each class prediction. The SoftMax activation function is formulated as follows.

Definition 5. *For $X = (x_1, \ldots, x_k)$, we have $\sigma_i(X) = \frac{\exp x_i}{\exp x_1 + \ldots + \exp x_k}$.*

3.2.7. Learning algorithm

Each neuron has a weight, and the goal is to adjust these weights to reach the overall minimum value of the loss function [15]. Our categorical cross-entropy loss function measures the disparity between the probability distribution predicted by the model and the actual probability distribution of labels in a dataset with multiple classes. J is minimized following two steps:

— Forward Propagation: it is the passage of input data through the network, where each layer performs specific operations such as convolution, activation, pooling, etc. The results from each layer are transmitted to the next layer, and this process repeats until reaching the output of the softmax layer, ultimately generating a prediction.

— Backpropagation: this process updates the weights of neurons based on the gradient descent of the loss function. The new weight value for each neuron depends on the difference between the predicted value and the observed one. As long as neurons have a high error rate, their new weights will be very small, with higher weights for neurons with lower error rates.

Regularized loss function. The incorporation of combined L_1 and L_2 regularization into our loss function represents a significant advancement in optimizing the weights of our neural network model. This unique approach leverages the distinct benefits of L_1 regularization, promoting sparsity by eliminating less important connections, and L_2 regularization, controlling weight growth to avoid excessive values. By combining these two techniques, our model achieves a subtle balance between selecting crucial features and maintaining optimization stability. This innovative strategy contributes to faster convergence during training, improved generalization to unseen data, and increased resilience against overfitting. The regularized loss function $J_{\text{reg}}(W^{(l)})$ for a layer l is defined as a combination of categorical cross-entropy loss and L_1/L_2 regularization terms,

$$J_{\text{reg}}\big(W^{(l)}\big) = -\frac{1}{m}\sum_{i=1}^{m}\sum_{c=1}^{C} Y_{i,c}\log\big(\hat{Y}_{i,c}\big) + \frac{\lambda_1}{2m}\sum_{i,j}\big|W_{i,j}^{(l)}\big| + \frac{\lambda_2}{2m}\sum_{i,j}\big(W_{i,j}^{(l)}\big)^2, \tag{5}$$

where m is the number of samples in the dataset, C is the number of classes, $Y_{i,c}$ is the actual value of class c for sample i, $\hat{Y}_{i,c}$ is the predicted value of class c for sample i, λ_1 and λ_2 are regularization coefficients for L_1 and L_2.

Gradient of regularized loss function. The gradient with respect to the weights $W^{(l)}$ is obtained by adding the gradients of categorical cross-entropy loss and L_1/L_2 regularization terms,

$$\frac{\partial J_{\text{reg}}}{\partial W^{(l)}} = \frac{\partial J}{\partial W^{(l)}} + \frac{\lambda_1}{2m}\cdot\operatorname{sign}\big(W^{(l)}\big) + \frac{\lambda_2}{m}\cdot W^{(l)}, \tag{6}$$

where $\frac{\partial J}{\partial W^{(l)}}$ is the gradient of categorical cross-entropy loss.

Update moments with bias correction and weights with Adam Optimizer.

Algorithm 1 ADAM Algorithm [16].

Inputs: Learning rate η, β_1, β_2, ε; initial parameters $W_{\text{old}}^{(l)}$
Initialization: $m_0 \leftarrow 0$, $v_0 \leftarrow 0$, $t \leftarrow 0$
while not converged
Calculate gradient $g_t \leftarrow \nabla J_{\text{reg}}\big(W_{\text{old}}^{(l)}\big)$
$t \leftarrow t+1$, $m_t \leftarrow \beta_1 \cdot m_{t-1} + (1-\beta_1)\cdot g_t$, $v_t \leftarrow \beta_2 \cdot v_{t-1} + (1-\beta_2)\cdot g_t^2$
$\hat{m}_t \leftarrow \frac{m_t}{1-\beta_1^t}$
$\hat{v}_t \leftarrow \frac{v_t}{1-\beta_2^t}$
$W_{\text{new}}^{(l)} \leftarrow W_{\text{old}}^{(l)} - \eta \cdot \frac{\hat{m}_t}{\sqrt{\hat{v}_t}+\varepsilon}$
$W_{\text{new}}^{(l)} \leftarrow W_{\text{new}}^{(l)} - \frac{\lambda_1}{2m}\cdot\operatorname{sign}\big(W_{\text{new}}^{(l)}\big) - \frac{\lambda_2}{m}\cdot W_{\text{new}}^{(l)}$
$W_{\text{old}}^{(l)} \leftarrow W_{\text{new}}^{(l)}$

Our model is a sequential Convolutional Neural Network (CNN) for image classification, see Figure 3.

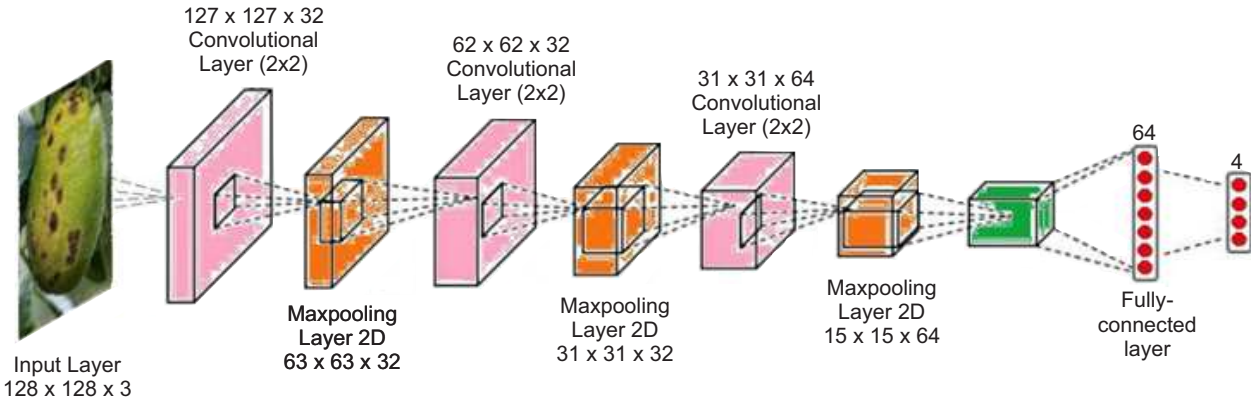


Fig. 3. Description of implemented model.

It begins with convolutional layers with ReLU activation and max-pooling to reduce spatial dimensions. Each Conv2D layer is equipped with L_1 and L_2 regularization, with respective rates of 0.01, allowing for the control of weights and ensuring model stability. The final layers include a flattening step, followed by dense layers with ReLU activation and dropout to prevent overfitting. The output layer employs softmax activation for multi-class classification. The model is compiled using the Adam optimizer with a learning rate 0.0001 and categorical cross-entropy loss. It is trained for 50 epochs on provided training data, focusing on accuracy as the evaluation metric. The training time is recorded to measure its efficiency. The selection of these parameters and hyper-parameters represents the culmination of an extensive series of experiments involving various regularization methods and numerous CNN architectures. The model configurations above represent the optimal settings we have identified to achieve the best possible results.

3.3. Transfer learning with ResNet-50

Transfer learning is a machine learning technique that involves repurposing a pre-trained model from one dataset to address a different problem with a distinct dataset but a related context. This approach spares us the effort of painstakingly searching for the optimal model and investing substantial time in constant testing and parameter tuning. This becomes particularly beneficial when dealing with sizable datasets and intricate tasks, where training a model from scratch can be time-intensive. Leveraging transfer learning mitigates the need for an extensive dataset to introduce a new model, yielding a more efficient and effective way to approach new implementations [17, 18].

Residual Block in ResNet-50

The residual block in ResNet50 consists of three key steps.

• Batch Normalization (BN).

Definition 6. *Let X be the input matrix and Y be the matrix after the convolution operation. Batch Normalization is applied to normalize Y, producing the normalized matrix $N^{(i)}$:*

$$N^{(i)} = \gamma^{(i)} \left(\frac{Y^{(i)} - \mu^{(i)}}{\sqrt{\sigma^{(i)2} + \varepsilon}} \right) + \beta^{(i)}, \tag{7}$$

where $Y^{(i)}$ is the output after the convolution at the i-th layer, $\gamma^{(i)}$ and $\beta^{(i)}$ are the parameter matrices of BN, $\mu^{(i)}$ and $\sigma^{(i)}$ are the means and standard deviations calculated along the appropriate axes, and ε is a small constant to avoid division by zero.

• Residual Block with ReLU: A residual block $F_i(X^{(i-1)})$ with $X^{(i-1)}$ as input is defined as

$$F_i\big(X^{(i-1)}\big) = \mathrm{RELU}\left(\mathrm{BN}\left(W_{i,j} * F_{i-1}(X^{(i-1)}) + X^{(i-1)}, \gamma_{i,j}, \beta_{i,j}\right)\right), \tag{8}$$

where $W_{i,j}$ is the convolution weight matrix, $\gamma_{i,j}$ and $\beta_{i,j}$ are the Batch Normalization parameters, $*$ is the convolution operator and ReLU is the activation function.

• Residual Connections: Residual connections are introduced by directly adding the input $X^{(i-1)}$ to the output of the residual block, producing the final matrix $F_i(X^{(i-1)})$.

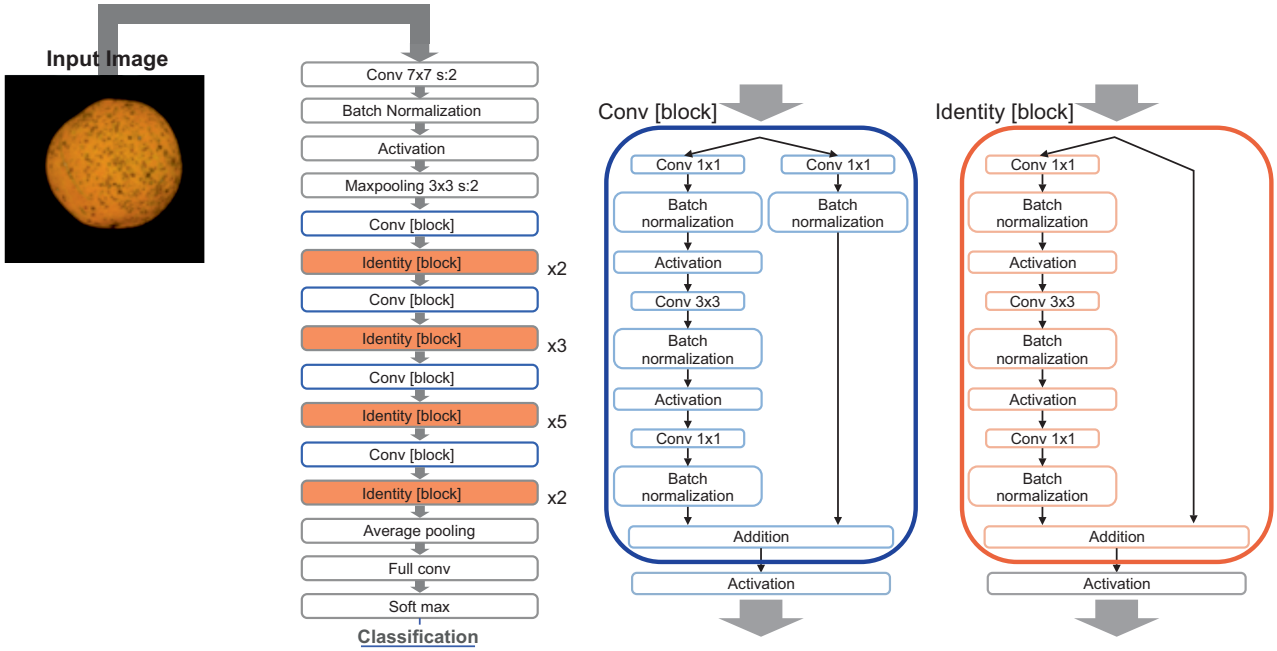


Fig. 4. Representation of the RESNET-50 architecture.

Table 2. Hyper-parameters used in the ResNet50-based model.

Model	ResNet50
Input shape	(128,128,3)
Weight	Initialized to ImageNet
Optimizer	Adamax
Loss function	Categorical crossentropy
Learning rate	0.0001
Regularization	L_1 and L_2 Regularization
Classifier	Softmax
Epochs	50
Batch size	32
Dropout rate	0.25

The overall structure of ResNet50 is formed by stacking multiple of these residual blocks. The final output is obtained by passing through these blocks successively. The network parameters, such as the weights of convolutions and Batch Normalization parameters, are adjusted during training using the gradient descent backpropagation algorithm, as mentioned earlier. Figure 4 represents the RESNET-50 architecture.

We employed ResNet-50 as our base model, as indicated in Table 2. This model was initially trained to recognize objects in the ImageNet dataset. We opted for it due to its widespread recognition and strong performance across various image classification tasks.

3.4. Comparison between the general connections and the residual connections used in ResNet

The architecture of a standard Convolutional Neural Network (CNN) and a residual block in ResNet50 presents a distinct contrast in how information is processed. The conventional CNN follows a sequential approach, progressively transforming features of an image across its layers. However, this method may face challenges such as the vanishing gradient during the training of deep networks. In contrast, ResNet50 introduces residual blocks that create shortcuts to facilitate the propagation of information. In these blocks, input information is added to the output rather than simply being transformed. This creates a "residual path", allowing the network to learn identity functions, thereby easing gradient propagation. This architecture addresses the vanishing gradient problem and enables the training of much deeper networks. Figure 5 shows this difference.

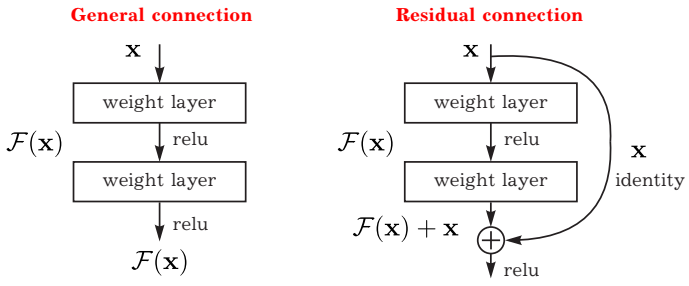


Fig. 5. The comparison between the general connections and the residual connections used in ResNet.

4. Results and discussion

4.1. Performance evaluation of CNN model

The accuracy curve of the CNN model, with a test accuracy of 92.17%, showcases its remarkable ability to classify test data successfully. This high performance reflects the model's reliability in its predictions. Simultaneously, the low loss of 4.0404 indicates its efficiency in minimizing errors during training, thereby enhancing its overall performance and predictive capability.

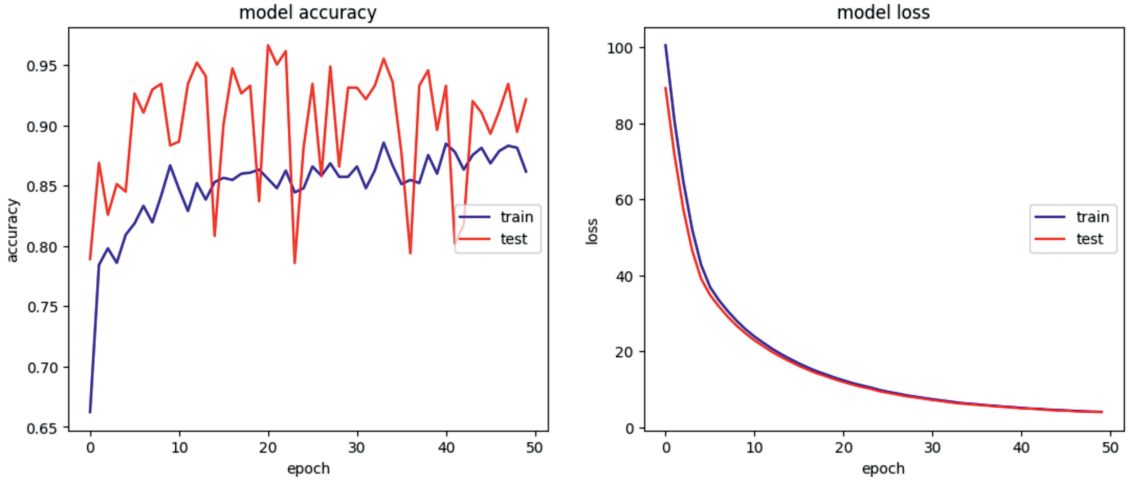


Fig. 6. Accuracy curve and loss curve of the CNN model.

The presented table illustrates the performance metrics of the CNN model across different classes of orange crop diseases. Notably, the "Fresh" and "Grenning" classes exhibit exceptional results with perfect accuracy, recall, and F1-Score, each scoring 100%. The "Blackspot" class achieved an accuracy of 76%, while maintaining a high recall of 96% and an F1-Score of 85%. The "Canker" class demonstrated a remarkable accuracy of 95%, with a recall of 69% and an F1-Score of 80%. These metrics provide a detailed insight into the model's proficiency in accurately classifying and recognizing various orange crop diseases.

The confusion matrix indicates that the model incorrectly classified 14 instances of the "Blackspot" class as "Canker". Similarly, 105 instances of the "Canker" class were mistakenly predicted as belonging to the "Blackspot" class. Additionally, 2 instances of the "Canker" class were erroneously classified as "Fresh". Despite these few misclassifications, the overall high accuracy, recall, and F1-score values, as mentioned earlier, reflect the CNN model's robustness in correctly classifying orange crop diseases.

Table 3. Performance metrics of the CNN model.

Class	Accuracy	Recall	F1-Score	Support
Blackspot	0.76	0.96	0.85	344
Canker	0.95	0.69	0.80	349
Fresh	1.00	1.00	1.00	552
Grenning	1.00	1.00	1.00	545

The cross-validation results showcase the robustness and generalization ability of our model across diverse folds. With an average accuracy of approximately 85.47%, our model consistently demonstrated its proficiency in correctly classifying instances. The F1-score, averaging around 78.21%, reflects a harmonious balance between precision and recall, indicative of the model's ability to handle both false positives and false negatives effectively. Furthermore, the impressive average AUC of approximately 97.67% attests to the model's discriminative power in distinguishing between classes.

4.2. Performance evaluation of ResNet-50

The ResNet50 model achieved outstanding performance, as evidenced by its accuracy curve. With an impressive test accuracy of 97.28%, the model exhibited an exceptional ability to correctly classify the dataset. Moreover, the model's remarkably low loss value of 1.6199 indicates minimal errors in

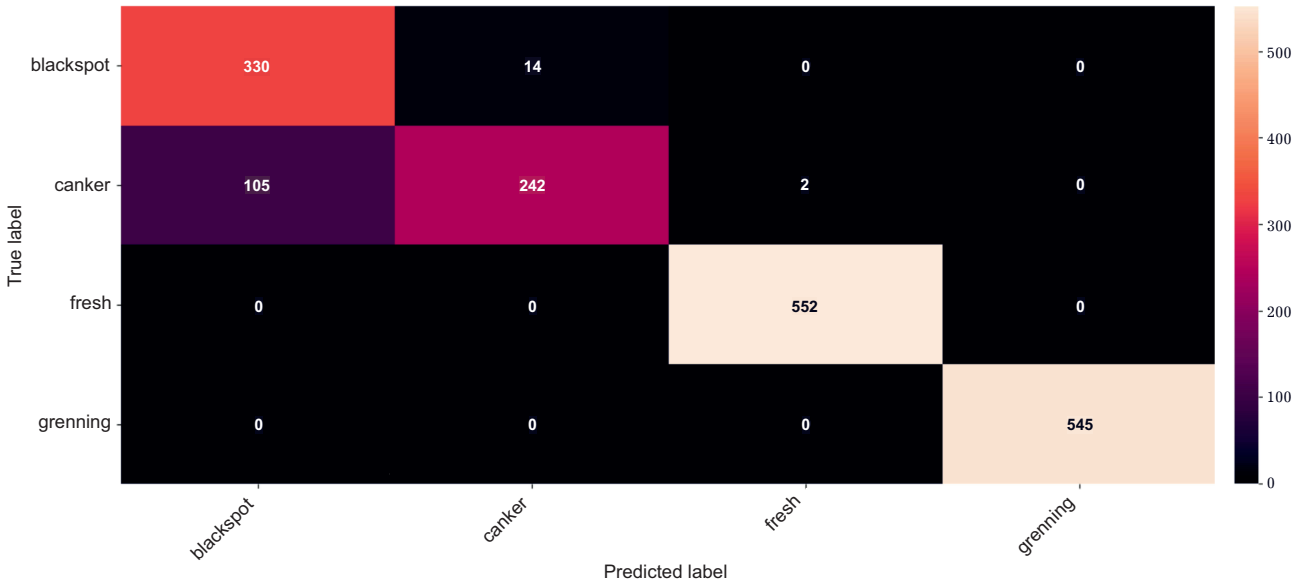


Fig. 7. The confusion matrix of CNN's model.

Table 4. Cross-validation results.

Fold	Accuracy (%)	F1-score	AUC
Fold 1	0.8938	0.8361	0.9861
Fold 2	0.8770	0.7817	0.9788
Fold 3	0.8379	0.7728	0.9733
Fold 4	0.8770	0.8568	0.9719
Fold 5	0.8379	0.7536	0.9902
Fold 6	0.8324	0.7536	0.9771
Fold 7	0.8044	0.6613	0.9687
Fold 8	0.7597	0.6333	0.9573
Fold 9	0.9385	0.9333	0.9890
Fold 10	0.8882	0.8332	0.9749
Average	0.8547	0.7821	0.9767

its predictions. These results underscore the ResNet50 model's exceptional accuracy and efficiency, making it a powerful choice for the given task.

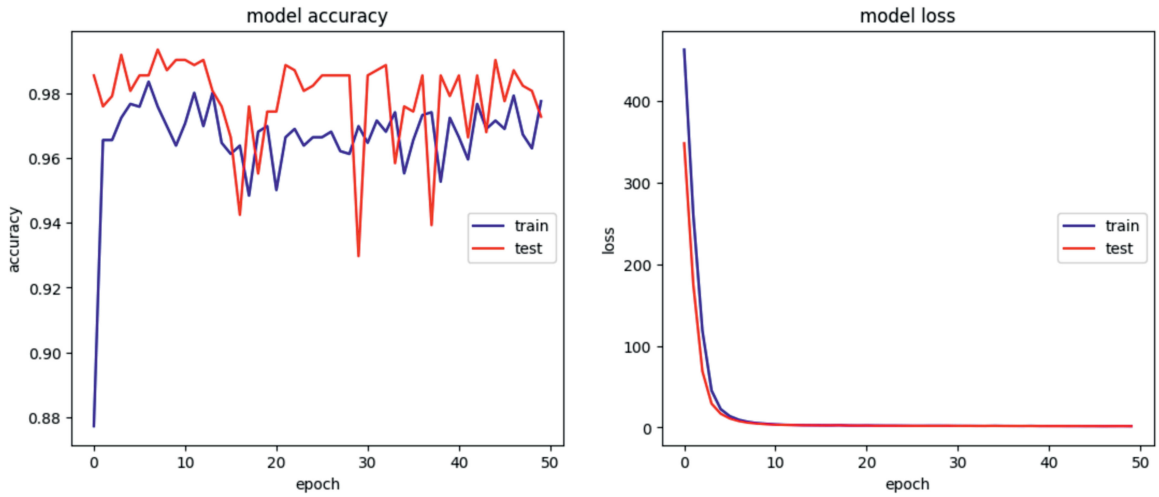


Fig. 8. Accuracy curve and loss curve of the ResNet50 model.

The table presents a detailed summary of the ResNet model's performance, showcasing the accuracy, recall, and F1-score for each class. Notably, the model achieved remarkable results across all classes, with an accuracy of 0.92 at 1.00.

In the confusion matrix, ResNet50 exhibited two misclassifications where the true label was "Blackspot", but the model incorrectly predicted "Canker". These limited misclassifications suggest that the model's performance in distinguishing different orange crop diseases remains highly accurate.

Table 5. Performance metrics of the ResNet50 model.

Class	Accuracy	Recall	F1-Score	Support
Blackspot	0.92	0.97	0.95	344
Canker	0.97	0.92	0.94	349
Fresh	1.00	1.00	1.00	552
Grenning	1.00	1.00	1.00	545

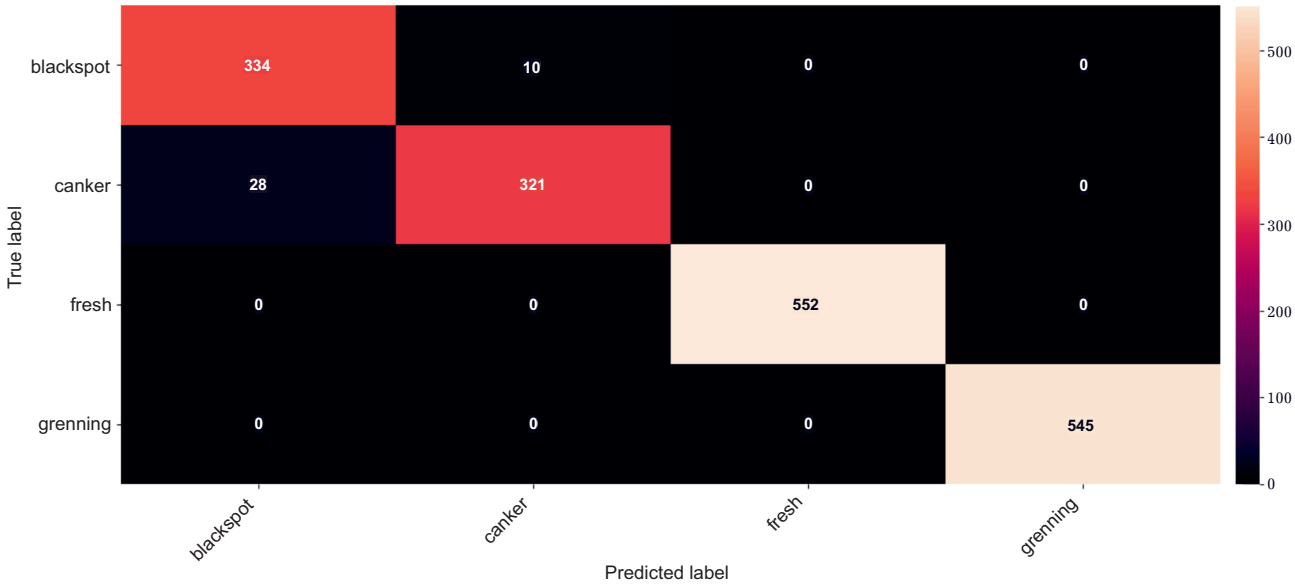


Fig. 9. ResNet50 Confusion Matrix.

The cross-validation results table provides a detailed view of the robustness of our model across different folds. Each row represents a specific fold, and metrics such as accuracy, F1-score, and AUC offer a comprehensive evaluation of the model's performance. Some folds demonstrated outstanding excellence, achieving perfect accuracy and F1-score of 100%. These results indicate that the model successfully generalized across diverse datasets. The overall average confirms the consistency of these high performances, instilling confidence in the model's ability to maintain outstanding performance under various conditions.

Table 6. Cross-validation results.

Fold	Accuracy (%)	F1-score	AUC
Fold 1	0.9832	0.9791	0.9991
Fold 2	0.9944	0.9950	1
Fold 3	1	1	1
Fold 4	0.9858	0.8568	0.9998
Fold 5	0.9921	0.7536	0.9997
Fold 6	1	1	1
Fold 7	0.9924	0.6613	0.9977
Fold 8	0.9932	0.6333	1
Fold 9	0.9801	0.9333	0.9998
Fold 10	1	1	1
Average	0.9932	0.9918	0.9996

4.3. Comparative analysis

Table 7 provides a detailed comparison of the accuracy of two models, CNN and ResNet50, across different classes. In the "Blackspot" class, ResNet50 outperforms CNN with an accuracy of 0.92 compared to CNN's 0.76. This improvement can be attributed to ResNet50's deeper architecture and skip connections, allowing it to capture more complex features and gradients effectively.

Moving to the "Canker" class, ResNet50 continues to show superiority with an accuracy of 0.97, surpassing CNN's accuracy of 0.95. The deeper and more sophisticated architecture of ResNet50 enables it to learn intricate patterns and nuances in the data, contributing to its enhanced performance.

Both models exhibit exceptional accuracy of 1.00 in the "Fresh" and "Grenning" classes. In these cases, both CNN and ResNet50 successfully classify instances with perfect accuracy, indicating their proficiency in recognizing distinct features of these classes.

Table 7. Comparative accuracy of models.

Class	CNN	ResNet50
Blackspot	0.76	0.92
Canker	0.95	0.97
Fresh	1.00	1.00
Grenning	1.00	1.00

The overall superior performance of ResNet50 over CNN can be attributed to its ability to mitigate the vanishing gradient problem through the use of residual connections. These connections facilitate the flow of gradients during backpropagation, enabling the model to learn more efficiently, especially in the case of deep networks. Consequently, ResNet50 demonstrates a higher accuracy across multiple classes, making it a more effective choice for the given task compared to the standalone CNN model.

The results also highlight a significant difference between the CNN and ResNet50 models in terms of training time. The CNN model converged more quickly, with a training time of 53.3754 seconds, while the ResNet50 model required 1139.9025 seconds for training. Despite this time difference, ResNet50 showed superior performance compared to CNN. This observation underscores ResNet50's ability to achieve higher accuracy, even though it required a longer training time.

5. Conclusion

In summary, this article sheds light on the critical challenges facing agriculture due to environmental crises and dwindling water resources, focusing specifically on the importance and vulnerability of oranges. It introduces an innovative approach that combines deep learning techniques like Convolutional Neural Networks (CNNs) with transfer learning (TL) and the use of combined L_1 and L_2 regularization to enhance the accuracy of disease classification in orange crops, even when data is limited. This fusion of methodologies offers a robust tool for effectively managing diseases in water-scarce regions, potentially revolutionizing disease detection and contributing to sustainable citrus cultivation practices.

The practical implications and future directions in Agricultural Informatics are poised to reshape the agriculture sector. The adoption of informatics solutions is yielding tangible benefits such as optimized operations, reduced resource wastage, and improved crop yields. Precision agriculture, combined with advanced regularization techniques, enhances both productivity and environmental sustainability. Early disease detection and data-driven management curb crop losses and minimize reliance on chemicals. Digitized market access and data-driven decision-making empower farmers and reduce intermediary costs. Looking ahead, advanced AI, machine learning, IoT, and blockchain will drive further innovation, fostering resilient, sustainable, and globally connected agriculture.

[1] Zhang D., Sial M. S., Ahmad N., Filipe A. J., Thu P. A., Zia-Ud-Din M., Caleiro A. B. Water scarcity and sustainability in an emerging economy: A management perspective for future. Sustainability. **13** (1), 144 (2020).

[2] Li S., Wu F., Duan Y., Singerman A., Guan Z. Citrus greening: Management strategies and their economic impact. HortScience. **55** (5), 604–612 (2020).

[3] Chlingaryan A., Sukkarieh S., Whelan B. Machine learning approaches for crop yield prediction and nitrogen status estimation in precision agriculture: A review. Computers and Electronics in Agriculture. **151**, 61–69 (2018).

[4] Fuentes A. F., Yoon S., Lee J., Park D. S. High-performance deep neural network-based tomato plant diseases and pests diagnosis system with refinement filter bank. Frontiers in Plant Science. **9**, 1162 (2018).

[5] Saha R., Neware S. Orange fruit disease classification using deep learning approach. International Journal of Advanced Trends in Computer Science and Engineering. **9** (2), 2297–2301 (2020).

[6] Dhiman P., Kaur A., Hamid Y., Alabdulkreem E., Elmannai H., Ababneh N. Smart Disease Detection System for Citrus Fruits Using Deep Learning with Edge Computing. Sustainability. **15** (5), 4576 (2023).

[7] Chen J., Chen J., Zhang D., Sun Y., Nanehkaran Y. A. Using deep transfer learning for image-based plant disease identification. Computers and Electronics in Agriculture. **173**, 105393 (2020).

[8] Zhang Y., Liu Y. P. Identification of navel orange diseases and pests based on the fusion of DenseNet and self-attention mechanism. Computational Intelligence and Neuroscience. **2021**, 5436729 (2021).

[9] Barman U., Choudhury R. D., Sahu D., Barman G. G. Comparison of convolution neural networks for smartphone image based real time classification of citrus leaf disease. Computers and Electronics in Agriculture. **177**, 105661 (2020).

[10] Vasconez J. P., Delpiano J., Vougioukas S., Cheein F. A. Comparison of convolutional neural networks in fruit detection and counting: A comprehensive evaluation. Computers and Electronics in Agriculture. **173**, 105348 (2020).

[11] Alom M. Z., Taha T. M., Yakopcic C., Westberg S., Sidike P., Nasrin M. S., Hasan M., Van Essen B. C., Awwal A. A., Asari V. K. A state-of-the-art survey on deep learning theory and architectures. Electronics. **8** (3), 292 (2019).

[12] Calin O. Deep Learning Architectures – A Mathematical Approach. Springer Series in the Data Sciences, Springer (2020).

[13] Wiatowski T., Bölcskei H. A mathematical theory of deep convolutional neural networks for feature extraction. IEEE Transactions on Information Theory. **64** (3), 1845–1866 (2017).

[14] Cao J., Pang Y., Li X., Liang J. Randomly translational activation inspired by the input distributions of ReLU. Neurocomputing. **275**, 859–868 (2018).

[15] Sun R. Optimization for deep learning: theory and algorithms. Preprint arXiv:1912.08957 (2019).

[16] Zhang J. Gradient descent based optimization algorithms for deep learning models training. Preprint arXiv:1903.03614 (2019).

[17] Yang Q., Zhang Y., Dai W., Pan S. J. Transfer Learning. Cambridge University Press (2020).

[18] He K., Zhang X., Ren S., Sun J. Deep residual learning for image recognition. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 770–778 (2016).

Кероване даними поєднання глибокого навчання та трансферного навчання для класифікації помаранчевої хвороби

Сгір А.[1], Зіані М.[1], Ель Хандрі К.[1,2,3]

[1] *Лабораторія LMSA, кафедра математики, факультет природничих наук, Університет Мухаммеда V у Рабаті, Марокко*
[2] *Школа штучного інтелекту і даних для бізнесу та суспільства Айвансіті, Франція*
[3] *Факультет медицини та фармації Університету Мохаммеда V у Рабаті, Лабораторія медичної біотехнології (MedBiotech)*

У сільському господарстві раннє виявлення хвороб сільськогосподарських культур є обов'язковим для сталого розвитку та максимізації врожайності. Наш інноваційний підхід, заснований на Agriculture 4.0, поєднує попередньо підготовлені моделі згорткових нейронних мереж (CNN) із розв'язками на основі даних для вирішення глобальних проблем, пов'язаних із дефіцитом води. Інтегруючи комбіновану техніку регулярізації L_1/L_2 до шарів нашої моделі, покращуємо їхню гнучкість, зменшуючи ризик ефекту перенавчання моделі. У наборі помаранчевих даних, який використовувався в наших експериментах, маємо 1790 зображень помаранчевого кольору, включаючи клас свіжих помаранч і три категорії захворювань. Застосовуючи цей набір даних для класифікації, наша модель демонструє помітну ефективність, а саме 92.17% для CNN та 97.28% для моделі ResNet-50. Оцінюючи такі показники, як точність, достовірність, повнота, показник F1, матриця невідповідностей та перехресна затвердження, наш підхід перевершує традиційні класифікатори, суттєво сприяючи стійкості інтелектуального сільського господарства та глобальної продовольчої промисловості в умовах зростаючого тиску дефіциту води.

Ключові слова: *багатокласова класифікація; помаранчева хвороба; дефіцит води; Agriculture 4.0; керований даними; глибоке навчання; згорткові нейронні мережі; трансферне навчання; ResNet-50.*