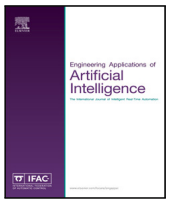
Engineering Applications of Artificial Intelligence 139 (2025) 109628

Contents lists available at ScienceDirect

Engineering Applications of Artificial Intelligence

journal homepage: www.elsevier.com/locate/engappai



Research paper

Model-agnostic local explanation: Multi-objective genetic algorithm explainer

Hossein Nematzadeh [a,*], José García-Nieto [a,b], Sandro Hurtado [a], José F. Aldana-Montes [a,b], Ismael Navas-Delgado [a,b]

[a] ITIS Software, Universidad de Málaga, Arquitecto Francisco Peñalosa 18, Málaga, 29071, Spain
[b] Departamento de Lenguajes y Ciencias de la Computación, Universidad de Málaga, Málaga, Spain

ARTICLE INFO

Keywords:
Citrus diseases
Multi-objective genetic algorithm explainer
Non-dominated Sorting Genetic Algorithm II
Residual network 50 layers
Local Interpretable Model-agnostic Explanations
Melanoma detection

ABSTRACT

Late detection of plant diseases leads to irreparable losses for farmers, threatening global food security, economic stability, and environmental sustainability. This research introduces the Multi-Objective Genetic Algorithm Explainer (MOGAE), a novel model-agnostic local explainer for image data aimed at the early detection of citrus diseases. MOGAE enhances eXplainable Artificial Intelligence (XAI) by leveraging the Non-dominated Sorting Genetic Algorithm II (NSGA-II) with an adaptive Bit Flip Mutation (BFM) incorporating densify and sparsify operators to adjust superpixel granularity automatically. This innovative approach simplifies the explanation process by eliminating several critical hyperparameters required by traditional methods like Local Interpretable Model-Agnostic Explanations (LIME). To develop the citrus disease classification model, we preprocess the leaf dataset through stratified data splitting, oversampling, and augmentation techniques, then fine-tuning a pre-trained Residual Network 50 layers (ResNet50) model. MOGAE's effectiveness is demonstrated through comparative analyses with the Ensemble-based Genetic Algorithm Explainer (EGAE) and LIME, showing superior accuracy and interpretability using criteria such as numeric accuracy of explanation and Number of Function Evaluations (NFE). We assess accuracy both intuitively and numerically by measuring the Euclidean distance between expert-provided explanations and those generated by the explainer. The appendix also includes an extensive evaluation of MOGAE on the melanoma dataset, highlighting its versatility and robustness in other domains. The related implementation code for the fine-tuned ResNet50 and MOGAE is available at https://github.com/KhaosResearch/Plant-disease-explanation.

1. Introduction

Citrus, including tangerine, orange, grapefruit, pomelo, lemon, and lime, are a group of fruits that grow in tropical areas and are rich sources of vitamin C. Citrus fruits are vulnerable to plant diseases such as black spot, canker, and greening (Rauf et al., 2019). Early detection of these diseases in the leaves of the trees leads to prompt treatment of the crops and can reduce financial losses for farmers by preventing the diseases from spreading in the orchards (Dananjayan et al., 2022).

Deep learning, and in special cases, Convolutional Neural Networks (CNNs), are machine learning techniques that provide robust image classifications in computer vision. Although CNNs can be developed from scratch, pre-trained networks are more commonly used. This is because pre-trained networks have previously learned to extract important features using multiple CNN layers on various classes. Many researchers prefer starting with pre-trained networks and adding new layers to customize the network for a specific classification task, including citrus disease classification (Elaraby et al., 2022; Çetiner, 2022). Some famous pre-trained networks include Inception (Khan et al., 2024), Xception (Shaheed et al., 2022), DenseNet (Zhu et al., 2023), ResNet (Giuseppe, 2021; Nematzadeh et al., 2023), VGG (Elaraby et al., 2022), ALexNet (Elaraby et al., 2022), MobileNet (Chen et al., 2021), and EfficientNet (Marques et al., 2020).

In recent years, explaining the prediction offered by machine learning algorithms has become inevitable, especially for complex algorithms such as the family of deep learning and ensemble-based methods (Barredo Arrieta et al., 2020; Yang et al., 2022). Explainers can be divided into model-agnostic and model-specific. Model-agnostic explainers are independent of the specific architecture of training models. In contrast, model-specific explainers use some details of the structure of the machine learning model. Thus, model-agnostic explainers can be

* Corresponding author.
E-mail addresses: hnematzadeh@uma.es, hn_61@yahoo.com (H. Nematzadeh), jnieto@uma.es (J. García-Nieto), sandrohr@uma.es (S. Hurtado), jfaldana@uma.es (J.F. Aldana-Montes), ismael@uma.es (I. Navas-Delgado).

https://doi.org/10.1016/j.engappai.2024.109628
Received 12 June 2024; Received in revised form 22 September 2024; Accepted 4 November 2024
Available online 20 November 2024

used to explain any machine learning training models, while model-specific explainers are suitable for the training models they are devised for. Local Interpretable Model-Agnostic Explanations (LIME) (Ribeiro et al., 2016) and Shapley Additive exPlanations (SHAP) (Lundberg and Lee, 2017) are the most well-known and practical model-agnostic explainers. Hurtado et al. (2022) compared the performance of LIME and SHAP on melanoma detection images dataset experimentally. SHAP has a model-agnostic kernel explainer that is extremely slow. Thus, various SHAPs optimized for explicit machine learning algorithms have been proposed. SHAP has two famous explainers for deep learning: gradient explainer and deep explainer.

Hurtado et al. (2022) experimentally showed how deep explainer loosed practicality by increasing the number of observations in background samples. Finally, this study concluded that LIME was faster than the gradient explainer in the melanoma detection dataset. Nematzadeh et al. (2023) proposed Ensemble-based Genetic Algorithm Explainer (EGAE), which has also been applied to the melanoma images dataset. The main objective of EGAE was to automate the image segmentation process of LIME. EGAE also increased the accuracy of explanation in comparison with LIME as well. EGAE leveraged multiple GAs (each with a different number of image segmentations) where final explanations were based on both consensus and majority voting of the GAs' results.

Gradient-Weighted Class Activation Map (Grad-CAM) (Selvaraju et al., 2017) is the most famous model-specific explainer among existing Convolutional Neural Networks (CNNs) visualization techniques (such as Vanilla Gradient, SmoothGrad, and Integrated Gradients) and is explicitly developed for CNN models. Grad-CAM is proposed to overcome the two limitations of CAM (Zhou et al., 2015). CAM is only applicable to Global Average Pooling (GAP) layers and does not support fully connected layers. Therefore, Grad-CAM supports any CNN models.

In this sense, several works have investigated explainability methods in agriculture. Bengamra Bidouk et al. (2023) proposed an eXplainable Artificial Intelligence (XAI) saliency method to elucidate the detection of potato diseases. Similarly, Bhandari et al. (2023) employed Grad-CAM and LIME to enhance model interpretability for identifying tomato leaf diseases. Meanwhile, Nahiduzzaman et al. (2023) utilized SHAP to provide explainability for deep learning models in the automatic classification of mulberry leaf diseases. Mehedi et al. (2022) not only used LIME to establish a more reliable prediction model for plant leaf disease detection but also offered clear explanations for the decisions made by the model. Furthermore, Wei et al. (2022) conducted a comprehensive evaluation, applying SmoothGrad, LIME, and Grad-CAM to a dataset featuring leaves affected by fruit diseases. However, most existing works need to comprehensively assess the quality of results in terms of accuracy and efficiency. Moreover, the continuous improvement of explainability techniques must contribute to artificial intelligence systems' broader acceptance and applicability across diverse contexts. Hence, this motivates us to propose a novel Multi-Objective Genetic Algorithm Explainer (MOGAE) utilizing Non-dominated Sorting Genetic Algorithm-II (NSGA-II) as a local image explainer to enhance the performance of the current explainability methods. To the best of our knowledge, MOGAE represents the first attempt to use NSGA-II as the core optimization algorithm to construct a model-agnostic explainer for image datasets. It follows EGAE (Nematzadeh et al., 2023) as the second attempt involving an evolutionary algorithm for creating a model-agnostic explainer for image datasets. In this regard, this paper extends and builds upon the work of Nematzadeh et al. (2023).

Consequently, a deep learning model is also developed to classify the imbalanced citrus diseases dataset accurately. Before classification, the dataset is partitioned into training, validation, and test sets using stratified data splitting. Subsequently, the data undergo oversampling and augmentation. The proposed deep learning model is based on a pre-trained ResNet50 model with additional fine-tuned layers for improved prediction. Finally, we apply the proposed MOGAE, along with EGAE (Nematzadeh et al., 2023) and LIME (Ribeiro et al., 2016), to gain insights into the prediction results for the citrus leaf diseases dataset (discussed in the paper) and the melanoma dataset (detailed in the Appendix). These explainers are compared based on their intuitive and numerical accuracy of explanations and the number of images they use for the explanation. Hence, the contributions of this research are as follows:

1. Proposing a Multi-Objective Genetic Algorithm Explainer (MOGAE) for the local explanation, enhancing the performance of EGAE (Nematzadeh et al., 2023).
    - MOGAE employs NSGA-II with the proposed adaptive Bit Flip Mutation (BFM), automatically adjusting the granularity of active superpixels in running time using densify and sparsify operators within a single run. Additionally, MOGAE utilizes optimal solutions from the Pareto front of NSGA-II for majority voting. In contrast, EGAE relies on automatic image segmentation by restarting GA multiple times, each with a different number of superpixels. As a result, MOGAE achieves a proper explanation using a few evaluated images with better accuracy in some scenarios.
2. Proposing a deep learning model for classifying citrus leaf diseases images using a fine-tuned, pre-trained ResNet50 model.
3. Applying MOGAE, EGAE, and LIME to experimentally compare each explainer's output for individual test samples on the citrus leaf diseases dataset (discussed in the paper) and the melanoma dataset (detailed in the Appendix), demonstrating the versatility of MOGAE across different scenarios.

The remainder of the paper is organized as follows.: Section 2 briefly introduces model-agnostic and model-specific explainers, along with recent related works on them. In Section 3, we explore the reasons behind using multi-objective optimization as an alternative approach to developing a local image explainer. Section 4 introduces the deep learning model for the citrus leaf diseases dataset. Following that, we provide a detailed explanation of Multi-Objective Genetic Algorithm Explainer (MOGAE). Section 5 investigates how MOGAE, EGAE, and LIME differ in their explanation results on the citrus leaf diseases dataset. The Appendix of this paper, which follows the conclusion in Section 6, provides extensive experiments of MOGAE on the melanoma detection dataset.

2. Related works

In this section, we provide brief explanations of model-agnostic explainers (LIME Ribeiro et al., 2016 and EGAE Nematzadeh et al., 2023) and model-specific explainers (Grad-CAM Selvaraju et al., 2017 and SHAP gradient explainer Lundberg and Lee, 2017), and works inspired by each of these explainers. Later in Section 5, EGAE and LIME will be applied to the prediction results of the proposed fine-tuned ResNet50 (Residual Network), and their explanations will be compared with the proposed MOGAE. As a brief explanation for ResNet50 (Giuseppe, 2021), the model is initially designed to classify 1000 classes from the ImageNet dataset. Moreover, the model also has good top-1 and top-5 accuracy on the ImageNet validation dataset. In addition, ResNet50 comprises over 25 million trainable parameters, which makes it appropriate for image classification and computer vision problems.

2.1. Model-agnostic explainers

Local Interpretable Model-Agnostic Explanations (LIME) (Ribeiro et al., 2016; Nematzadeh et al., 2023) generally works by minimizing the approximation of $\sigma(x)$ in Eq. (1) in which x is the selected sample for explanation. Likewise, π_x is the set of samples in the vicinity of x for the explanation the user should specify. The more significant

number of π_x directly affects the results' better reproducibility, mainly when the input image is segmented into a great number of superpixels. Remember that a superpixel is a cluster of pixels grouped together based on certain characteristics, such as color or texture. However, the side effect of such a great set of π_x is the burden in execution time. The basis of LIME is that we cannot make any assumption on f (the actual training model). Thus, any interpretable model in G (such as linear regression, decision trees, etc.) can be theoretically used for g as the explainer of f. In fact, $\Omega(g)$ should be low enough to be interpretable by humans. Generally, each member of π_x is assigned a weight based on the distance to x. Likewise, the prediction of f on π_x regarding the target class should be calculated. Then, an interpretable model in G is selected for explanation. Assuming $g \in G$ is a linear regression, a linear model can be fitted using π_x, the predictions of f over π_x, and the set of weights for π_x members. The coefficients of linear regression can be used to understand the contribution of each image superpixel on the prediction of the target class.

$$\sigma(x) = \underset{g \in G}{\operatorname{argmin}} L(f, g, \pi_x) + \Omega(g) \tag{1}$$

The image explainer in LIME faces certain challenges. Therefore, the literature contains numerous works aimed at enhancing LIME's capabilities and addressing potential challenges by adding value to the existing version of LIME including Anchor LIME (Ribeiro et al., 2018), KL-LIME (Peltola, 2018), NormLIME (Ahern et al., 2019), LIME-Aleph (Rabold et al., 2020), MPS-LIME (Shi et al., 2020), SurvLIME (Kovalev et al., 2020), LIMEcraft (Hryniewska et al., 2022), and our recent evolutionary-based contribution, Ensemble-based Genetic Algorithm Explainer (EGAE) (Nematzadeh et al., 2023). Notably, EGAE is the first model-agnostic explainer designed for image datasets, utilizing evolutionary algorithms, as far as our knowledge extends. EGAE (Nematzadeh et al., 2023) automated the image segmentation in LIME, which was previously done manually by defining the number of superpixels for the input image as a hyperparameter. EGAE addressed this limitation by employing multiple Genetic Algorithms (GAs), each with a unique number of superpixels. Each GA generated an explanation. EGAE leveraged these explanations and ultimately produced two explanations for the input image using consensus and majority voting. Likewise, EGAE eliminated the need to specify the top features to be displayed. It automatically identified the informative pixels of the image that positively contribute to the prediction and discarded the non-informative pixels.

Additionally, experimental results demonstrated that EGAE could generate more accurate explanations for the melanoma detection dataset than LIME. However, EGAE takes more time compared to LIME due to its evolutionary approach. Thus, let R_i represent the result (explanatory image) generated by the ith run of the Genetic Algorithm (GA) in Eq. (2) so that n_i is the number of genes in the chromosomes of the ith GA. CV checks if all pixels are 'on' in different GA results in Eq. (3) while MV checks if the majority are 'on' in different GA results in Eq. (4). In Eqs. (3) and (4), $P(x, R_i)$ determines whether pixel x is 'on' in result R_i. If the pixel is 'on', $P(x, R_i)$ equals 1; otherwise, it equals 0. Likewise, N is the total number of Genetic Algorithms (GAs). It is important to mention that, to avoid unexpected results, EGAE discarded the image(s) belonging to R that disrupted consensus voting. Therefore, in some cases, the size of R (and equivalently N) may be less than the total number of GAs.

$$R_i = GA\_Explainer(n_i) \tag{2}$$

$$CV(x) = \begin{cases} 1, & if \sum_{i=1}^{N} P(x, R_i) = N \\ 0, & otherwise \end{cases} \tag{3}$$

$$MV(x) = \begin{cases} 1, & if \sum_{i=1}^{N} P(x, R_i) \geq \frac{N}{2} + 1 \\ 0, & otherwise \end{cases} \tag{4}$$

The successful application of GA for explaining the prediction model on the melanoma detection dataset in Nematzadeh et al. (2023) served as inspiration for our work in this paper, where we aim to accomplish the following:

- To evaluate the application of EGAE on the citrus leaf diseases dataset.
- To propose a Multi-Objective Genetic Algorithm Explainer (MO-GAE).

2.2. Model-specific explainers

Gradient-Weighted Class Activation Map (Grad-CAM) (Selvaraju et al., 2017) uses the gradient of the last CNN layer to unveil the black box of deep learning. Regions with higher gradients indicate greater importance for the model's prediction. Initially, Grad-CAM calculates the derivative of the winning class (y^c) with respect to each of the activation maps (A^k) in Eq. (5). A partial derivative exists for any neurons within an activation map so that A^k_{ij} is the gradient calculated via back propagation. Next, the average of all gradients together is calculated using global average pooling in Eq. (6), assuming each feature map has the size of $z = i \times j$. In fact, Eq. (6) converts the $i \times j \times k$ gradients to k gradients. α^c_k implicitly denotes the importance level of each of the feature maps so that c is the winning class and k are the feature maps. The last step for heatmap visualization of the Grad-CAM is to multiply α^c_k (which is equivalent to the weights) by feature maps within a ReLU function in Eq. (7). The ReLU function in Eq. (7) keeps only the positive weights (α^c_k). Grad-CAM is specifically designed for CNN models and cannot be used for other machine learning algorithms.

$$\frac{\partial y^c}{\partial A^k_{ij}} \tag{5}$$

$$\alpha^c_k = \frac{1}{z} \sum_i \sum_j \frac{\partial y^c}{\partial A^k_{ij}} \tag{6}$$

$$L^c_{Grad-CAM} = ReLU\left(\sum_k \alpha^c_k \times A^k \right) \tag{7}$$

In this regard, the literature contains many works on applying Grad-CAM on CNNs to identify the model's important features, as shown in Kim et al. (2023) and Wu and Zhao (2022). To address the limitations of Grad-CAM, Li et al. (2023) observed that the feature resolution of Grad-CAM decreases as network layers deepen when applied to vibration signals in machine fault diagnosis. Consequently, Li et al. (2023) introduced a Multi-Layer Grad-CAM (MLG-CAM) approach, enhancing the activation map's resolution and explanatory power.

The Gradient Explainer explains the model using expected gradients by extending shapely values to player game theory. Unlike Kernel Shapley Additive exPlanations (Kernel SHAP), a model-agnostic estimation approach for Shapley values, Gradient Explainer is specifically designed to explain CNN models. Shapley's value works by calculating the marginal contribution of features and weighting them. The more important features have greater SHAP values. Finally, feature attributions should sum to the prediction value. These attributions can be negative or positive since a feature can lower or raise a predicted value. The general formulation of shapely is to create the power set of n features. Thus, SHAP requires exponentially massive training of the 2^n models. Lundberg and Lee (2017) proposed some approximations and sampling as workarounds. Eq. (8) shows the general SHAP values calculation so that the marginal contribution of each feature along with the respective weights are calculated. Here, F is the total number of features, and f represents an individual feature within the $set(1, 2, \ldots, F)$.

$$SHAP_{f(x)} = \sum_{f \in set} \left[|set| \times \binom{F}{|set|}^{-1} \left(Predict_{set}(x) - Predict_{set/f}(x) \right) \right] \tag{8}$$

In this regard, Hurtado et al. (2022) investigated the strength of the SHAP Gradient Explainer on the melanoma detection dataset in terms of reproducibility and execution time. This research concluded that the SHAP Gradient Explainer performed slower than LIME on melanoma images.

Table 1
Comparison of image explainers with local explanation on classification data.

Image explainer	Scope of explanation	Model dependency	Attribution of features	Approach of explanation	Evolutionary algorithms	Year
LIME (Ribeiro et al., 2016)	Local	Model-agnostic	Positive and negative	Manual image segmentation	Not applicable	2016
EGAE (Nematzadeh et al., 2023)	Local	Model-agnostic	Positive	Automatic image segmentation	Genetic algorithm	2023
Grad-CAM (Selvaraju et al., 2017)	Local	Model-specific	Positive	Gradients	Not applicable	2017
SHAP Gradient (Lundberg and Lee, 2017)	Local	Model-specific	Positive and negative	Expected gradients	Not applicable	2017

2.3. Insights from literature

Table 1 categorizes the discussed explainers. Although there are other types of explainers, such as those in the rule extraction category (Hailesilassie, 2016), which, like RuleXAI (Macha et al., 2022), provide both local and global explanations, or Belief-Rule-Based (BRB) (Nimmy et al., 2023) designed to ensure the trustworthiness of interpreted time-series decisions, our research scope is specifically centered on comparing existing local image explainers for classification data. Thus, all of the image explainers in Table 1 belong to the local explanation category, meaning they are more suited and provide explanations for the predictions of a machine learning model for a specific input instance.

Furthermore, Table 1 also shows that both SHAP Gradient Explainer and LIME can identify positive or negative feature attributions. However, Grad-CAM, with its ReLU function, and EGAE are inherently optimized to highlight positive attributions. It is important to note that LIME and EGAE are model-agnostic, whereas Grad-CAM and SHAP Gradient Explainer are model-specific. Among these image explainers, EGAE is the only one that utilizes an evolutionary algorithm. Manual segmentation in LIME involves the expert specifying the number of superpixels. Automatic segmentation would mean the algorithm determines the number of superpixels without expert input, which can be more convenient. According to Table 1, the proposed MOGAE image explainer, which will be discussed in detail in Section 4.2, will provide local explanations and is model-agnostic. Additionally, like EGAE, MOGAE will only highlight image regions that positively contribute to the prediction. It achieves this by automatically adjusting the granularity of active superpixels through an adaptive Bit Flip Mutation that exploits densify and sparsify operators within NSGA-II. For a fair comparison, MOGAE will be evaluated alongside LIME and EGAE with majority voting as model-agnostic explainers using the citrus leaf diseases dataset in Section 5 and the melanoma dataset in Appendix.

3. Multi-objective optimization and explanation

In this section, we argue that, based on our hypotheses, multi-objective optimization using evolutionary algorithms can serve as an alternative method to develop a local image explainer.

According to multi-objective optimization (assuming the goal is maximization) in Eq. (9), the fitness function is represented as a vector of size m, unlike in single-objective optimization where the fitness function is a scalar value. The scalar nature of the fitness function in single-objective optimization is crucial for determining the relative quality of solutions and selecting the best one by comparing and ordering them. However, ordering becomes challenging when the fitness function is a vector.

$$\max_x f_i(x) \quad \forall i \in I, x \in X, I = \{1, 2, 3, \ldots, m\}, f : X \to \mathbb{R}^m \tag{9}$$

There are two alternatives for resolving multi-objective problems. The first approach involves decomposition techniques such as weighted sum, goal programming, goal attainment, and *epsilon*-constraint. However, a limitation of decomposition techniques is that different runs of these methods can result in different solutions.

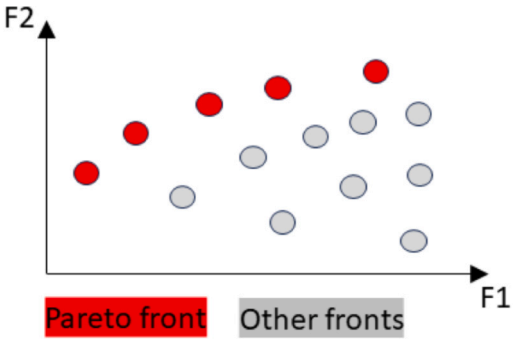


Fig. 1. Multi-objective optimization.

The second approach involves leveraging multi-objective evolutionary algorithms, which aim to produce a set of good solutions known as the Pareto front in Fig. 1, using minimal information about the problem domain. Unlike decomposition techniques, multi-objective evolutionary algorithms provide diverse solutions without being affected by the variability seen in decomposition techniques. These algorithms are particularly advantageous in solving complex, real-world problems where the objective functions are often conflicting and non-linear, making them a powerful tool in multi-objective optimization.

We identify three compelling reasons for using Multi-Objective Genetic Algorithm, specifically Non-dominated Sorting Genetic Algorithm-II (NSGA-II), as an effective method to create a local image explainer:

1. Assuming the image is segmented into H superpixels, the problem of finding informative superpixels — in the sense that the search space is exponential (2^H) — can become theoretically impossible as 2^H grows large and falls into the category of hard problems. Therefore, using evolutionary algorithms as a method of optimization is one alternative for solving this problem.
2. Additionally, our investigation has revealed no direct or indirect relationship between the model's prediction for a given class on a solution (F2 in Fig. 1) and the number of active superpixels (F1 in Fig. 1). Consequently, the problem of identifying informative superpixels can be formulated as a multi-objective optimization task.
3. Finally, as the problem of finding informative superpixels is intrinsically a discrete problem, and the Genetic Algorithm (GA) is a powerful optimization technique for discrete problems, we have decided to formulate the problem using the Multi-Objective Genetic Algorithm using NSGA-II.

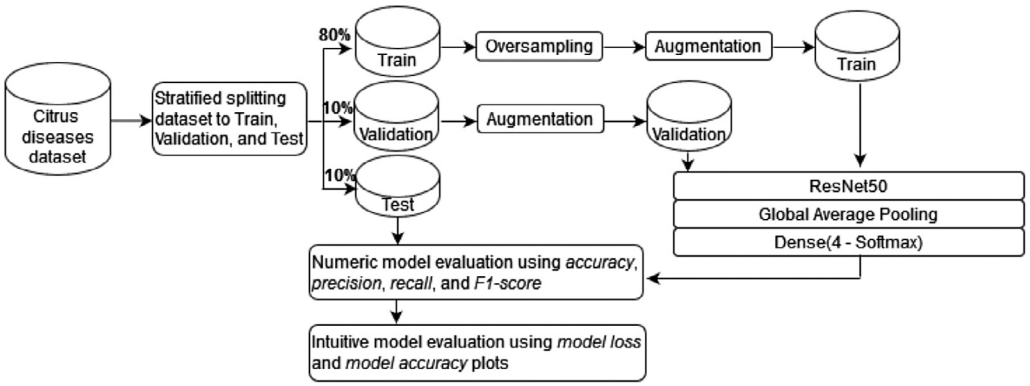


Fig. 2. The pipeline for developing the fine-tuned ResNet50 for classification of citrus leaf diseases.

This work represents a direct extension of our recent contribution, EGAE (Nematzadeh et al., 2023), and builds upon it by addressing the following improvements. EGAE utilized multiple GAs initialized with a different number of superpixels (ranging from small to large) to cover all possible settings for explanation. However, MOGAE automatically adjusts the granularity of active superpixels using the proposed adaptive Bit Flip Mutation (BFM) through densify and sparsify operators. Additionally, since EGAE utilized GA, it was necessary for EGAE to be run multiple times, each time with a different number of superpixels, to obtain diverse solutions. This was done to enable the application of voting strategies on the generated solutions. However, MOGAE will use NSGA-II and thus will utilize the optimal solutions from the Pareto front generated in a single run. As a result, MOGAE might potentially use fewer images to generate the explanatory image compared to EGAE, especially when dealing with a high number of input segmentations in certain scenarios.

4. Proposed method

Fig. 2 illustrates the pipeline for the proposed prediction model for classifying citrus leaf diseases. Initially, a pre-trained ResNet50 model is fine-tuned specifically for classification of the citrus leaf diseases dataset. The model's robustness is then assessed using accuracy, precision, recall, and F1-score metrics.

After the development of a sufficiently reliable prediction model, the Multi-Objective Genetic Algorithm Explainer (MOGAE) is presented, with its corresponding pipeline depicted in Fig. 3. Finally, the explanation of the proposed model shown in Fig. 2 is explored using the presented MOGAE, the Ensemble-based Genetic Algorithm Explainer (EGAE), and the Local Interpretable Model-agnostic Explanations (LIME).

As such, Section 4 is divided into two subsections. Section 4.1 provides the preprocessing steps applied to the citrus leaf diseases dataset. This includes details on stratified splitting, the oversampling algorithm, augmentation techniques, and the corresponding fine-tuned ResNet50 prediction model. Section 4.2 delves into the technical aspects of MOGAE.

4.1. Data preparation and model development

The citrus leaf diseases dataset under study has 596 samples of leaves with four classes of black spot, canker, greening, and healthy (Rauf et al., 2019). It is worth mentioning that the original dataset has five classes. The melanose class is excluded from the dataset and will not be used in this research. The reasons are very few images in the melanose class (13 images), the low quality of the images, and the inconsistency of images with other classes regarding the image background, dimensions, and leaves. Likewise, we suspect some minor mislabeling in existing classes. However, since the proposed model in Fig. 2 achieves good accuracy, we did not exchange the possibly mislabeled data. The dataset is split into train, validation, and test sets with proportions of 80%, 10%, and 10%, respectively. Since the dataset is imbalanced, as is evident in Table 2, the split strategy adheres to stratified sampling to initially preserve the distribution of classes in each training, validation, and test set. Second, the training set is oversampled to avoid bias and potential risks. The significant risk of an imbalanced train is that the accuracy most likely reflects the underlying class distribution in the training set. Oversampling balances the distribution of classes by adding new samples in minority classes. The new samples resulting from oversampling are identical to existing samples because oversampling duplicates the data as shown in Algorithm 1. Identical data do not add any information to the training set, and thus, data augmentation can generate slightly modified copies of existing data. The augmentation techniques we used are rotation, width shift, height shift, and horizontal flip. Additionally, rescaling is applied as a preprocessing step to normalize the pixel values. According to Fig. 2, the training set is oversampled and augmented. The validation set is also augmented as well. In contrast, oversampling and augmentation are not applied on the test set as unseen data.

Table 2
Specification of the citrus disease dataset (Rauf et al., 2019).

Data	Total observations	Black spot	Canker	Greening	Healthy
Train before balancing	487	137	131	164	46
Train after balancing	656	164	164	164	164
Validation	59	17	16	20	6
Test	59	17	16	20	6

Algorithm 1 Oversampling of minority classes

Input: *Train*, *Target*, *Class*, *Size*
Output: *NewTrain_x* , *NewTrain_y*
1: *UpsampleID* ← Find and save all indices for the *Class* to be upsampled
2: *NewTrain_x* ← All observations in *Train* except those in *UpsampleID*
3: *NewTrain_y* ← All targets in *Target* except those in *UpsampleID*
4: *New_x*, *New_y* ← Resample (*Train* [*UpsampleID*], *Target* [*UpsampleID*]) with the size of majority class *Size*
5: *NewTrain_x* ← *NewTrain_x* + *New_x*
6: *NewTrain_y* ← *NewTrain_y* + *New_y*
7: Return *NewTrain_x*, *NewTrain_y*

It is primarily investigated whether adding new dense or dropout layers could improve the prediction model's performance. Finally, it is experimentally concluded that the current model architecture is fine and adding extra layers could not increase the performance of the training model. The model architecture in Fig. 2 is trained, and the best weights are saved for reusability. The learning rate in the training model is reduced dynamically with the patience of 3 and factor of 0.1.

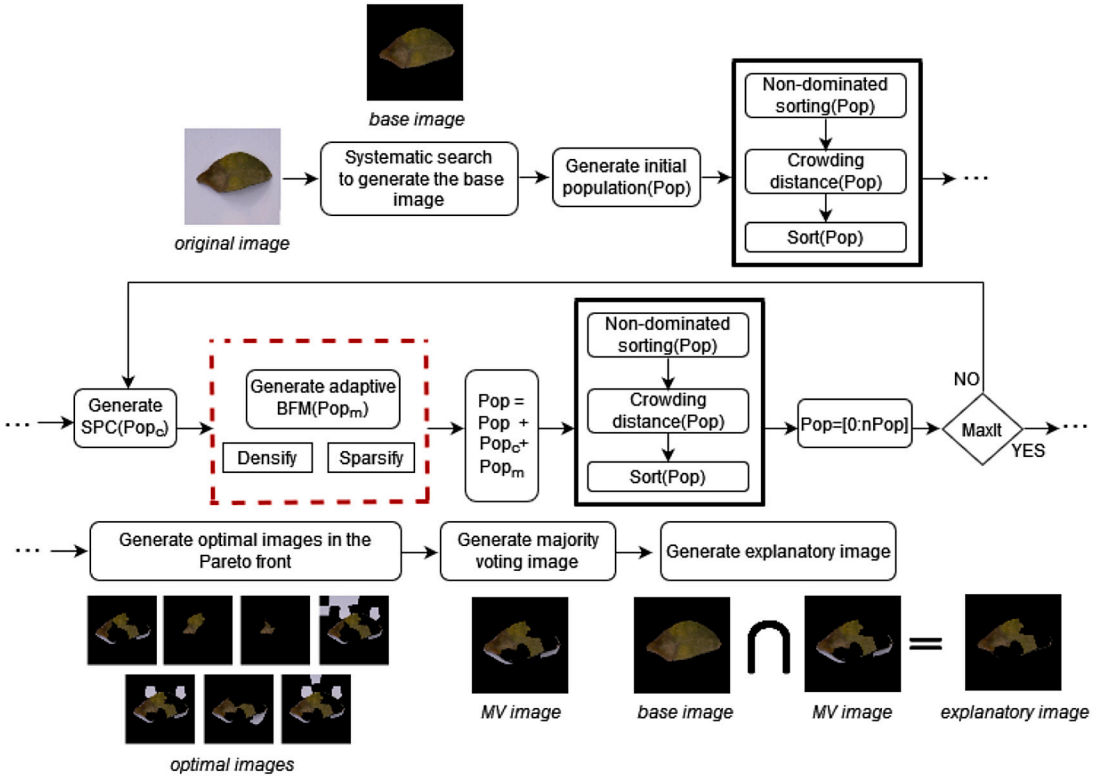


Fig. 3. The pipeline for developing MOGAE and a step-by-step visualization using an arbitrary leaf image with black spot disease.

The early stopping is tunned to stop training with the patience of 10 on no change in improving the validation loss. The model stops training if the validation loss is not improved in 10 consecutive epochs. The maximum number of epochs is set to 30. The citrus leaf diseases data have the size of (256, 256, 3) which explicitly means the images are colored. The ResNet50, by default, is set to accept input images of size (224, 224, 3). Thus, the input shape of ResNet50 is changed to (256, 256, 3) to process the citrus leaf diseases data. Furthermore, ResNet50 has 1000 classes in its last dense softmax layer. So, the output layer of ResNet50 should have been exchanged with a newly added softmax dense layer with four nodes accordingly.

The pseudo-code in Algorithm 1 shows how observations in a particular class are resampled. The input of Algorithm 1 are the entire observations in the training set (*Train*) and their respective class labels (*Target*), the minority class label that is intended to oversampling (*Class*) and the size of the majority class (*Size*). First, the observations belonging to the selected minority class for oversampling are identified in *UpsampleID* at Line 1. Then, the records in *UpsampleID* are excluded from the *Train* and *Target* to form *NewTrain_x* and *NewTrain_y* at lines 2–3. Next, the resample function duplicates the observations in *UpsampleID* up to the size of the majority class to form *New_x* and *New_y* at line 4. Finally, the oversampled observations and their respective class labels are added to *NewTrain_x* and *NewTrain_y* at Lines 5–6. The same procedure should be applied for the remaining minority classes as well, but this time *NewTrain_x* and *NewTrain_y* should be passed as *Train* and *Target* arguments. Likewise, the class argument (*Class*) should change accordingly. However, the size of the majority class does not change. The entire new observations obtained from oversampling (*NewTrain_x*, *NewTrain_y*) should be augmented. The augmentation in this research contains rotation range of 40, width shift range of 0.1, height shift range of 0.1, and horizontal flip activated. The training and validation sets are augmented, and the test set remains unchanged (neither oversampled nor augmented). Later, the outcome of the proposed model in Fig. 2 on individual test samples will be explained by EGAE, the presented MOGAE, and LIME in Section 5.4 after the training model has been sufficiently fitted.

4.2. MOGAE

According to Fig. 3, the Multi-Objective Genetic Algorithm Explainer (MOGAE) initially conducts a systematic search within a sufficiently small search space (2^h). The outcome of the systematic search is the *base image* with the fewest superpixels and the highest prediction accuracy for the given class. This *base image* offers valuable general insights into the region of the *original image* containing informative pixels.

Subsequently, a multi-objective optimization employing NSGA-II is formulated. NSGA-II is designed to generate optimal solutions within an extensive search space (2^H), where $H \gg h$. MOGAE employs adaptive Bit Flip Mutation (BFM), allowing active superpixels' granularity to be densified or sparsed in running time (on the fly) to prevent premature convergence. The outcome of MOGAE is several *optimal images* lie in the Pareto front. In the final step, majority voting (in addition to majority voting, other voting strategies can also be employed, such as consensus voting, which has been further investigated in the Appendix section) is applied to the valid images of the *optimal images*, generating the majority voting image (*MV image*). The *MV image* identifies informative superpixels that actively appear in the Pareto front solutions. While the *MV image* effectively explains the model, a deeper investigation is carried out to exclude potentially non-informative superpixels. This is accomplished by intersecting the *base image* with the *MV image*, resulting in the creation of the *explanatory image*. This image comprises only the most informative section, positively contributing to the prediction.

4.2.1. Systematic search

The systematic search aims to identify the specific area within an *original image* that contains the most relevant information for prediction. This process involves segmenting the *original image* into a small number of superpixels h (in this research, $h = 5$) using a segmentation algorithm (in this research, SLIC). Depending on the number of superpixels specified by the user (h), generating 2^h perturbations of an image is possible using the segmentation algorithm. Each perturbation is represented via a binary array so that 0 and 1 represent non-active

and active superpixels. Non-active superpixels are not displayed and remain in the background, while active superpixels are displayed. The goal is to identify the most informative perturbation (*base image* in Fig. 3) of an *original image* with the fewest active superpixels and the highest prediction accuracy ($\hat{y}$ or yhat), determined based on the real class label of the *original image*. Given the small value of h, conducting a systematic search utilizing Algorithm 2 to find the optimal perturbation based on fitness provided in Eq. (10) is feasible. α and β are weights experimentally set to 0.2 and 0.8, respectively. $Pred(P)$ shows the perturbation $\hat{y}$ corresponding to the *original image*'s class label. n_{as}^{P} is the number of active superpixels in the corresponding perturbation. h is also defined in Eq. (11) so that n_{ns}^{p} is the number of non-active superpixels in a given perturbation. Thus, a perturbation with one active superpixel ($n_{as}^{P} = 1$) will weigh $\frac{h}{h} = 1$ (highest). In contrast, a perturbation with all possible superpixels active ($h = n_{as}^{P}$) will weigh $\frac{1}{h}$ (lowest).

$$Fitness(P) = \alpha \times \left[\frac{h - n_{as}^{P} + 1}{h}\right] + \beta \times Pred(P) \tag{10}$$

$$h = n_{as}^{P} + n_{ns}^{P} \tag{11}$$

Algorithm 2 shows the steps to generate the *base image* such that temp_image becomes the resulting image after applying the perturbation t[i]. When the Algorithm 2 stops, temp_image is equal to base_image. This *base image* highlights the general informative area of the *original image*. In the next phase, MOGAE is employed to further pinpoint finer informative regions that theoretically should be located inside the *original image*.

Algorithm 2 Systematic search

Input: h, original_image
Output: base_image
1: Temp ← 0
2: superpixels ← SLIC(original_image, h)
3: t ← Create a list of bit strings representing numbers from 1 to 2^h -1
4: **for** i=1 to $2^h - 1$ **do**
5: active_pixels ← np.where(t[i] == 1)
6: mask ← np.zeros(superpixels.shape)
7: **for** active in active_pixels **do**
8: mask[superpixels == active] ← 1
9: **end for**
10: temp_image ← original_image * mask
11: temp_fit ← Calculate the fitness of the temp_image using Eq. (10)
12: **if** temp_fit > Temp **then**
13: Temp ← temp_fit
14: base_image ← temp_image
15: base_perturbation ← t[i]
16: **end if**
17: **end for**
18: Return base_image

4.2.2. Metaheuristic search

The metaheuristic search leverages the NSGA-II algorithm with an adaptive Bit Flip Mutation (BFM). This adaptive BFM dynamically adjusts the granularity of active superpixels in running time (on the fly) to prevent premature convergence by employing its densify and sparsify operators. By automatically adapting the granularity of active superpixels, NSGA-II aims to select the finest possible areas in the *original image* that positively contribute to the model's prediction. Consequently, the user does not need to determine the number of superpixels before generating explanations.

To identify the finest areas that positively contribute to the prediction, the *original image* should be segmented into a large number of superpixels (H), expanding the search space for the proposed NSGA-II to 2^H, where $H \gg h$ (in this research, $H = 50$). In the proposed NSGA-II, each candidate solution is of a struct type containing nine fields: Size, ValidAcc, Chromosome, Pred, Fitness, Rank, DominationSet, DominatedCount, CrowdingDistance, and Image. As such, "Size" represents the number of active superpixels in the solution (n_{as}^{S}), "ValidAcc" is a binary value (0 or 1) depending on the correspondence between the real class label of the solution and the calculated class label. If the real class label equals the calculated class label "ValidAcc" is set to 1; otherwise, it is set to 0. "Chromosome" represents the perturbation of the solution in the form of a binary bit string. "Pred" represents the model's prediction on the solution ($\hat{y}$) for a given class, and the "Fitness" of each solution is a vector defined according to Eq. (12). The value of H can be calculated using Eq. (13) which is identical to Eq. (11).

$$Fitness(S) = \left[\frac{H - n_{as}^{S} + 1}{H}, Pred(S)\right] \tag{12}$$

$$H = n_{as}^{S} + n_{ns}^{S} \tag{13}$$

The NSGA-II with adaptive BFM begins by creating the initial population Pop of size $nPop$ as described in Eq. (14) where x_i (Chromosome) is a vector of decision variables for the ith candidate solution, each with a size of H. Each variable in x_i is randomly generated as 0 or 1 with a $\Omega = 0.5$ probability.

$$Pop = x_1, x_2, \ldots, x_{nPop} \tag{14}$$

After generating each candidate solution, its fitness is calculated using Eq. (12). Each candidate solution corresponds to an image (Image) generated from the random binary array (Chromosome). Following this, non-dominated sorting and crowding distances are computed to sort the candidate solutions effectively. Consequently, Rank, DominationSet, DominatedCount, and CrowdingDistance for each candidate solution are initialized. In Algorithm 3 (multiple instructions in a line are separated by a semicolon, as at line 2), the Pareto front is calculated from lines 1–17, and other front members are computed at lines 18–35. Likewise, P1 dominates P2 at line 8 of Algorithm 3, if Eq. (15) holds.

$$P1 \succcurlyeq P2 \quad \Leftrightarrow \quad \forall_i : P1_i \geq P2_i, \quad \exists_{i_0} : P1_{i_0} > P2_{i_0} \tag{15}$$

Algorithm 4 utilizes Eqs. (16)–(17) to calculate the crowding distance of solutions within the same front. Given that our fitness function comprises two objective functions, we constrain Eqs. (16)–(17) to operate with two objectives. This approach ensures that the distance is computed individually for each objective in Eq. (16). Consequently, the summation of all distances for each objective in Eq. (17) results in the general crowding distance.

$$d_i^j = \frac{|Obj_j^{i+1} - Obj_j^{i-1}|}{Obj_j^{max} - Obj_j^{min}} \tag{16}$$

$$d_i = d_i^1 + d_i^2 \tag{17}$$

Now that ranks and crowding distances of solutions are computed, the solutions within the population can be sorted. Solutions with smaller ranks and greater crowding distances are considered better. In the implementation, solutions are first sorted based on crowding distances separately and then on ranks to ensure correct sorting, as shown in Algorithm 5. This will reflect the theory so that the solutions must be sorted by their rank and then by their crowding distance within each rank. In addition to the sorted population, Algorithm 5 also provides the Pareto front members and the corresponding best prediction accuracy, denoted as $\hat{y}$, for the intended class.

Until now, in the first generation, random solutions (Pop) have been generated and sorted based on computed ranks using non-dominated

Algorithm 3 Non-dominated sorting

Input: Pop
Output: Pop, Fronts

for each candidate solution P in Pop **do**
  P.DominationSet ← EmptySet; P.DominatedCount ← 0
end for
Fronts{1} ← []
for i=1 to nPop **do**
  for j=i+1 to nPop **do**
    P1 ← Pop[i]; P2 ← Pop[j]
    if P1 Dominate P2 **then** ▷ Apply changes if P1 dominates P2 or vice versa
      P1.DominationSet.append(j)
      P2.DominatedCount ← P2.DominatedCount+1
    end if
    Pop[i] ← P1; Pop[j] ← P2
  end for
  if Pop[i].DominatedCount==0 **then**
    Fronts{1}.append(i); Pop[i].rank=1
  end if
end for
k ← 1
while True **do**
  Q ← []
  for i=Fronts{k} **do**
    P1 ← Pop[i]
    for j=P1.DominationSet **do**
      P2 ← Pop[j]; P2.DominatedCount ← P2.DominatedCount-1
      if P2.DominatedCount==0 **then**
        Q.append(j); P2.rank ← k+1
      end if
      Pop[j] ← P2
    end for
  end for
  if isempty(Q) **then**
    break
  end if
  Fronts{k + 1} ← Q; k ← k + 1
end while
Return Pop, Fronts

Algorithm 4 Crowding distance

Input: Pop, Fronts
Output: Pop

front_layers ← numel(Fronts)
for layers in front_layers **do**
  h ← |Fronts{layers}|
  for i=1 to h **do**
    $CD_i \leftarrow 0$
  end for
  nObj=len(Fronts{layers}.Fit)
  for j=1 to nObj **do**
    sort (Fronts{layers},f_j)
    $CD_1 \leftarrow$ inf; $CD_h \leftarrow$ inf
    for i=2 to h-1 **do**
      $CD_i \leftarrow CD_i + d_i^j$ based on Eqs. (16) and (17)
    end for
  end for
  Update the CD values of corresponding solutions in Pop with related CDs
end for
Return Pop

Algorithm 5 Sort

Input: Pop
Output: Pop, ParetoFront, maxAcc

ParetoFront ← []
CDSO ← Sort order the population based on the crowding distance descendingly
Pop ← Pop[CDSO]
RSO ← Sort order the population based on the ranks
Pop ← Pop[RSO]
ParetoFront ← append all solutions with rank of 1
maxAcc ← Save the maximum prediction accuracy ($\hat{y}$) in Pareto front
Return Pop, ParetoFront, maxAcc

sorting and distances measured by crowding distance. Subsequently, in the NSGA-II with adaptive BFM, the crossover population (Pop_c) and mutation population (Pop_m) are iteratively generated and added to the initial population (Pop) until a stopping criterion is met. To maintain the quality of the population, Pop is sorted according to ranks and crowding distances, ensuring that only the top $nPop$ solutions are retained.

The crossover operation in this study follows the existing Single-Point Crossover (SPC) procedure: Let H represents the length of the solution vectors x_1 and x_2. A random index C is generated between 1 and $H-1$. Two parents are selected using Binary Tournament Selection (BTS) for the crossover operation. BTS randomly chooses two individuals from the population and selects the one with a lower rank; if ranks are equal, it chooses the individual with a greater crowding distance.

Two offspring solutions, z_1 and z_2, are created as described in Eq. (18), where z_1 is obtained by concatenating the first C elements from x_1 with the remaining elements from x_2, and z_2 is formed by concatenating the first C elements from x_2 with the remaining elements from x_1. The resulting offspring solutions z_1 and z_2 are then returned for the next generation.

The SPC in NSGA-II repeatedly generates new offspring until the size of crossover population reaches $nPop_c$ (refer to Table 3). Eq. (18) demonstrates SPC, with $\oplus$ denoting the concatenation sign.

$$z_i = \begin{cases} [x_1[1:C] \oplus x_2[C+1:H]], & \text{if } i = 1 \\ [x_2[1:C] \oplus x_1[C+1:H]], & \text{if } i = 2 \end{cases} \tag{18}$$

The mutation operator in the proposed NSGA-II avoids premature convergence using densify and sparsify operators along with Bit Flip Mutation (BFM) as described in Algorithm 6 inside the main body of MOGAE. Algorithm 6 calls the corresponding functions of BFM, densify, and sparsify in Eqs. (19)–(21) to generate Pop_m. The expression $\frac{y}{\hat{y}}$ in Algorithm 6 signifies the ratio between the prediction accuracy of the *original image* (represented as y) and the highest prediction accuracy within the Pareto front (represented as $\hat{y}$). We refer to $\Theta_i = \frac{y}{\hat{y}}$ as the deviation in the present iteration. This ratio is used to compare the prediction accuracy of the *original image* (y) relative to the best accuracy achieved within the Pareto front ($\hat{y}$). A larger value of the fraction ($\Theta_i > 3$) indicates a larger deviation between y and $\hat{y}$ while a smaller value of the fraction ($\Theta_i \leq 3$) indicates that y is closer to $\hat{y}$.

While $\Theta_i > 3$, the Algorithm 6 is allowed to escape premature convergence within 10 consecutive iterations (controlled by TagCheck) using the BFM operator, as shown in Eq. (19) where X is a randomly selected solution in Pop (for each invocation of the corresponding function) with the length of H. Similarly, when $\Theta_i \leq 3$, Algorithm 6 attempts to escape premature convergence within 10 several consecutive iterations (controlled by TagFlag) using the BFM operator, as defined in Eq. (19). In this process, J represents a set of randomly selected

Algorithm 6 Adaptive BFP with densifying and sparsing operators

1: $\Theta_i \leftarrow \frac{y}{\hat{y}}$
2: **if** $\Theta_i > 3$ and TagCheck = 10 **then**
3: $\Omega \leftarrow \Omega + \delta \times (1 - \Omega)$
4: **for** k=1 to nPopm **do**
5: Popm[k] ← Densify(X, Ω)
6: **end for**
7: TagCheck ← 0
8: **else if** $\Theta_i > 3$ and TagCheck < 10 **then**
9: **for** k=1 to nPopm **do**
10: Popm[k] ← BFM(X)
11: **end for**
12: **else if** $\Theta_i \leq 3$ and TagFlag = 10 **then**
13: $\Omega \leftarrow \Omega - (\delta \times \Omega)$
14: **for** k=1 to nPopm **do**
15: Popm[k] ← Sparsify (X, Ω)
16: **end for**
17: TagFlag ← 0
18: **else if** $\Theta_i \leq 3$ and TagFlag < 10 **then**
19: **for** k=1 to nPopm **do**
20: Popm[k] ← BFM(X)
21: **end for**
22: **end if**

indices indicating the positions of the bits to be flipped. The size of set J is 10% of the length of H.

$$BFM(X) = \begin{cases} 1 - x_i, & if \quad i \in J;\ 1 \leq i \leq H \\ x_i, & otherwise \end{cases} \tag{19}$$

However, suppose the BFM operator cannot escape premature convergence after 10 consecutive iterations while $\Theta_i > 3$. In that case, we increase the granularity of active superpixels by raising the chance of generating active superpixels through updating Ω at line 3 and using the $densify$ operator at line 5 (TagCheck was initially set to 0, and δ = 0.2). The $densify$ operator in Eq. (20) does not alter bits with values of 1. Instead, it modifies 0 bits to 1 with a probability determined by $r \leq \Omega$, where r is a random variable in the range [0, 1].

$$Densify(X, \Omega) = \begin{cases} 1 - x_i, & if \quad i \in \{i \mid x_i = 0\} \quad \&\& \quad r \leq \Omega \\ x_i, & otherwise \end{cases} \tag{20}$$

Similarly, suppose the BFM operator cannot escape premature convergence after 10 consecutive iterations while $\Theta_i \leq 3$. In that case, we decrease the granularity of active superpixels by raising the chance of generating non-active superpixels through updating Ω at line 13 and using the $sparsify$ operator at line 15 (TagFlag was initially set to 0, and δ = 0.2). The $sparsify$ operator in Eq. (21) does not alter bits with values of 0. Instead, it modifies 1 bit to 0 with a probability determined by $r \leq \Omega$, where r is a random variable in the range [0, 1].

$$Sparsify(X, \Omega) = \begin{cases} 1 - x_i, & if \quad i \in \{i \mid x_i = 1\} \quad \&\& \quad r \leq \Omega \\ x_i, & otherwise \end{cases} \tag{21}$$

After each iteration, TagCheck and TagFlag are updated according to the current deviation in the present iteration (Θ_i) and the previous deviation in the preceding iteration (Θ_{i-1}), as outlined in Algorithm 7.

Fig. 4 illustrates the impact of increasing or decreasing δ on Ω across 20 consecutive iterations in both the $densify$ and $sparsify$ operators as described in Algorithm 6. It is evident that by increasing δ from 0.2 to 0.4 in the $densify$ operator, the curve widens outward. Similarly, this occurs when we double the value of δ from 0.2 to 0.4 in the $sparsify$ operator. In other words, δ alters the gradient of the original curve through the Ω formulation. We experimentally concluded that δ = 0.2 can produce satisfactory results in our data. Additionally, it is crucial to note that the range of Ω in both $densify$ and $sparsify$ operators lies within [0, 1]. This characteristic is essential according to Eqs. (20)–(21).

Algorithm 7 Updating TagCheck and TagFlag at the end of each iteration

1: **if** $\Theta_i > 3$ **then**
2: **if** $\Theta_i = \Theta_{i-1}$ **then**
3: TagCheck ← TagCheck + 1;
4: **else**
5: TagCheck ← 1;
6: **end if**
7: **else**
8: **if** $\Theta_i = \Theta_{i-1}$ **then**
9: TagFlag ← TagFlag + 1;
10: **else**
11: TagFlag ← 1;
12: **end if**
13: **end if**

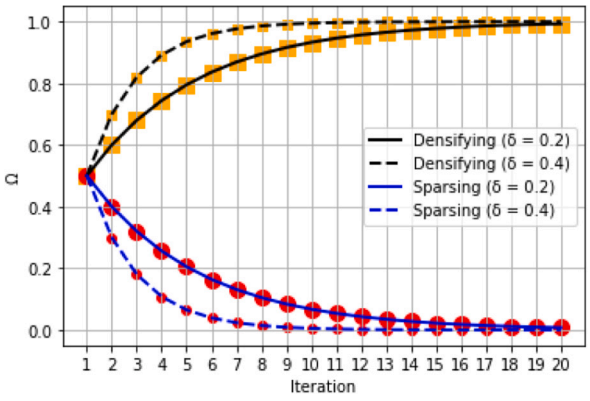


Fig. 4. The impact of δ on Ω.

Table 3
Parameters of NSGA-II.

NSGA-II parameters	Value
Population size (nPop)	35
Probability of crossover (P_c)	0.9
Probability of mutation (P_m)	0.4
Crossover population size ($nPop_c$)	$nPop_c = 2 \times \lceil P_c \times \left(\frac{nPop}{2}\right) \rceil = 32$
Mutation population size ($nPop_m$)	$nPop_m = \lceil P_m \times nPop \rceil = 14$
Crossover operator	Single-Point Crossover (SPC)
Mutation operator	Adaptive Bit Flip Mutation (BFM) with densify and sparsify operators
Selection	Non-dominated sorting and crowding distance
Maximum number of iterations	100
Stopping criterion	Reaching the maximum number of iterations

After NSGA-II, MOGAE employs majority voting on the images within the Pareto front that accurately predict the actual class of the *original image* in the iteration with minimal deviation (*optimal images* in Fig. 3) to generate the corresponding *MV image* in Fig. 3. Usually, the *MV image* effectively highlights the informative areas of the *original image*. However, to more effectively eliminate potential non-informative superpixels from the *MV image*, we also compute the intersection of the *base image* and the *MV image*, resulting in the final *explanatory image* in Fig. 3. In summary, the NSGA-II parameters for reusability and reproducibility of the results are given in Table 3.

5. Results

Before evaluating the results, we introduce the evaluation criteria for both the prediction pre-trained ResNet50 model and the MOGAE explainer in Section 5.1. We also introduce the experimental platform, including the software and hardware used in Section 5.2. Next, we evaluate the prediction model in Section 5.3 and assess the MOGAE explainer in Section 5.4.

5.1. Evaluation criteria

Briefly, the proposed training model in Fig. 2 is evaluated using accuracy, precision, recall, and F1-score in Eqs. (22)–(26) as well as the model loss and model accuracy plots. Accuracy is generally defined as the ratio of correct predictions to the total number of predictions. This is reflected in Eq. (22) and Eq. (23) for binary and multiclass classifications, respectively. In Eq. (23), y_i refers to the real class label, and $h(x_i)$ refers to the predicted class label, both for the i-th sample in the test set (TS). In the following, TP, TN, FP, and FN stand for True Positive, True Negative, False Positive, and False Negative, respectively. In the case of multiclass classification, the average strategy is used to calculate precision, recall, and F1-score.

$$Accuracy_{binary} = \frac{TP + TN}{TP + FP + FN + TN} \tag{22}$$

$$Accuracy_{multiclass} = \frac{\sum_{x_i \in TS} 1(y_i = h(x_i))}{|TS|} \tag{23}$$

$$Precision = \frac{TP}{TP + FP} \tag{24}$$

$$Recall = \frac{TP}{TP + FN} \tag{25}$$

$$F1 - score = 2 \times (\frac{Precision \times Recall}{Precision + Recall}) \tag{26}$$

In evaluating the MOGAE explainer and comparing it with EGAE and LIME, we focused on two primary criteria: numeric accuracy of explanation, and Number of Function Evaluations (NFE). Although accuracy can be intuitively investigated, we also assess accuracy through explanation error, calculated by measuring the Euclidean distance between the *actual_explanation* (provided by experts) and the *calculated_explanation* (computed by the explainer) in Eq. (27) (Nematzadeh et al., 2023).

$$error = \text{Euclidean dist}(actual\_explanation - calculated\_explanation) \tag{27}$$

The *normalized_error* is calculated using Eq. (28) to standardize the error values across all images and transform them into a range of [0,1]. The best-case scenario (denoted as *min_error* in Eq. (28)) occurs when the *calculated_explanation* perfectly matches the *actual_explanation*, resulting in zero error. In contrast, the worst-case scenario (denoted as *max_error* in Eq. (28)) occurs when the *calculated_explanation* has no overlapping pixels with the *actual_explanation*, leading to a positive error value that varies from image to image. A lower value of the *normalized_error* in Eq. (28) indicates a more accurate explanation. To enhance understanding, the delineations made by agricultural experts (referred to as *actual_explanation* for the citrus leaf disease dataset) are illustrated using the SLIC segmentation algorithm, as depicted in Fig. 6.

$$normalized\_error = \frac{error - min\_error}{max\_error - min\_error} = \frac{error}{max\_error} \tag{28}$$

NFE refers to the count of times the fitness function in an evolutionary algorithm (NSGA-II in this paper) is called. This count explicitly represents the number of images used by NSGA-II for explanation. NFE serves as a fair and reliable metric, more trustworthy than CPU time, especially when comparing different algorithms implemented in various ways. In other words, the number of images an explainer employs for explanation is a fairer measurement criterion compared to the time spent on execution. The general formula for NFE in evolutionary algorithms is presented in Eq. (29) (Nematzadeh et al., 2023), where I represents the number of solutions in the initial population, and O represents the total count of offspring generated in each iteration. Eq. (30) is a specific instance of Eq. (29) utilized in this paper. However, MOGAE's NFE has an additional burden of 2^h due to the systematic search. Given that h is very small (recall that $h \ll H$), disregarding 2^h in MOGAE is not crucial. It is important to mention that the number of images required by LIME for evaluation is manually allocated by the expert before execution.

$$NFE = I + [O \times number\ of\ iterations] \tag{29}$$

$$NFE = Pop + [(Pop_c + Pop_m) \times number\ of\ iterations] \tag{30}$$

This paper also benefits from statistical analysis using the Friedman test and the Nemenyi post-hoc test.

5.2. Experimental setup

To ensure a fair comparison among LIME, EGAE,[1] and MOGAE, the SLIC segmentation algorithm is utilized by all three explainers. In LIME, the random state is set to *None* because we avoid artificially ensuring result reproducibility by fixing the random number generation process. Subsequently, the three explainers are evaluated based on the intuitive and numeric accuracy of the explanation and the NFE. Furthermore, a discussion on the level of dependency on hyperparameters is presented to provide sufficient evidence for explainer selection. For the evaluation of our customized ResNet50, we use average = 'macro' for multiclass classification. This means that each metric in Eqs. (24)–(26) is calculated individually for each class, and the average of these metrics is then computed.

All the experiments have been conducted in a virtualization environment on a private high-performance cluster computing platform. This infrastructure is located at the Ada Byron Research Center at the University of Málaga (Spain) and comprises several IBM hosting racks for storage, virtualization units, server compounds and backup services. Our virtualization platform is hosted in this computational environment. Concretely, this platform is made up of a CPU with Intel(R) Xeon(R) Gold 6130 @ 2.10 GHz, a maximum of 2 TB of HDD, a maximum of 64 GB of RAM, and Ubuntu 20.04.3 LTS (GNU/Linux 5.4.0-1049-kvm x86 64). All the implementations for the deep learning training model and simulations of explanations are coded and executed in the Python 3.9 software environment.

5.3. Model evaluation

The training ResNet50 model proposed in Fig. 2 achieves the test accuracy of 0.95 in Table 4, which is sufficiently good and can be relied on to explain the predictions. Furthermore, the reported precision, recall, and F1-score confirm the robustness of the proposed training model. The confusion matrix in Table 5 also demonstrates the prediction accuracy of each class. This level of accuracy is achieved on 59 test samples. The model loss and accuracy in Fig. 5 clearly confirm that the training model is neither overfitted nor underfitted. The validation loss in model loss improves from infinity to 4.98 and gradually decreases to almost 0.1. Fig. 5 clearly shows that the validation loss and training loss both decrease and stabilize at a specific point with a minimal gap between them. The same happens for model accuracy, where both training accuracy and validation accuracy increase and stabilize at a specific point with very little gap between them.

[1] https://github.com/KhaosResearch/EGAE.

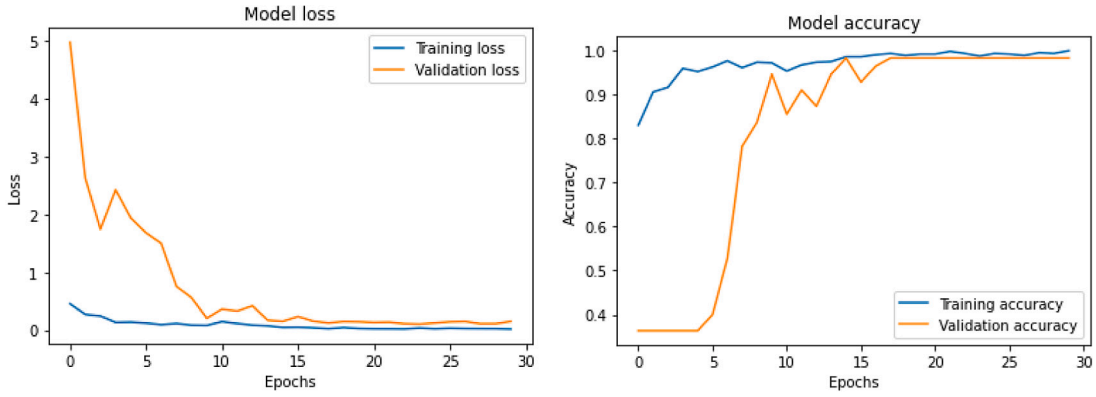


Fig. 5. Model loss and model accuracy evaluation.

Table 4
Accuracy metrics achieved by the proposed model in Fig. 2.

Accuracy train	Accuracy validation	Accuracy test	Precision	Recall	F1-score
0.99	0.97	0.95	0.96	0.96	0.96

Table 5
Confusion matrix achieved by the proposed model in Fig. 2.

Classes		Actual labeling of classes				
		Black spot	Canker	Greening	Healthy	Total
Predicted labeling of classes	Black spot	16	0	1	0	17
	Canker	0	16	0	0	16
	Greening	2	0	18	0	20
	Healthy	0	0	0	6	6

5.4. Model explanation

Four images with a high level of prediction have been chosen from the existing four classes of citrus leaf disease categories to explain the proposed model in Fig. 2. These images are depicted in the first row of Fig. 6. Furthermore, the actual delineations provided by agricultural experts and the corresponding delineations by SLIC are presented in the second and third rows of Fig. 6, respectively. Unlike the disease classes, the healthy class exhibits no symptoms. Consequently, the agricultural experts could not specify a specific healthy area in the image.

In Fig. 7, the explanations provided by MOGAE for the images in Fig. 6 are displayed. As MOGAE employs NSGA-II, it is likely not to converge to a single unique solution in every run. Hence, in Fig. 7, each row represents the MOGAE results obtained from three consecutive runs. Fig. 7 serves as compelling evidence that MOGAE accurately identifies the informative areas in its explanations, even when minor non-informative areas exist. Segmenting the *original image* into 50 superpixels opens up the possibility of generating many global optima or multiple local optima within the search space. Absolutely, the inherent complexity due to the segmentation of the image into 50 superpixels can indeed challenge NSGA-II's convergence, resulting in different solutions across runs. Undoubtedly, the consistency in highlighting some or all parts of the informative leaf area across these varied solutions is attributed to the effectiveness of adaptive Bit Flip Mutation (BFM) with densify and sparsify operators employed within the process.

In the second row of Fig. 7, MOGAE demonstrates its capability to explain the leaf affected by canker disease. In Fig. 6, agricultural experts marked two distinct areas related to the canker sample causing the infection. The initial attempt at explaining the canker sample in Fig. 7 converged to one of those two marked areas. However, the second and third attempts converged to the other area, albeit with some portions including non-infected parts of the image. This also holds for explaining the black spot in the first row of Fig. 7. In the third row of Fig. 7, MOGAE accurately pinpoints small areas of the image for an explanation regarding the greening infection, aligning with the delineation provided by experts in Fig. 6. Regarding the healthy image, MOGAE seems to provide explanations using the entire leaf, with some focus on the marginal areas.

Table 6
Normalized error of explanation.

Leaf sample	LIME-I (Ribeiro et al., 2016)	LIME-II (Ribeiro et al., 2016)	EGAE with majority voting (Nematzadeh et al., 2023)	MOGAE
Black spot	0.69	0.64	**0.05**	0.12
Canker	0.87	0.89	**0.23**	0.25
Greening	0.83	0.82	0.35	**0.13**
Healthy	0.95	0.95	0.53	**0.52**

For a more intuitive comparison, EGAE with majority voting and the existing LIME library are used for explanations in Fig. 8. Both EGAE and MOGAE implicitly determine the number of images they use for explanation through NFE in Eq. (29). However, in LIME, the number of samples (num_samples) needs to be predetermined by the expert, which can be considered one of its limitations. It is important to note that a higher num_samples results in more accurate and stable explanations but also increases execution time. To ensure a fair comparison, we assigned the same NFEs of MOGAE to the corresponding num_samples parameter in LIME.

Comparing explanations provided by EGAE and MOGAE, we can intuitively see that EGAE seems to provide a better explanation for the black spot. At the same time, MOGAE appears to be more accurate in explaining the greening sample. Additionally, EGAE and MOGAE seem to behave almost identically in explaining canker and healthy samples.

However, a compelling argument can be made for the superior performance of both EGAE and MOGAE over LIME. LIME exhibits a tendency to identify most margins as areas that contribute positively to the prediction in canker and healthy samples. Notably, in the case of greening, LIME displays a critical shortcoming by marking certain areas, previously delineated as important by agricultural domain experts, in red, indicating a negative contribution to the prediction.

Table 6 presents the normalized error of explanation, calculated using Eq. (28), based on the actual explanations in Fig. 6 compared to the explanations derived for MOGAE in Fig. 7 alongside the corresponding explanations of EGAE and LIME in Fig. 8. The results in Table 6 represent the average values obtained from three consecutive runs.

EGAE and MOGAE employ distinct techniques and specify the number of images used during the explanation process. The average Number of Function Evaluations (NFE) of EGAE from three runs were assigned to LIME in the LIME-I experiment to ensure a fair comparison. Similarly, the NFE of MOGAE was allocated to LIME in the LIME-II

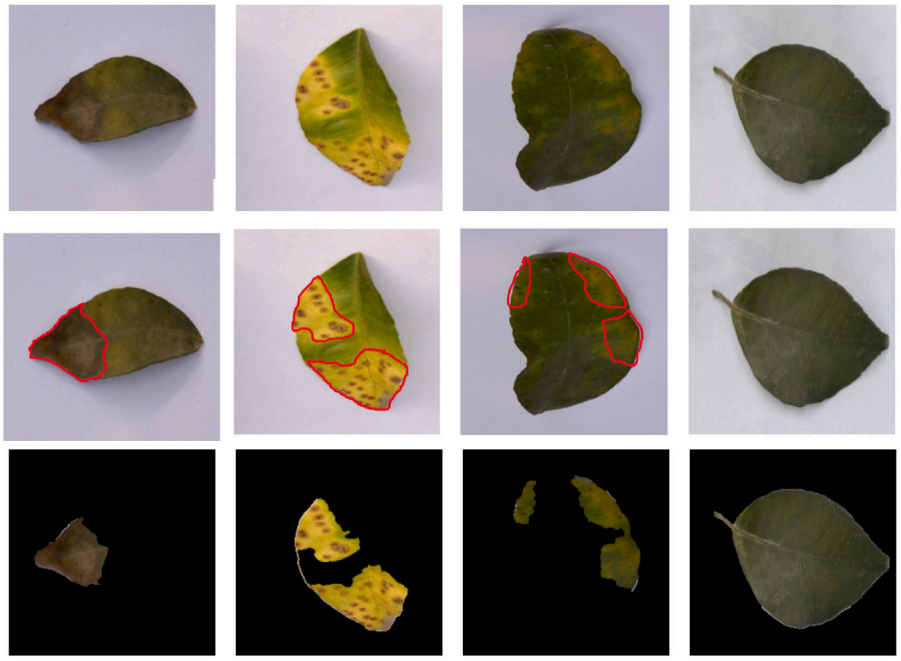


Fig. 6. Row 1 exhibits selected sample test images for explanation, arranged from left to right: Black spot, Canker, Greening, and Healthy. Row 2 displays delineations outlined in red lines provided by agricultural experts, while Row 3 showcases the corresponding delineations generated by SLIC.

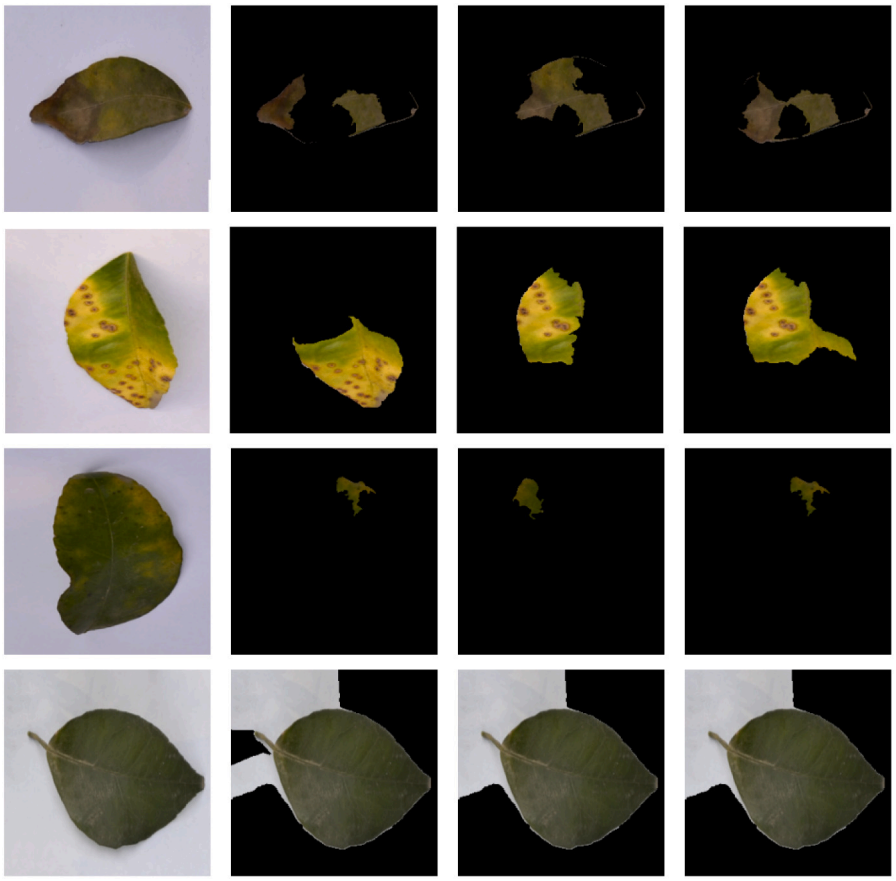


Fig. 7. Explanation of MOGAE when the input images are segmented into 50 superpixels (H = 50). In each row, from left to right, you can observe the original image followed by three attempts at explanation generated by MOGAE.

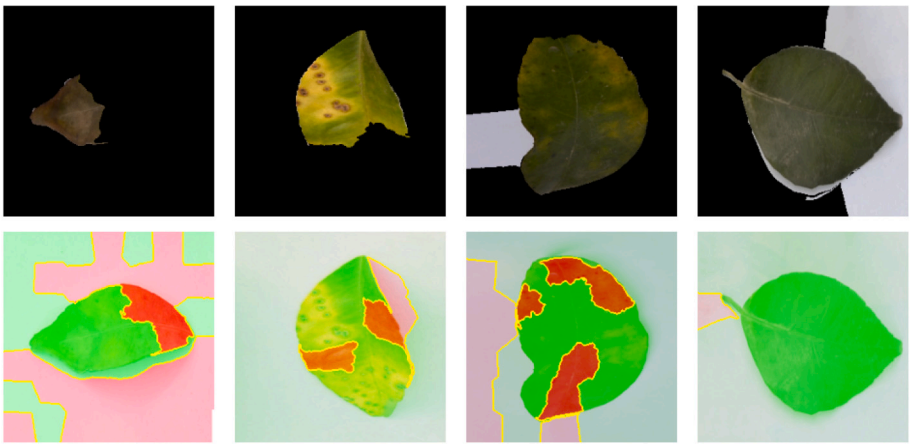


Fig. 8. Explanation of EGAE with majority voting and LIME in first and second rows, respectively. One attempt of each explainer is depicted and the input images are segmented into 50 superpixels (H = 50).

experiment. Notably, Table 6 indicates a marginal difference between LIME-I and LIME-II, displaying slight improvement for black spot and greening samples with LIME-II compared to LIME-I.

The substantial explanation error observed for canker and healthy samples stems from LIME-I and LIME-II, identifying most margins as significant areas. While it is evident that both EGAE and MOGAE outperform LIME, the numerical analysis in Table 6 does not decisively indicate superiority between EGAE and MOGAE. Interestingly, both EGAE and MOGAE exhibit nearly identical behavior in explaining canker and healthy samples. However, MOGAE demonstrates considerable superiority over EGAE in greening, whereas EGAE excels in explaining the black spot. Consequently, while EGAE and MOGAE might offer similar explanations in certain scenarios, there are instances where one may outperform the other.

In order to provide statistical validation of the observed performance differences, we conducted the **Friedman test** to compare the performance of the different methods (LIME-I, LIME-II, EGAE, and MOGAE) across the various diseases (Black spot, Canker, Greening, and Healthy). This test is appropriate because we are comparing multiple methods (LIME-I, LIME-II, EGAE, and MOGAE), and the data represent repeated measures on the same samples (different diseases). The Friedman test is a non-parametric test suitable for paired data and more than two groups. The results of the Friedman test are *Friedman statistic*: 10.42 with *p-value*: 0.0153. As the *p-value* is below 0.05, we can conclude that there is a statistically significant difference between the methods (LIME-I, LIME-II, EGAE, and MOGAE). To further investigate these differences, we performed the **Nemenyi post-hoc test**, revealing significant differences for LIME-I vs. MOGAE (*p-value*: 0.031) and LIME-II vs. MOGAE (*p-value*: 0.042), and EGAE vs. MOGAE with (*p-value*: 0.053), close to the threshold of 0.05. Hence, we contend that both EGAE and MOGAE demonstrate complete superiority over LIME and are strongly recommended for use. Fig. 9 compares the average Number of Function Evaluations (NFE) for EGAE and MOGAE across three runs. MOGAE remains unaffected by the segmentation algorithm's output, consistently utilizing the same number of images regardless of the number of segments generated. Conversely, EGAE's performance is contingent upon the segmentation algorithm's output. Therefore, in Fig. 9, EGAE was executed under two settings: EGAE-I with 50 superpixels and EGAE-II with 100 superpixels.

Fig. 9 notably illustrates that EGAE-I (light blue bars) employs fewer images compared to MOGAE (red bars) when utilizing 50 segments. Conversely, with an increase in input segments to 100 in EGAE-II (dark blue bars), a corresponding rise in NFE is evident compared to EGAE-I.

For example, EGAE-II utilizes a significantly greater number of images than MOGAE during the optimization process in the canker sample. However, in the case of greening, although the NFE of EGAE-II has

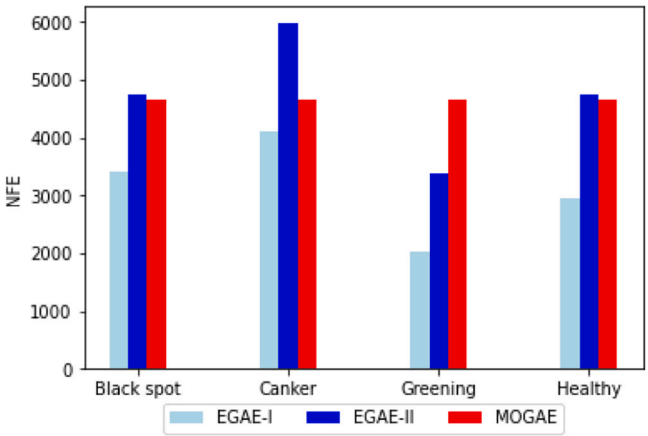


Fig. 9. NFE comparison of EGAE and MOGAE.

increased in comparison to EGAE-I, it remains lower than that of MOGAE. Nonetheless, this reduction in NFE for EGAE-II does not represent an advantage in the greening sample, as the normalized explanation error of MOGAE remains notably lower than that of EGAE-II.

MOGAE and EGAE-II generally show similar execution times when the number of superpixels (or input segments, H) is set to 100, with both methods requiring a comparable duration to generate explanations. However, when the number of superpixels is reduced, such as at H = 50, EGAE-I demonstrates a faster execution time compared to MOGAE. The sample execution times for canker in a single run with LIME (with the num_samples set to match the NFE of MOGAE, which is 4651), MOGAE (NFE = 4651), and EGAE-II (NFE = 6214), while H = 100, are 120 s, 1025 s, and 1194 s, respectively, while the execution time for EGAE-I (NFE = 3950) in a single run with H = 50 is 777 s. Overall, the experiments generally indicate a direct relationship between the number of superpixels and both NFE and execution time in EGAE; increasing the number of superpixels generally tends to raise NFE and, consequently, execution time. In contrast, for MOGAE, both execution time and NFE remain constant, irrespective of the number of superpixels.

It should be noted that the computational complexity of MOGAE (in its metaheuristic search) is generally similar to that of NSGA-II, as MOGAE builds upon the NSGA-II algorithm. NSGA-II has a time complexity of $O(MN^2)$, where N represents the population size and M is the number of objectives.

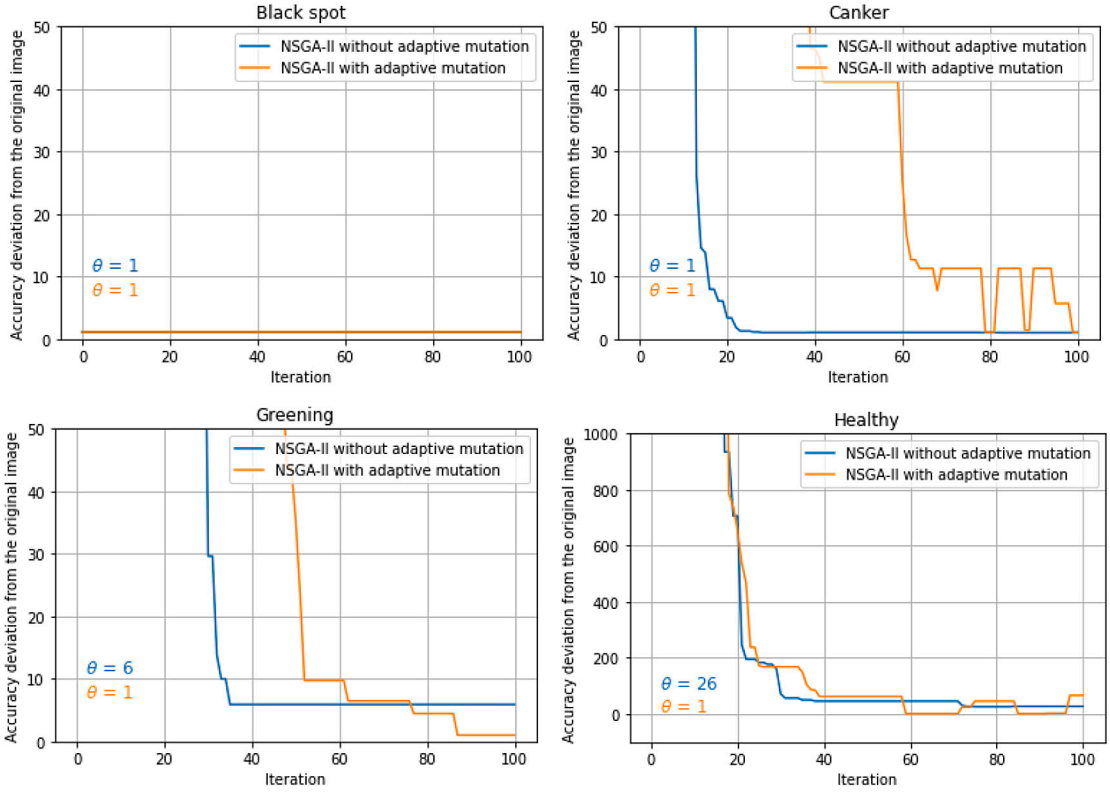


Fig. 10. Accuracy deviation (θ) between the best solution in the Pareto front and the original image across consecutive iterations in NSGA-II for a scenario where the input images are segmented into 100 superpixels (H = 100).

Generally, in both EGAE and MOGAE, the normalized explanation error tends to remain almost consistent, irrespective of whether they operate with 50 or 100 superpixels.

In Fig. 10, the variations in Θ, as mentioned in Algorithm 6, can be observed across iterations within a single run of NSGA-II (the meta-heuristic section of MOGAE). Accordingly, the value of Θ_i represents the deviation between the accuracy of the best solution within the Pareto front and the accuracy of the *original image* in the present iteration. Fig. 10 shows that Θ undergoes a gradual decrease across iterations. This decrease is particularly sharp in the cases of canker, greening, and healthy at the start, gradually smoothing out through subsequent iterations. However, NSGA-II appears to have discovered highly effective solutions for black spot from the beginning.

Moreover, Fig. 10 implies that the *optimal images* generated by the proposed NSGA-II incorporating adaptive Bit Flip Mutation (BFM) with densify and sparsify operators exhibit minimal deviation in accuracy from the *original image*. In this context, Fig. 10 illustrates the values of Θ in NSGA-II with adaptive mutation (orange plot) and without adaptive mutation (blue plot). The optimal value of Θ is 1, indicating no deviation between the prediction accuracy of best solution in the Pareto front and the *original image*. For greening and healthy images, when adaptive mutation is not employed, the quality of the best solution in the Pareto front is significantly diminished. The values of Θ are 6 and 26 in NSGA-II without adaptive mutation, signifying that the prediction accuracy of the best solution in the Pareto front is 6 times and 26 times worse than the prediction accuracy of the corresponding *original image*, respectively. Consequently, adaptive mutation almost guarantees the convergence of NSGA-II in MOGAE.

In Fig. 11, the number of members within the Pareto front is depicted across iterations in NSGA-II with $H = 50$. Initially, the Pareto front starts with fewer members, which aligns with expectations. However, over multiple consecutive iterations, the count of Pareto front members gradually rises until all 35 population members contribute to the Pareto front. If NSGA-II discovers new sets of optimal solutions during optimization, sudden decreases in the Pareto front's population might occur. However, this is followed by a gradual increase again. It is important to note that these 35 solutions within the Pareto front might not necessarily be unique. In summary, the insights gained from the plots in Fig. 11 affirm the dynamic nature of NSGA-II's exploitation and exploration, providing a comprehensive view of the optimization process.

As a concluding note, both EGAE and MOGAE effectively alleviate the need for some of the hyperparameter tunings that are a prerequisite in LIME. LIME requires the user to specify the number of superpixels upfront. EGAE addresses this issue by employing multiple consecutive single-objective genetic algorithms, each with a predefined number of superpixels. Thus, it automatically generates explanations by utilizing the outcome of each genetic algorithm through voting strategies. Although MOGAE does not incorporate multiple settings for the number of superpixels, it employs automatic granularity adjustments for active superpixels to strive for the most detailed explanation possible within the predefined number of superpixels and majority voting of valid images within the Pareto front. Valid images are solutions located on the Pareto front where "ValidAcc" is equal to 1.

In addition, the num_samples and num_features parameters are two other parameters that should be tuned in LIME. The num_samples parameter is used to specify the number of perturbations to apply to the input image when generating an explanation. A higher value for num_samples will result in a more detailed explanation, as LIME will consider more permutations of the image pixels. However, it will also take longer to generate the explanation. Both EGAE and MOGAE implicitly calculate the appropriate number of perturbations during optimization through the value of NFE, which is a considerable advantage.

The num_features parameter in LIME for image explanation determines the maximum number of features to consider when generating an explanation. A higher value of num_features allows LIME to examine

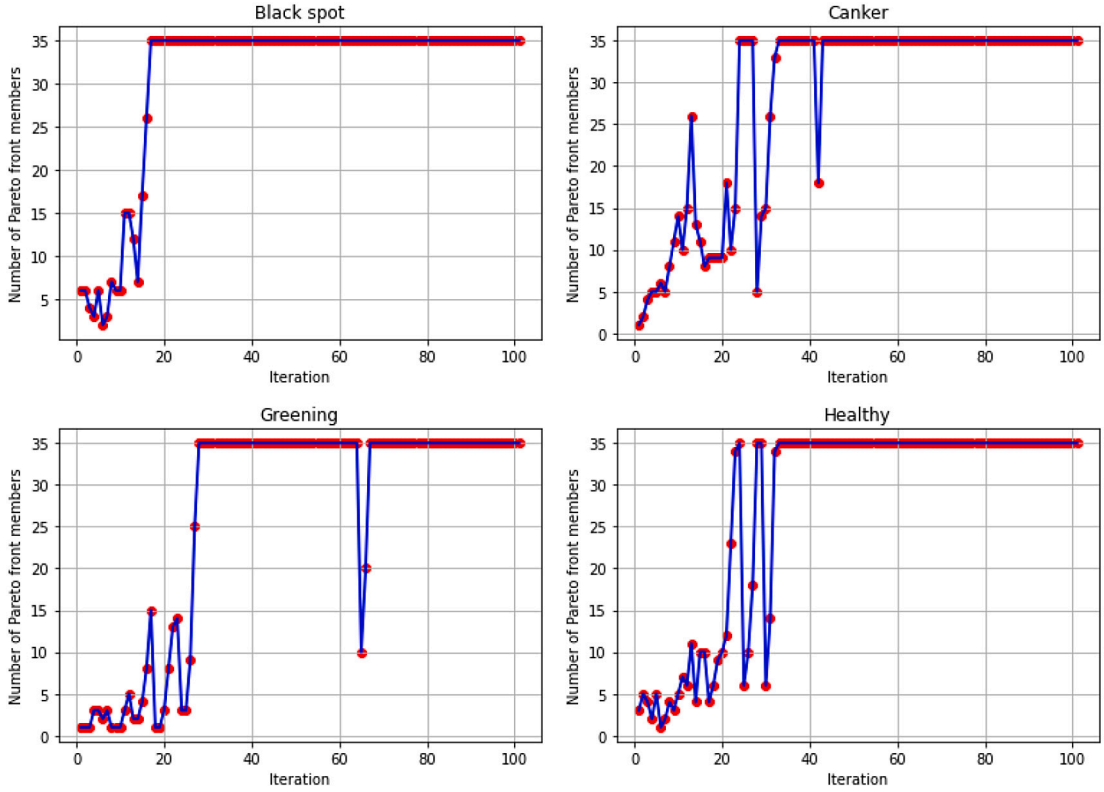


Fig. 11. Size of the Pareto front population in NSGA-II when the input images are segmented into 50 superpixels (H = 50) and nPop = 35.

more of the image pixels, potentially leading to a more detailed explanation. However, it has been observed practically that LIME tends to include margins of the image as factors that positively contribute to the prediction. In some instances, LIME may even overlook the critical regions of the image while identifying features that positively impact the model's prediction when num_features is set to the maximum value. In contrast, Both EGAE and MOGAE automatically determine the final explanation for the user without requiring the specification of the maximum number of features to consider for explanation generation.

6. Conclusion

This paper presents a model-agnostic Multi-Objective Genetic Algorithm Explainer (MOGAE) to explain the models' predictions on individual test samples. MOGAE consists of two main components: systematic search and metaheuristic search. While most of the explanation process is conducted within the metaheuristic search using NSGA-II, the systematic search plays a positive role in identifying and removing potentially non-informative areas in certain scenarios. Furthermore, the proposed NSGA-II employs adaptive Bit Flip Mutation (BFM) with densify and sparsify operators, allowing NSGA-II to dynamically adjust the granularity of active superpixels to escape local optima and premature convergence. As a consequence, a predictive model for citrus leaf diseases dataset is deveoped that utilizes a customized pre-trained ResNet50 architecture. The existing dataset undergoes preprocessing, oversampling, and augmentation. The robustness of the proposed method is assessed through metrics such as model loss, model accuracy, and performance criteria, including accuracy, precision, recall, and F1-score.

Intuitive and numerical evaluations (normalized explanation error and NFE) demonstrate that MOGAE outperforms LIME significantly. This is attributed to MOGAE's ability to discard more non-informative and trivial areas of the image. In a brief comparison between Ensemble-based Genetic Algorithm Explainer (EGAE), MOGAE, and LIME, the following facts emerge:

- We cannot clearly distinguish between EGAE and MOGAE in terms of explanation accuracy. In some images, MOGAE is more accurate, while in others, EGAE performs better. However, based on both intuitive and numeric investigations, both EGAE and MOGAE are more accurate than LIME. Therefore, we recommend using both methods in different scenarios.
- Regarding NFE, EGAE requires fewer images than MOGAE in scenarios with fewer superpixels. In contrast, for a great number of superpixels (100, for instance), EGAE may utilize a higher number of images, even surpassing MOGAE in some cases. Therefore, it can be concluded that MOGAE is generally more efficient than EGAE in scenarios with a greater number of superpixels. In contrast to EGAE and MOGAE, LIME does not have the ability to implicitly determine the number of images for evaluation.
- The number of perturbed images (num_samples) and the maximum number of features to consider when generating an explanation (num_features) in LIME must be determined by the expert prior to explanation. Furthermore, the number of superpixels of the *original image* in Fig. 3 should be determined by the expert using the corresponding segmentation algorithm in advance. However, there is no clear guidance on how to choose these parameters, and they may vary depending on the dataset and the model. Therefore, this can be seen as a limitation of LIME, as it requires expert knowledge and trial-and-error to find the optimal values. However, both EGAE and MOGAE do not need these parameter tuning.

- Additionally, the experimental results reveal that although MOGAE generally identifies essential areas for explanation, it shares the non-reproducible nature with EGAE and LIME. Likewise, MOGAE, akin to EGAE, might produce unexpected outcomes in a minority of cases, primarily attributed to the inherent nature of evolutionary algorithms.

The segmentation algorithm used by the explainer can directly affect the quality and accuracy of the explanations, because it determines how the image is divided into superpixels and how the superpixels are perturbed. Different segmentation algorithms can yield distinct superpixel characteristics, encompassing diverse numbers, sizes, shapes, and locations, capturing various facets of image features. Additionally, these algorithms may entail disparate computational costs and complexities, impacting the speed and efficiency of the explainer. Consequently, future research directions could concentrate on improving the performance of image segmentation algorithms in this paper. Additional future directions for this work include exploring the ability of other evolutionary algorithms to generate explanatory images, as well as evaluating MOGAE's performance in other domains.

CRediT authorship contribution statement

Hossein Nematzadeh: Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Methodology, Investigation, Formal analysis, Conceptualization. **José García-Nieto:** Writing – review & editing, Validation, Supervision, Conceptualization. **Sandro Hurtado:** Writing – review & editing, Visualization, Validation. **José F. Aldana-Montes:** Supervision, Project administration, Funding acquisition. **Ismael Navas-Delgado:** Writing – review & editing, Supervision, Funding acquisition.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work has been partially funded by grants PID2020-112540RB-C41, AETHER-UMA (A smart data holistic approach for context-aware data analytics: semantics and context exploitation) and QUAL21 010UMA (Junta de Andalucía). It is also funded for open access charge: Universidad de Málaga/CBUA.

Appendix

We also investigated the results of MOGAE on the melanoma dataset using the corresponding fine-tuned ResNet50 model. We showcased three test samples from the classes of melanoma, seborrheic-keratosis, and nevus, providing their respective numerical and illustrative evaluations. Readers are encouraged to refer to the comprehensive Appendix file[2] for additional images and evaluations. The comprehensive Appendix file includes not only the intuitive results but also calculations of errors, normalized errors, and NFE, offering a detailed and extensive numeric comparison between EGAE and MOGAE. The selected images in Fig. 12 contain potential noise, such as scale (present in all images in the first row) and a blue sign (found on the nevus image at the rightmost). The second row in Fig. 12 shows the delineations generated by SLIC, following the clinicians' guidelines.

[2] https://github.com/KhaosResearch/Plant-disease-explanation/blob/main/Appendix.pdf.

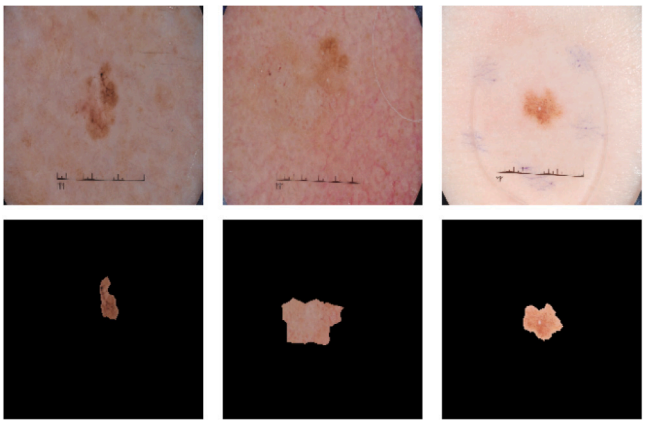


Fig. 12. Row 1 exhibits selected sample test images for explanation, arranged from left to right: melanoma, seborrheic-keratosis, and nevus. Row 2 displays delineations generated by SLIC, as per the clinicians' guidelines.

Table 7
Normalized error of explanation: LIME-I adopts the setting where the num_samples parameter is set to 5337, 7053, and 5727 for melanoma, seborrheic-keratosis, and nevus, respectively, to maintain identical conditions with EGAE. LIME-II, on the other hand, assigns the NFE of MOGAE (4651) to the num_samples parameter. The input images are segmented into 100 superpixels, and subsequently, the num_features parameter in LIME is set to 100.

Skin pigmentation image	LIME-I (Ribeiro et al., 2016)	LIME-II (Ribeiro et al., 2016)	EGAE with majority voting (Nematzadeh et al., 2023)	MOGAE
Melanoma	0.75	0.79	**0.16**	0.27
Seborrheic-keratosis	0.79	0.78	0.46	**0.38**
Nevus	0.71	0.69	0.57	**0.42**

Fig. 13 illustrates the explanations generated by MOGAE, with three attempts shown for each image. Each image is segmented into 100 superpixels (H = 100) using the SLIC segmentation algorithm. The figure demonstrates that MOGAE generally captures the informative areas of the image, although it also includes some minor non-informative areas in its explanations.

Fig. 14 shows how EGAE with majority voting (first row) and LIME (second row) explain the same images, with one attempt for each image. For a fair comparison, we assigned the average NFE of EGAE, calculated across five runs, to the num_samples parameter of LIME, as LIME does not implicitly calculate the intended number of images for evaluation and requires this input as a hyperparameter. Therefore, the num_samples parameter of LIME is set to 5337, 7053, and 5727 samples for melanoma, seborrheic keratosis, and nevus, respectively. Additionally, the num_features hyperparameter is set to 100, and both superpixels with positive (green) and negative (red) attributions are shown. It is evident that LIME identifies more areas (superpixels) of the image as positive attributions, visualized in green, compared to EGAE and even MOGAE in Fig. 13.

Table 7 shows the average normalized error of explanation across three runs of each method. The normalized error calculated in Table 7 also show that both EGAE with majority voting and MOGAE explain more accurately than LIME-I and LIME-II. In Table 7 LIME-I refers to the LIME in which the num_samples parameter is identically set, as stated in Fig. 14. However, in LIME-II we used the NFE of MOGAE which is 4651 for all three images. Table 7 indicates that assigning the NFE values of EGAE and MOGAE to LIME does not significantly affect the accuracy of the explanations. In terms of explanation accuracy for the melanoma pigmentation image, EGAE outperforms MOGAE. However, for seborrheic-keratosis and nevus, MOGAE provides more accurate explanations. It is important to note that MOGAE uses fewer images for

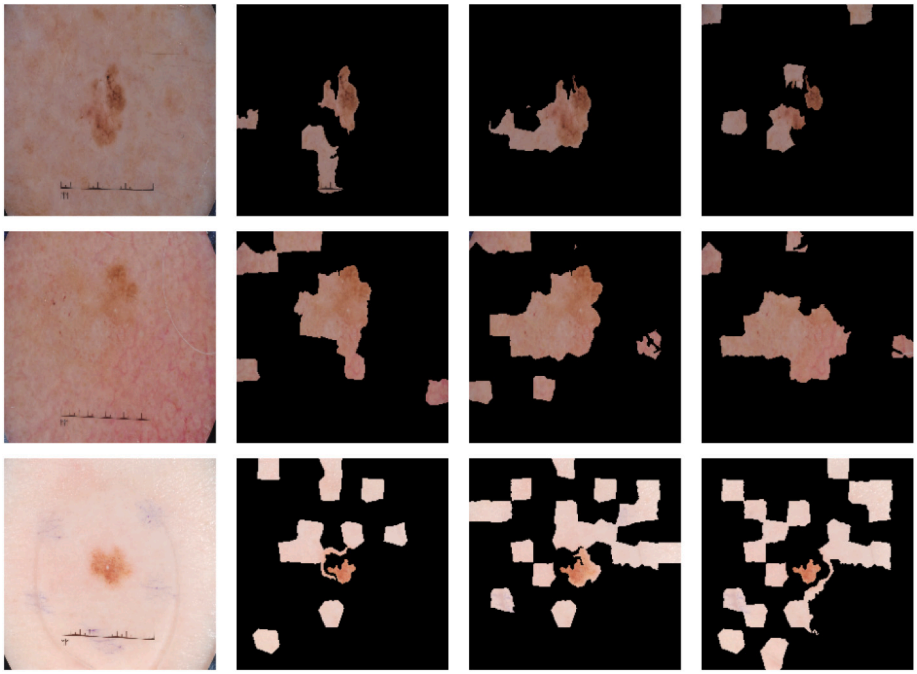


Fig. 13. Explanation of MOGAE when the input image is segmented into 100 superpixels (H = 100). In each row, from left to right, you can observe the original image followed by three attempts at explanation generated by MOGAE.

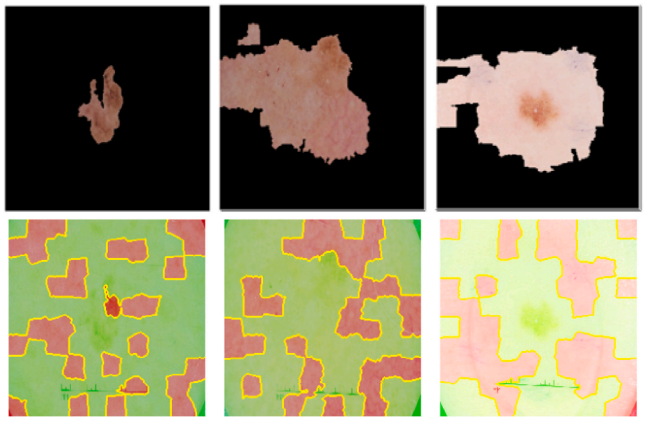


Fig. 14. Row 1 shows an attempt of EGAE explanation for, from left to right, melanoma, seborrheic-keratosis, and nevus. Row 2 shows the LIME explanation for the same images. The num_samples parameter in LIME is set to 5337, 7053, and 5727 for images in order of appearance from left to right. The input images are segmented into 100 superpixels, and subsequently, the num_features parameter in LIME is set to 100.

evaluations across all three skin pigmentation types. This aligns with the findings in Fig. 9 that when the number of superpixels is high (H = 100), MOGAE tends to be more efficient than EGAE.

Data availability

The related link to implementation, code, and data are provided in the paper.

References

Ahern, I., Noack, A., Guzman-Nateras, L., Dou, D., Li, B., Huan, J., 2019. NormLime: A new feature importance metric for explaining deep neural networks.

Barredo Arrieta, A., Díaz-Rodríguez, N., Del Ser, J., Bennetot, A., Tabik, S., Barbado, A., Garcia, S., Gil-Lopez, S., Molina, D., Benjamins, R., Chatila, R., Herrera, F., 2020. Explainable artificial intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI. Inf. Fusion 58, 82–115.

Bengamra Bidouk, S., Zagrouba, E., Bigand, A., 2023. Explainable AI for deep learning based potato leaf disease detection. In: 2023 IEEE International Conference on Fuzzy Systems.

Bhandari, M., Shahi, T.B., Neupane, A., Walsh, K.B., 2023. Botanicx-ai: Identification of tomato leaf diseases using an explanation-driven deep-learning model. J. Imaging 9 (2), 1–16.

Çetiner, H., 2022. Citrus disease detection and classification using based on convolution deep neural network. Microprocess. Microsyst. 95, 104687.

Chen, J., Zhang, D., Suzauddola, M., Zeb, A., 2021. Identifying crop diseases using attention embedded MobileNet-V2 model. Appl. Soft Comput. 113, 107901.

Dananjayan, S., Tang, Y., Zhuang, J., Hou, C., Luo, S., 2022. Assessment of state-of-the-art deep learning based citrus disease detection techniques using annotated optical leaf images. Comput. Electron. Agric. 193, 106658.

Elaraby, A., Hamdy, W., Alanazi, S., 2022. Classification of citrus diseases using optimization deep learning approach. Comput. Intell. Neurosci. 9153207.

Giuseppe, C., 2021. A ResNet-50-based convolutional neural network model for language id identification from speech recordings. In: Proceedings of the Third Workshop on Computational Typology and Multilingual NLP. pp. 136–144.

Hailesilassie, T., 2016. Rule extraction algorithm for deep neural networks: A review. Int. J. Comput. Sci. Inf. Secur. 14, 376–381.

Hryniewska, W., Grudzień, A., Biecek, P., 2022. LIMEcraft: Handcrafted superpixel selection and inspection for visual explanations. Mach. Learn..

Hurtado, S., Nematzadeh, H., García-Nieto, J., Berciano-Guerrero, M.-Á., Navas-Delgado, I., 2022. On the use of explainable artificial intelligence for the differential diagnosis of pigmented skin lesions. In: Bioinformatics and Biomedical Engineering: 9th International Work-Conference, IWBBIO 2022, Maspalomas, Gran Canaria, Spain, June 27–30, 2022, Proceedings, Part I. Springer-Verlag, Berlin, Heidelberg, pp. 319–329.

Khan, M.N., Das, S., Liu, J., 2024. Predicting pedestrian-involved crash severity using inception-v3 deep learning model. Accid. Anal. Prev. 197, 107457.

Kim, S.H., Park, J.S., Lee, H.S., Yoo, S.H., Oh, K.J., 2023. Combining CNN and Grad-CAM for profitability and explainability of investment strategy: Application to the KOSPI 200 futures. Expert Syst. Appl. 225, 120086.

Kovalev, M.S., Utkin, L.V., Kasimov, E.M., 2020. SurvLIME: A method for explaining machine learning survival models. Knowl.-Based Syst. 203, 106164.

Li, S., Li, T., Sun, C., Yan, R., Chen, X., 2023. Multilayer grad-CAM: An effective tool towards explainable deep neural networks for intelligent fault diagnosis. J. Manuf. Syst. 69, 20–30.

Lundberg, S., Lee, S.-I., 2017. A unified approach to interpreting model predictions. arXiv:1705.07874.

Macha, D., Kozielski, M., ukasz Wróbel, L., Sikora, M., 2022. RuleXAI—A package for rule-based explanations of machine learning model. Softw. 20, 101209.

Marques, G., Agarwal, D., de la Torre Díez, I., 2020. Automated medical diagnosis of COVID-19 through EfficientNet convolutional neural network. Appl. Soft Comput. 96, 106691.

Mehedi, M.H.K., Hosain, A.S., Ahmed, S., Promita, S.T., Muna, R.K., Hasan, M., Reza, M.T., 2022. Plant leaf disease detection using transfer learning and explainable AI. In: 2022 IEEE 13th Annual Information Technology, Electronics and Mobile Communication Conference. IEMCON, IEEE, pp. 0166–0170.

Nahiduzzaman, M., Chowdhury, M.E., Salam, A., Nahid, E., Ahmed, F., Al-Emadi, N., Ayari, M.A., Khandakar, A., Haider, J., 2023. Explainable deep learning model for automatic mulberry leaf disease classification. Front. Plant Sci. 14, 1–19.

Nematzadeh, H., García-Nieto, J., Navas-Delgado, I., Aldana-Montes, J.F., 2023. Ensemble-based genetic algorithm explainer with automized image segmentation: A case study on melanoma detection dataset. Comput. Biol. Med. 155, 106613.

Nimmy, S.F., Hussain, O.K., Chakrabortty, R.K., Hussain, F.K., Saberi, M., 2023. An optimized belief-rule-based (BRB) approach to ensure the trustworthiness of interpreted time-series decisions. Knowl.-Based Syst. 271, 110552.

Peltola, T., 2018. Local interpretable model-agnostic explanations of Bayesian predictive models via Kullback-Leibler projections. In: Workshop on Explainable Artificial Intelligence - Stockholm, Sweden.

Rabold, J., Deininger, H., Siebers, M., Schmid, U., 2020. Enriching visual with verbal explanations for relational concepts – combining LIME with aleph. In: Cellier, P., Driessens, K. (Eds.), Machine Learning and Knowledge Discovery in Databases. Springer International Publishing, pp. 180–192.

Rauf, H.T., Saleem, B.A., Lali, M.I.U., Khan, M.A., Sharif, M., Bukhari, S.A.C., 2019. A citrus fruits and leaves dataset for detection and classification of citrus diseases through machine learning. Data Brief 26, 104340.

Ribeiro, M.T., Singh, S., Guestrin, C., 2016. "Why should I trust you?": Explaining the predictions of any classifier. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. Association for Computing Machinery, pp. 1135–1144.

Ribeiro, M., Singh, S., Guestrin, C., 2018. Anchors: High-precision model-agnostic explanations. In: IEEE (Ed.), Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence.

Selvaraju, R.R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., Batra, D., 2017. Grad-CAM: Visual explanations from deep networks via gradient-based localization. In: 2017 IEEE International Conference on Computer Vision. ICCV, pp. 618–626.

Shaheed, K., Mao, A., Qureshi, I., Kumar, M., Hussain, S., Ullah, I., Zhang, X., 2022. DS-CNN: A pre-trained xception model based on depth-wise separable convolutional neural network for finger vein recognition. Expert Syst. Appl. 191, 116288.

Shi, S., Zhang, X., Fan, W., 2020. A modified perturbed sampling method for local interpretable model-agnostic explanation. CoRR abs/2002.07434.

Wei, K., Chen, B., Zhang, J., Fan, S., Wu, K., Liu, G., Chen, D., 2022. Explainable deep learning study for leaf disease classification. Agron. 12 (5), 1–15.

Wu, D., Zhao, J., 2022. Understand how CNN diagnoses faults with grad-CAM. In: Yamashita, Y., Kano, M. (Eds.), 14th International Symposium on Process Systems Engineering. In: Computer Aided Chemical Engineering, vol. 49, Elsevier, pp. 1537–1542.

Yang, G., Ye, Q., Xia, J., 2022. Unbox the black-box for the medical explainable AI via multi-modal and multi-centre data fusion: A mini-review, two showcases and beyond. Inf. Fusion 77, 29–52.

Zhou, B., Khosla, A., Lapedriza, A., Oliva, A., Torralba, A., 2015. Learning deep features for discriminative localization. CoRR.

Zhu, H., Wang, W., Ulidowski, I., Zhou, Q., Wang, S., Chen, H., Zhang, Y., 2023. MEEDNets: Medical image classification via ensemble bio-inspired evolutionary DenseNets. Knowl.-Based Syst. 280, 111035.