Received: July 11, 2023. Revised: August 13, 2023.

International Journal of
Intelligent Engineering & Systems

http://www.inass.org/

Cloud-based Paddy Plant Pest and Disease Identification using Enhanced Deep Metric Learning and k-NN Classification with Augmented Latent Fusion

Hendri Darmawan[1] **Mike Yuliana**[1]* **Moch. Zen Samsono Hadi**[1]

[1]*Department of Informatics and Computer Engineering, Politeknik Elektronika Negeri Surabaya, Surabaya, 60111, Indonesia*
*Corresponding author's Email: mieke@pens.ac.id

Abstract: Identifying pests and diseases is vital for rice farming, but the lack of experts often limits it. We propose a novel method that uses deep metric learning and k-NN classifier to classify pests and diseases in paddy plants. We experimented using different distance metrics, latent dimensions, base models, and optimizers to train the neural model that produces latent representation from images. Then, we used k-NN retrieval approach to find and label similar images from the latent space. During the inference phase, we manipulated the original image using various transformations and fused them with its latent representation to enhance the quality of latent space. The results show that our method surpasses the conventional deep learning classification (softmax classifier). Specifically, our method achieved maximum accuracy of 0.920 on ResNet-50 and 0.878 on ResNet-152. In comparison, the softmax classifier only achieved maximum accuracy of 0.752 on the same modeling scheme. Our method can produce more discriminative and robust data representations for classification tasks. Moreover, latent fusion from input augmentation during inference can also improve accuracy up to 19.2%. We also deployed the best model from our experiment on serverless cloud computing, allowing users to use the platform for prediction and monitoring through GIS.

Keywords: Paddy plant pest and disease, Deep metric learning, Distance metrics, k-NN, Latent fusion, Serverless cloud computing.

1. Introduction

Paddy plant pests and diseases are significant sources of anxiety for farmers, as they can reduce productivity or cause crop failure [1]. Identifying these issues requires knowledge and experience, but there is a shortage of officers to monitor and control them in Indonesia. Therefore, there is a need for an automated and accurate diagnosis system that can help farmers detect and manage paddy health problems. Recent advances in computer vision have enabled fast and accurate diagnosis of plant diseases. However, traditional image recognition algorithms have limited performance for plant pest and disease recognition, as mentioned by Liu et al. [2]. They face challenges such as similar symptoms, slight leaf-background differences, low contrast, considerable leaf scale variation, and noisy leaf images. Therefore, we need an advanced identification algorithm to overcome these problems. Deep learning (DL) can automatically extract features using convolutional kernels optimized with specific algorithms. However, conventional deep learning classification has a drawback where the extracted features are difficult to understand intuitively, and the model cannot rationalize its predictions. Moreover, deep learning classification is based on supervised learning, which requires a large amount of labeled data for each class. The output is a fixed probability model, which makes them less adaptable to new domains.

To overcome these limitations, we propose an enhanced classification algorithm, using latent fusion augmented images incorporating deep metric learning with k-NN approximation (FADMAKA). Deep metric learning (DML) is an approach that focuses on measuring the similarity or dissimilarity between samples by using a neural model to learn the optimal distance metric [3]. DML utilizes a neural model to learn an optimal distance metric, enabling automated feature extraction and non-linear transformations into a latent space [4]. This latent space allows for interpretability and flexibility, making the model's predictions more intuitive and adaptable to new domains. These representations possess heightened discriminative qualities and can be effectively separated by interpretable machine learning models like k-NN, SVM, or logistic regression [5]. We employed various augmentation techniques during inference, including rotation, flipping, and shifting, to generate distinct latent representations for each image. These representations were averaged to create a final representation, termed "augmented latent fusion," to enhance robustness and diversity. Subsequently, we adopted k-NN classification to determine each image's class by comparing its latent representation with all training images in the latent database. The chosen k-NN classifier offers simplicity, interpretability, and non-parametric properties. We deployed our optimal model to the cloud, providing a scalable and accessible solution for paddy plant pest and disease identification that can be accessed through a browser. We created a world map service that shows the time-series data to track the prediction over time. We also suggested some solutions and preventive actions to deal with the identified threats on our platform.

The paper's subsequent sections discuss related works on plant pest and disease recognition (section 2), provide a detailed explanation of the proposed method (section 3), present experimental setup, results, and analysis (section 4), and conclude with a summary of the main findings (section 5).

2. Literature study

The discussion in this section focuses on several state-of-the-art methods for identifying pests and diseases in paddy plants. Lu et al. [6] utilized median filtering, histogram equalization, and edge segmentation techniques on the rice sheath blight images. They concatenated the color features (R, G, and B values) with the texture features (mean, contrast ratio, and entropy), which were then used as input for the neural network. However, this method has limited feature representation and may not capture the complex patterns of images. High-dimensional input data can also result from this technique, which can cause the curse of dimensionality and raise the risk of overfitting. Ni et al. [7] proposed a method that uses data augmentation and deep learning to classify rice pests and diseases. They also designed a new model, RepVGG_ECA, that adds efficient channel attention (ECA) attention mechanism to the RepVGG model to enhance the feature extractor. This approach relies on supervised learning, which necessitates a substantial volume of labeled data, leading to its heavy reliance on labeled samples. DML can leverage unlabeled data, reducing the dependance on labeled samples because it only requires a distance function that can compare pairs or triplets of samples rather than their actual class labels. Malathi et al. [8] utilized deep learning techniques for classifying ten pests in paddy crops. They employed augmented pest images for training and experimented with various DCNN architectures. The ResNet-50 model with fine-tuning achieved the highest accuracy. This method also relies on supervised learning, leading to limited adaptability to new domains due to fixed probability outputs. Moreover, the prediction output is hard to intuitively understand. Rahman et al. [9] proposed deep learning-based methods for classifying diseases and pests in rice plants using CNNs. They experimented with two types of CNN architectures: large-scale and small-scale architectures. They also introduced a new architecture that used a simple CNN with lower model complexity but high performance. The best model with the highest accuracy was VGG16 with fine-tuned training. This method is also based on supervised learning, which has the same limitations as the previous methods.

Our proposed method introduced several improvements over the existing methods. First, unlike previous research that only focused on predicting pests or diseases separately, our study could identify both pests and diseases simultaneously. Second, our method used DML with augmented latent fusion instead of conventional deep learning. Third, we could enhance the classifier outcome with new samples without retraining the model, as we could expand the database with new samples to improve the k-NN algorithm. Fourth, we also deployed the best model to the cloud, which offered a scalable and accessible solution not only in research paper but also in practice. Table 1 provides a snapshot of some relevant research in identifying pests and

Table 1. Related works on paddy plant pest and disease

Approach	**Method**	**Object**		**Deploy**
		Pest	**Disease**	
Lu et al. [6]	DL	–	✓	–
Ni et al. [7]	DL	✓	✓	–
Malathi et al. [8]	DL	✓	–	–
Rahman et al. [9]	DL	✓	✓	–
Ours	DML	✓	✓	✓

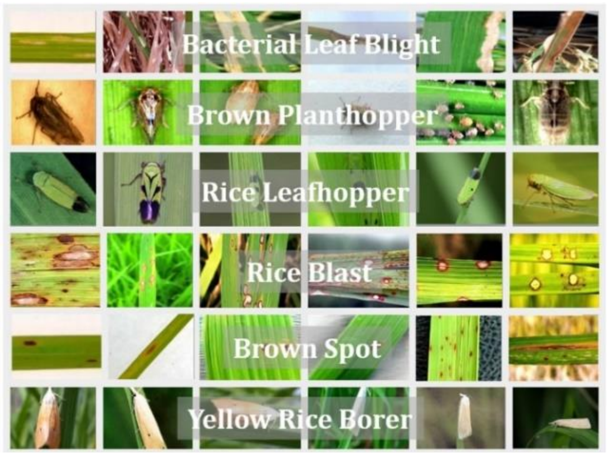

Figure. 1 Sample images for each class

diseases in rice plants and how it compares to our work.

3. Proposed methodology

In this section, we will describe the overall research framework, which has five main components: (1) data preparation consisting of image acquisition and augmentation; (2) image pre-processing; (3) triplet sampling and model development; (4) accuracy testing and comparison with baseline and state-of-the-art methods; and finally (5) model deployment.

3.1 Image acquisition

In this study, we acquired image data from various sources, such as Large-scale dataset for identifying insect pests: IP102 (field environment) [10], Rice leaf diseases data set (white background) [11], Rice plant diseases (field environment) [12], and also some images scraped from Google images. Fig. 1 illustrates sample images for each class. We curated different datasets with varied conditions to ensure image quality and diversity. Manual analysis and cropping were conducted to augment the image samples, especially for cases with multiple diseases in a single image. Duplicate images were removed, and mislabeled images were corrected. The data was split into three parts: training set (80%), validation set (20%), and holdout set (100 new samples per class). Table 2 presents the data distribution across these sets.

3.2 Image augmentation

Image augmentation varies training images to increase their number and address class imbalance [13]. For augmentation during training, we used random rotation with an angle range from -15 degrees to +15 degrees represented by Eq. (1), horizontal flip represented by Eq. (2), and blur up to m = 1 pixel represented by Eq. (3).

$$T_R(x,\gamma) = x'(p\cos\gamma - q\sin\gamma, p\sin\gamma + q\cos\gamma, C) \quad (1)$$

$$T_H(x) = x'(p, W-q-1, C) \quad (2)$$

$$T_B(x,m) = \frac{1}{(2m+1)^2}\sum_{i=-m}^{m}\sum_{j=-m}^{m} x'(p+i, q+j, C) \quad (3)$$

where p, q are the pixel coordinates in the image x, γ is angle of rotation, W is the image's width, m is the radius of the square region used in the box blur filter, and C is the channel.

3.3 Image pre-processing

We used a uniform resolution of 64x64 pixels with RGB channel. The image contrast was enhanced by adjusting the image intensity range to a broader target range using the contrast stretching method, calculated by Eq. (4) [14].

$$P_{out\_C} = \left(P_{in\_C} - c_C\right)\frac{(b_C - a_C)}{(d_C - c_C)} + a_C \quad (4)$$

where P_{out_C} is the output pixel, and P_{in_C} is the input pixel. The formula stretches the input range $[c_C, d_C]$ to the output range $[a_C, b_C]$ for each channel $C \in \{R, G, B\}$. After that, the image pixels were scaled to be between 0.0 and 1.0, then normalized with mean and standard deviation using Eq. (5).

$$x_{norm\_C} = \frac{x_C - \mu[C]}{\sigma[C]} \quad (5)$$

where x_{norm_C} is the normalized image and x_C is the original image for each channel. Each channel's mean and standard deviation values are calculated from ImageNet, given by $\mu = [0.485, 0.456, 0.406]$ and $\sigma = [0.229, 0.224, 0.225]$, arrays of length 3 to the respective RGB channel.

3.4 Triplet sampling

Training and validation sets were restructured into anchor, positive, and negative parts, as shown in Fig. 2, to train the triplet network that maps images to the latent space. In latent space, we aim for the anchor and positive images (same class) to be close, while the anchor and negative images (different classes) are distant. Triplet sampling is vital for learning a discriminative feature. The triplets should be informative and challenging, meaning that the

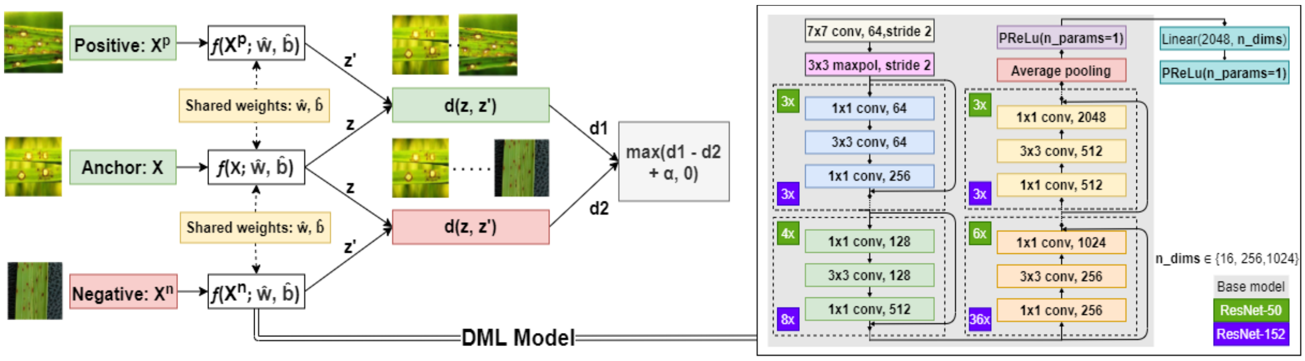

Figure. 2 Triplet network

Table 2. Distribution of data for different pathogens

Class	Pathogen name	Amount of data		
		Train	Validation	Holdout
0	Bacterial leaf blight	1171	293	100
1	Brown planthopper	1175	293	100
2	Rice leafhopper	1435	359	100
3	Rice blast	2526	641	100
4	Brown spot	1298	324	100
5	Yellow rice borer	1254	314	100

positive image is similar but not identical to the anchor image, and the negative image is dissimilar but not too distant from the anchor image.

3.5 Phase 1: DML model

The DML model is part of deep metric learning that maps high-dimensional data to low-dimensional space (latent). In this study, we fine-tuned IMAGENET1K_V2 pre-trained weights on ResNet-50 and ResNet-152 as base models in our case [15]. ResNet-50 and ResNet-152 are CNN models with 50 and 152 layers, respectively, organized into residual blocks with skip connections. Both models use bottleneck design to decrease computation cost and parameter size. We used parametric rectified linear unit (PReLU) activation function, as shown in Fig. 2. PReLU improves ReLU by using a learnable parameter to control the slope of the negative part of the activation function [16]. The DML model learns discriminative representations based on the distance between latent representations, optimized with an objective function that measures the similarity or dissimilarity.

3.5.1. Objective function

Triplet margin loss decreases the distance between anchor input (X_i) and positive input (X^p), while increasing the distance between the anchor input (X_i) and a negative input (X^n) in the latent space. Let $f(\cdot)$ denotes DML model, where $f(X_i; w_t, b_t)$ is the DML model that is fed with X_i from N samples. The terms w_t and b_t are weights and biases that will be optimized using Eq. (6). We obtain latent representation as the output for anchor, positive, and negative input, then calculate d, i.e. a specific distance metric, calculated using Eqs. (7-9). We added a margin to increase the distance between the anchor and the negative, denoted as α of 1.0. In principle, a more significant margin produces a more distinct latent space but makes converging the training process harder.

$$\hat{w}, \hat{b} = \underset{w_t, b_t}{\arg\min}\ \mathcal{L}_T$$
$$\mathcal{L}_T = \sum_{i=1}^{N} max\Big(d\big(f(X_i; w_t, b_t), f(X^p;\ w_t, b_t)\big) - d\big(f(X_i;\ w_t, b_t), f(X^n;\ w_t, b_t)\big) + \alpha, 0\Big) \tag{6}$$

The latent dimension is given by h, finding h is a trade-off between efficient (small size) and effective (large size) [17]. We tested latent dimensions for $h \in \{16, 256, 1024\}$, and we varied the epochs based on latent space size: 100 for 16, 200 for 256, and 300 for 1024. In this study, we experimented using several metrics, including Euclidean distance calculated using Eq. (7) [18], cosine distance calculated using Eq. (8) [19], and Pearson correlation calculated using Eq. (9) [20-21]. Pearson correlation is not a distance function because it violates triangle inequality. However, Pearson correlation can measure the linear relationship between two latent to estimate the similarity by indicating their linear closeness.

$$d(z, z') = \sqrt{\sum_{i=1}^{h} (z_i - z'_i)^2} \tag{7}$$

$$d(z, z') = 1 - \frac{\sum_{i=1}^{h} z_i z'_i}{\sqrt{\sum_{i=1}^{h} z_i^2 \sum_{i=1}^{h} z'^2_i}} \tag{8}$$

$$d(z, z') = 1 - \frac{\sum_{i=1}^{h} (z_i - \bar{z})(z'_i - \bar{z}')}{\sqrt{\sum_{i=1}^{h} (z_i - \bar{z})^2 \sum_{i=1}^{h} (z'_i - \bar{z}')^2}} \tag{9}$$

Algorithm 1: FADMAKA with augmented set 3

```
1   Input:
      Training set $\mathcal{D}_t = \{(x_i^t, y_i^t)\}_{i=1}^{n_t}$,
      Validation set $\mathcal{D}_v = \{(x_i^v, y_i^v)\}_{i=1}^{n_v}$,
      Holdout set $\mathcal{D}_e = \{(x_i^e, y_i^e)\}_{i=1}^{n_e}$,
2   Data augmentation and pre-processing
      - Augment $\mathcal{D}_t$ by Eq. (1) to Eq. (3)
      - Pre-process $x_i^t, x_i^v, x_i^e$ by Eq. (4) to Eq. (5)
3   Triplet sampling
      - Resample $\mathcal{D}_t$and $\mathcal{D}_v$ as triplet sample
4   Phase 1: develop DML model
      - Define DML model $f(\cdot)$ that outputs latent representation $z \in R^h$
5   Define objective function
      - Define triplet loss using Eq. (6) with a specific distance metric from Eq. (7) to Eq. (9)
6   DML model training
      - Train $f(\cdot)$ using $x_i^t$ for obtaining final $\widehat{w}$ and $\hat{b}$ by solving Eq. (6) using optimization SGD in Eq. (10) or AdamW in Eq. (11)
      - Validate and checkpoint $f(\cdot)$ using $\mathcal{D}_v$
7   Compute latent for training and validation set
      - $\mathrm{z}^{\mathrm{t}} = \{f(x_i^t; \widehat{w}, \hat{b}) \mid x_i^t \in \mathcal{D}_t\}$
      - $\mathrm{z}^{\mathrm{v}} = \{f(x_i^v; \widehat{w}, \hat{b}) \mid x_i^v \in \mathcal{D}_v\}$
8   Phase 2: k-nearest neighbors
      - Find the optimal $k$ value that maximizes the accuracy of $\mathrm{z}^{\mathrm{v}}$ in $\mathrm{z}^{\mathrm{t}}$ database
9   Compute latent for holdout set and classify
      - $z_i^e = \{f(x_i^e; \widehat{w}, \hat{b}) \mid x_i^e \in \mathcal{D}_e\}$
      - $z_i^r = \{f(T_R(x_i^e, \gamma);\ \widehat{w}, \hat{b}) \mid x_i^e \in \mathcal{D}_e\}$
      - $z_i^h = \{f(T_H(x_i^e);\ \widehat{w}, \hat{b}) \mid x_i^e \in \mathcal{D}_e\}$
      - $z_i^s = \{f\big(T_S(x_i^e, (\Delta p, \Delta q))\big);\ \widehat{w}, \hat{b}) \mid x_i^e \in \mathcal{D}_e\}$
      - $z_i^f = \frac{1}{4}(z_i^e + z_i^r + z_i^h + z_i^s)$
      - k-NN $z_i^f$ with $k$ from step 8 in $\mathrm{z}^{\mathrm{t}}$ database
10  Output: predicted class of $x_i^e$
```

where $z \in \mathbb{R}^h$ to denote the latent space of X_i and $z' \in \mathbb{R}^h$ to denote the latent space of any other input (X^p or X^n). $\bar{z}$ and $\bar{z}'$ is the average of the latent representation of z and z' that can be calculated by $\bar{z} = \frac{1}{h}\sum_{i=1}^{h} z_i$ and $\bar{z}' = \frac{1}{h}\sum_{i=1}^{h} z'_i$.

3.5.2. Optimization algorithm

The optimization algorithm aims to adjust the neural model weights to achieve the minimum loss value. We experimented with different optimizers, including SGD and AdamW. SGD works by updating model weights by taking small steps in the opposite direction of the gradient of the objective function. The equation for SGD algorithm is shown in Eq. (10).

$$\theta_{t+1} = \theta_t - \eta \nabla_\theta \mathcal{L}_T(w_t, b_t) \tag{10}$$

where $\theta_t = [\mathrm{w_t}, \mathrm{b_t}]$ and $\theta_{t+1} = [\mathrm{w_{t+1}}, \mathrm{b_{t+1}}]$ are the model parameters at iteration t and $\mathrm{t}+1$. Note that $\mathrm{w_{t+1}}$ is updated the weight parameter, $\mathrm{b_{t+1}}$ is updated the bias parameter, η is the learning rate, and $\nabla_\theta \mathcal{L}_T(w_t, b_t)$ are the gradients of the loss function with respect to w_t and b_t. On the other hand, AdamW is an adaptive optimization algorithm that decouples weight decay regularization terms from the gradient update, which can reduce the large weights on the model during training to avoid overfitting. The equations for AdamW algorithm are denoted as follows Eq. (11) [22].

$$\begin{gathered} m_{t+1} = \beta_1 m_t + (1-\beta_1)\nabla_\theta \mathcal{L}_T(w_t, b_t) \\ v_{t+1} = \beta_2 v_t + (1-\beta_2)\big(\nabla_\theta \mathcal{L}_T(w_t, b_t)\big)^2 \\ \widehat{m}_{t+1} = \frac{m_{t+1}}{1-\beta_1^{t+1}} \\ \hat{v}_{t+1} = \frac{v_{t+1}}{1-\beta_2^{t+1}} \\ \theta_{t+1} = \theta_t - \eta\lambda\theta_t - \frac{\eta \widehat{m}_{t+1}}{\sqrt{\hat{v}_{t+1}}+\epsilon} \end{gathered} \tag{11}$$

where m_{t+1} and v_{t+1} are the first and second-moment estimates of the gradient with respect to the parameters, $\widehat{m}_{t+1}$ and $\hat{v}_{t+1}$ are the bias-corrected estimates of m_{t+1} and v_{t+1} to improve reliability, β_1 and β_2 are exponential decay rates for the moment estimates, we used 0.9 and 0.999, respectively. ϵ is a small constant to prevent division by zero, we used 1e-8, and λ is the weight decay coefficient we used 1e-2. We used the initial learning rate η of 0.001 for SGD and AdamW. A large learning rate may miss the minimum, while a small one may approach too slowly or get trapped in local minima [23]. However, AdamW is a method that adapts the learning rate for different parameters because it uses the average of the first and second moments of the gradients.

3.6 Phase 2: classification task

In this research, we compared two classifiers: FADMAKA with k-NN for our proposed method and softmax classifier to implement the baseline and state-of-the-art methods as a comparison model.

3.6.1. Extract and fuse latent space

After optimizing the DML model, we got the final weights and biases, denoted as $\widehat{w}$ and $\hat{b}$. We obtain the latent space using $f(\cdot)$ for the training set, denoted as z^t, the validation set, denoted as z^v, and the holdout set, denoted as z^e. We use n_t, n_v, and n_e to denote the number of train, validation, and holdout sets, respectively. They are defined as follows Eq. (12).

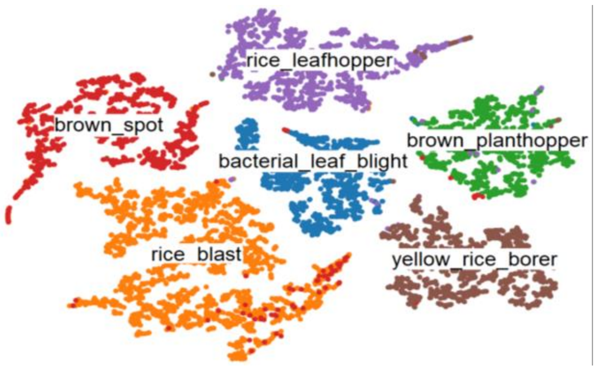

Figure. 3 2D latent space using t-SNE: ResNet-50 with AdamW, Pearson 16dims

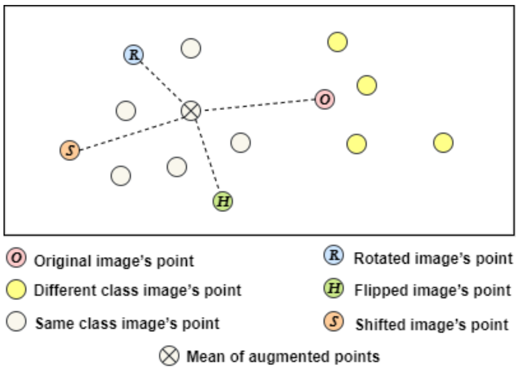

Figure. 4 Augmented latent fusion anatomy

$$\begin{aligned} z^t &= \{z_i^t \mid i = 1, \dots, n_t\} \\ z^v &= \{z_i^v \mid i = 1, \dots, n_v\} \\ z^e &= \{z_i^e \mid i = 1, \dots, n_e\} \end{aligned} \tag{12}$$

where $z_i^t = f(x_i^t;\ \widehat{w}, \widehat{b})$, $z_i^v = f(x_i^v;\ \widehat{w}, \widehat{b})$, and $z_i^e = f(x_i^e;\ \widehat{w}, \widehat{b})$. We also used random augmentation during inference to improve representative latent space, as different image matrices can produce different latent representations. We obtain the latent space for the rotated holdout set, denoted as z^r, the horizontal flipped holdout set, denoted as z^h, and the shifted holdout set, denoted as z^s. Note that $n_e = n_r = n_h = n_s$ is the number of holdout images. They are defined as follows Eq. (13).

$$\begin{aligned} z^r &= \{z_i^r \mid i = 1, \dots, n_r\} \\ z^h &= \{z_i^h \mid i = 1, \dots, n_h\} \\ z^s &= \{z_i^s \mid i = 1, \dots, n_s\} \end{aligned} \tag{13}$$

where $z_i^r = f(T_R(x_i^e, \gamma);\ \widehat{w}, \widehat{b})$, $z_i^h = f(T_H(x_i^e);\ \widehat{w}, \widehat{b})$, and $z_i^s = f(T_S(x_i^e, (\Delta p, \Delta q));\ \widehat{w}, \widehat{b})$. Here, T_R is the random rotation function stated in Eq. (1) with $\gamma \sim U(-15{,}15)$. T_H is horizontal flipping function stated in Eq. (2), and T_S is random shifting function from -10 to +10 in the x and y axes, stated in Eq. (14).

$$T_S(x, (\Delta p, \Delta q)) = x'(p - \Delta p, q - \Delta q, C) \tag{14}$$

where $(\Delta p, \Delta q) \sim U(-10{,}10) \times U(-10{,}10)$. After that, we fused the latent space by calculating the average of each dimension, where denoted as z^f. The calculation for the augmented set 3 is as follows Eq. (15).

$$z^f = \{z_i^f \mid i = 1, \dots, n_f\} \tag{15}$$

where $z_i^f = \frac{1}{4}(z_i^e + z_i^r + z_i^h + z_i^s)$ and n_f is the number of element latent set, which is equal to the number of holdout set. Depending on the augmented set scenario, we may calculate the average of different latent spaces. For the augmented set 2, we averaged z^e, z^r, and z^h. For the augmented set 1, we only averaged z^e and z^r.

3.6.2. k-NN

We used k-NN to classify the latent representation as it is easily separable with distance information. Algorithm 1 contains a summary of the FADMAKA method we propose. For an input image during inference, we retrieved similar images from z^t, i.e. latent space of the training set. The k-NN algorithm ranks the scores and selects a few nearest neighbors to assign the class prediction of the input image. Fig. 3 shows the 2D representation of the latent space z^t, which we obtained by applying t-SNE dimensionality reduction with perplexity 15 with automatically optimized learning rate [24]. The figure reveals that each class forms its own cluster and is well separated from the others. However, some classes are not well separated and are mixed with the wrong clusters. It may indicate they have similar features, such as brown spot and rice blast. This is reasonable because some images of rice blast also show brown spot on the same leaf. Augmented latent fusion can improve the robustness of latent representation, as shown in Fig. 4. The original latent point is close to a different class that it does not belong to. But when we use the augmented input and take the average of the latent points along each dimension, the augmented point shifts towards the same class. Nevertheless, it also means that we need to calculate more than one latent representation for each input during inference, depending on how much we vary the augmentation.

3.6.3. Softmax classifier

We compared the accuracy of FADMAKA with the softmax classifier (baseline) using fine-tuned

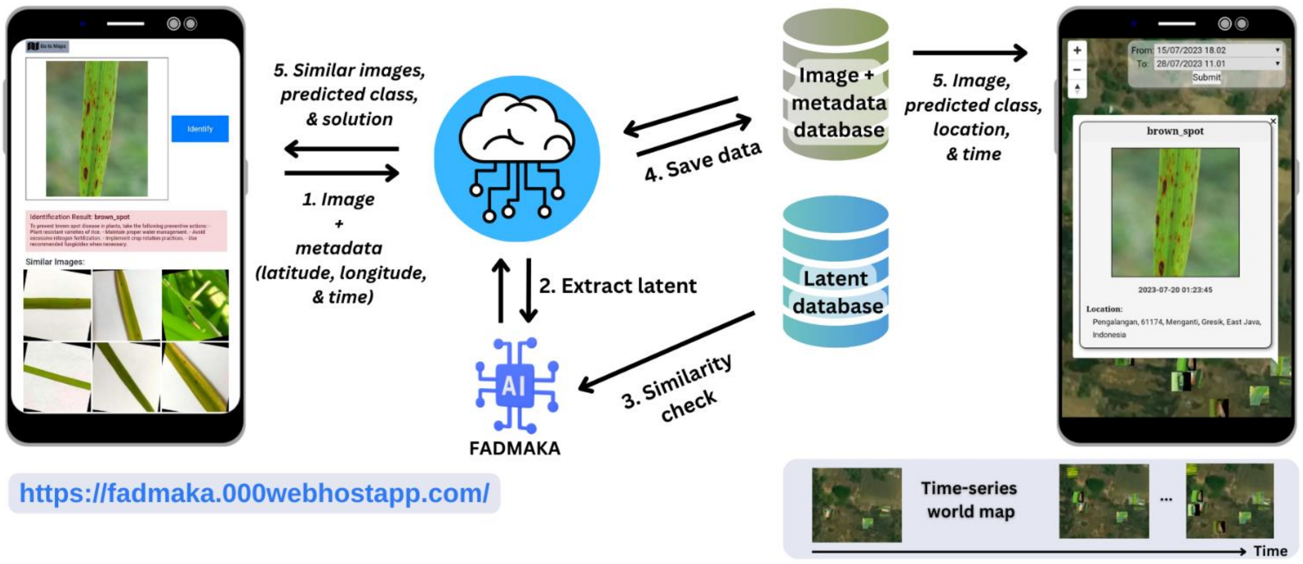
5. Similar images, predicted class, & solution
1. Image + metadata (latitude, longitude, & time)
2. Extract latent
3. Similarity check
4. Save data
Image + metadata database
Latent database
5. Image, predicted class, location, & time
FADMAKA
Identify
Identification Result: brown_spot
Similar Images:
brown_spot
Location:
Submit
https://fadmaka.000webhostapp.com/
Time-series world map
Time

Figure. 5 Implementation system for identification and location mapping

techniques also on ResNet-50 and ResNet-152 base models. We trained the models for 300 epochs, a sufficiently high number to avoid underfitting. We also implemented checkpointing weights to prevent overfitting issues [23]. For these scenarios, we appended a fully connected layer with six nodes and a softmax layer in the last layer of the base model from Fig. 2 for the classification task. We experimented by changing optimizers with SGD and AdamW. Furthermore, we also reproduced the state-of-the-art methods using a softmax classifier and compared their accuracy with FADMAKA. We applied the categorical cross-entropy loss function as the standard loss function for multi-class deep learning classification Eq. (17) [25].

$$\begin{gathered} \widehat{W}, \widehat{B} = \underset{w_t, b_t}{\arg\min}\, \mathcal{L}_X \\ \mathcal{L}_X = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{K} y_{ij}\log\left(s_{ij}\right) \end{gathered} \tag{17}$$

where N is the number of samples, K is the number of classes, y_{ij} is the true label of the i-th sample for the j-th class, and s_{ij} is the predicted probability from softmax layer of the i-th sample for the j-th class.

3.7 Model deployment

We deployed the best model from this research on a cloud platform using a serverless cloud service with FastAPI and GIS for the mapping system, as shown in Fig. 5. In the implementation phase, the user needs to take a picture with GPS enabled on their device, and then send the data to our API. The DML model then extracts the latent representation from the image and compares it with the latent in the database using k-NN. After that, the algorithm predicts the threat class, writes it to the EXIF image along with the GPS location and time, and saves it in the image database. The user can monitor the prediction results on our world map service, which uses a time-series world map to track environmental changes [26]. We also used reverse geocoding technique to find the human-readable address based on its coordinates from the EXIF data. Besides the time-series world map that can help monitor pest or disease outbreaks in a specific region, we also provide recommendations based on the predicted threat to assist the user in managing the problem. The user can also see the predicted class and similar images on the main page as feedback to verify the prediction by looking at the retrieved images from the database, an advantage of FADMAKA that can provide interpretable results, unlike conventional deep learning classification.

Moreover, we can enrich the database with new samples to enhance the classifier without retraining the DML weights. Theoretically, using the few-shots learning approach, we can also directly apply it to identify many other pests and diseases by saving the latent representation from the new classes in the latent database. This is possible because the DML model does not output a probability distribution that sums up to one like a softmax classifier does. Instead, it outputs a latent representation that captures the similarity or dissimilarity between images. The DML model learns to distinguish between images by creating a considerable margin distance in the latent space [27].

4. Experimental results and discussion

In this experiment, we ran our model on a hardware device consisting of an Intel® Xeon® CPU @ 2.00GHz with 13GB of RAM and a Tesla P100-PCIE 16GB graphics card (CUDA 11.4) with the seed 2023 for the reproducibility of the random process. We used three different sets of data for training, validation, and testing. The training set was used to train the model by adjusting the model parameters through a backpropagation algorithm with a batch size of 16. The validation data was used to select the best model during training using checkpointing and to choose the k value for the k-NN algorithm. After completing the training, we loaded the best model parameters from the checkpoint to evaluate the holdout set.

4.1 Choosing k from validation set's latent space

This subsection aims to determine the optimal value of k that produces the best FADMAKA testing accuracy for each scenario on the validation set. We can observe in Fig. 6, that models perform best at a specific $k \in \{1,6,11,\ldots,196\}$. Thus, selecting the appropriate k value is vital for developing the model with high and stable performance on unseen data. According to Fig. 6, the median of maximum accuracy on the validation set for ResNet-50 with SGD is 0.872, while for AdamW it is 0.970. For ResNet-152, the median of maximum accuracy with SGD is 0.869, while for AdamW it is 0.986. AdamW achieves higher median maximum accuracy of all tested scenarios compared to SGD. This is likely due to its ability to adaptively adjust the learning rate during training and maintain optimal weight values, because of weight decay regularization which reduces overfitting. In contrast, SGD may experience fluctuations in the gradient, leading to oscillations in the objective function and longer convergence times. Additionally, SGD may also be prone to overfitting when using non-Euclidean distance metrics in higher dimensions as shown in Fig. 6. Higher base model complexity can also lead to higher median accuracy in the AdamW scenario. This may be because higher DML model complexity optimized by SGD can also increase the risk of overfitting.

4.2 Evaluation on holdout set

In this subsection, we evaluated holdout set using k values from the validation set in the previous sub-section and the k-NN classifier on the latent space database from the train set. Holdout data measures how well the model can predict unseen data, not part of the training or validation process. Tables 3 and 4 show the FADMAKA accuracy with different distance metrics, optimizers, and latent dimensions using holdout set. Based on the data presented, the median accuracy for ResNet-50 with SGD is 0.708, while for AdamW it is 0.745. In comparison, the median accuracy for ResNet-152 with SGD is 0.715, while for AdamW it is 0.755. Our findings on holdout evaluation indicate that AdamW outperforms SGD and that ResNet-152 achieves higher accuracy than ResNet-50. It is in line with our previous analysis using the validation set. ResNet-152 has more layers and parameters than ResNet-50, which allows it to extract more complex and deep features from the data. However, more layers and parameters imply that the model needs more time and resources for inference and the risk of overfitting. The results in Tables 3 and 4 also show that increasing the latent dimension does not constantly improve the holdout accuracy, even with more training epochs, because a higher latent dimension means that the model can learn more complex or diverse features, but it also increases the risk of overfitting, which is when the model memorizes the training data and fails to generalize to unseen data. The optimal ResNet-50 model had a latent dimension of 16, 36 k nearest neighbors, was optimized using AdamW with Pearson correlation, and achieved an accuracy of 0.772. In contrast, the best ResNet-152 model had a latent dimension of 1024, 6 k nearest neighbors, was optimized using AdamW with Euclidean distance, and achieved an accuracy of 0.789.

4.3 Comparison with the baseline models

We evaluated the performance of the best models from FADMAKA schemes by comparing its accuracy with the baseline softmax classifier performed on holdout set as shown in Table 5 and Table 6. FADMAKA without augmented latent outperformed the deep learning classification, achieved an accuracy of 0.772 on ResNet-50 with a latent dimension of 16 and a computation time of 5.59 ms, and an accuracy of 0.789 on ResNet-152 with a latent dimension of 1024 and a computation time of 31.45 ms. Moreover, fusing augmented latents on images can improve FADMAKA accuracy, but it also increases computation time. For ResNet-50, the accuracy increases by 11%, 15%, and 19.2% with the fusion of augmented type 1, type 2, and type 3, respectively, and the computation times are 11.6 ms, 16.06 ms, and 20.96 ms, respectively. For ResNet-152, the accuracy increases by 2.4%, 6.7%, and 11.3% with the fusion of augmented type 1, type 2, and type 3, respectively, and the computation times are 44.9 ms, 58 ms, and 70.35 ms, respectively. We

Received: July 11, 2023. Revised: August 13, 2023.

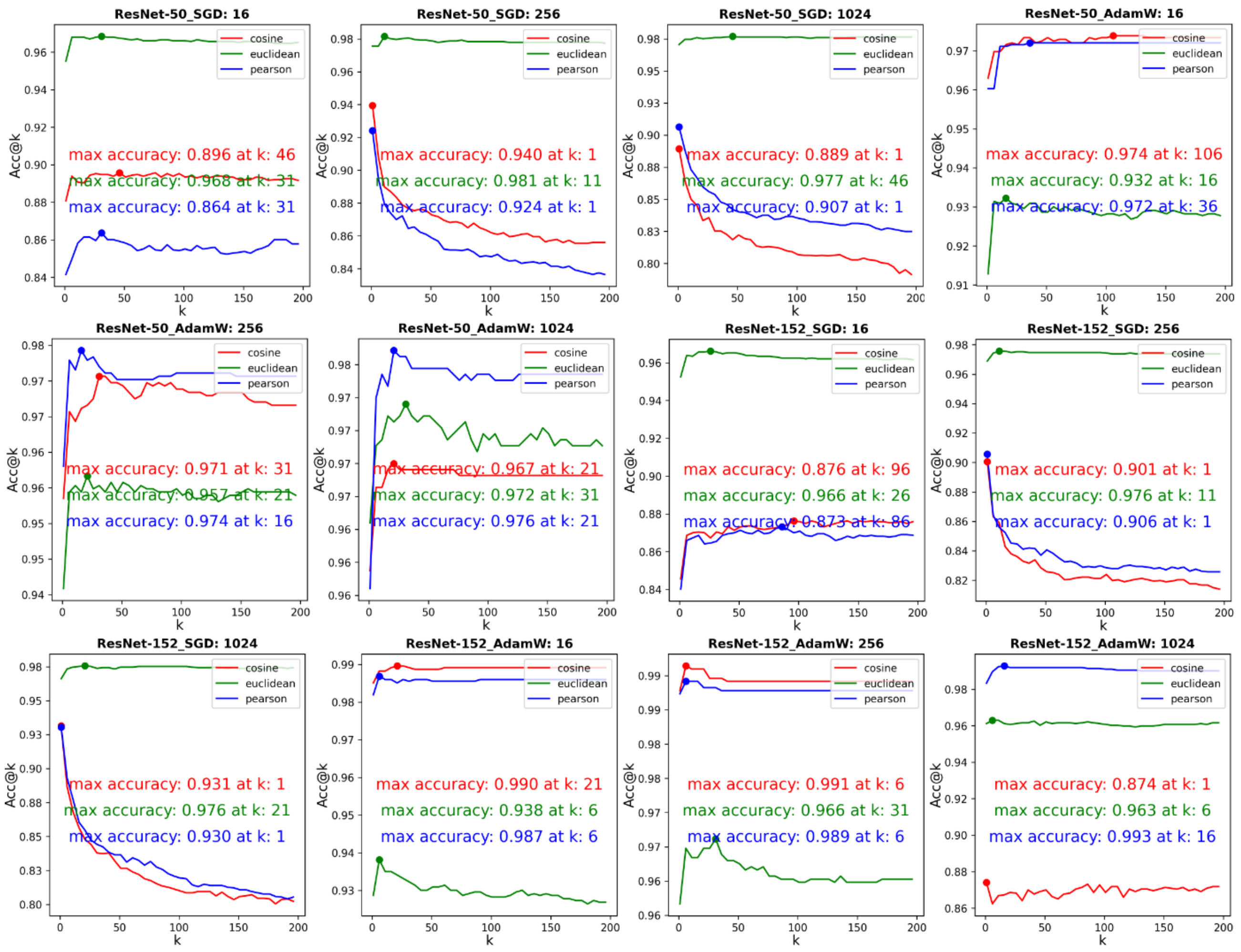

Figure. 6 Validation set accuracy of DML with various k-NN settings for all FADMAKA schemes

Table 3. Holdout set accuracy of DML with the best k-NN settings using ResNet-50

Optimizer	**Latent dimension**	**Max epochs**	**Cosine distance**		**Euclidean distance**		**Pearson correlation**	
			k-NN	**Accuracy**	**k-NN**	**Accuracy**	**k-NN**	**Accuracy**
SGD	16	100	46	0.679	31	0.730	31	0.690
	256	200	1	0.733	11	0.752	1	0.723
	1024	300	1	0.687	46	0.707	1	0.708
AdamW	**16**	**100**	106	0.745	16	0.745	**36**	**0.772**
	256	200	31	0.740	21	0.742	16	0.712
	1024	300	21	0.753	31	0.760	21	0.738

Table 4. Holdout set accuracy of DML with the best k-NN settings using ResNet-152

Optimizer	**Latent dimension**	**Max epochs**	**Cosine distance**		**Euclidean distance**		**Pearson correlation**	
			k-NN	**Accuracy**	**k-NN**	**Accuracy**	**k-NN**	**Accuracy**
SGD	16	100	96	0.723	26	0.733	86	0.710
	256	200	1	0.667	11	0.750	1	0.715
	1024	300	1	0.703	21	0.660	1	0.752
AdamW	16	100	21	0.783	6	0.713	6	0.755
	256	200	6	0.733	31	0.785	6	0.760
	1024	**300**	1	0.717	**6**	**0.789**	16	0.753

found that ResNet-152 without fusion augmentation with 1024 latent dimensions achieved the highest accuracy. However, when we added fusion augmentation, its accuracy did not significantly increase compared to the ResNet-50 with 16 latent dimensions. This might happen because the high dimension created more variation in the augmented latent. As a result, when the latent points were

Table 6. Accuracy of FADMAKA on holdout set

Base model	**Optimizer (lr: 0.001)**	**Loss function**	**Max epochs**	**Augmented latent**	**Latent dimension**	**k-NN**	**Inf. time (ms)**	**Acc**
ResNet-50	AdamW	Triplet Pearson	100	–	16	36	5.59	0.772
ResNet-50	AdamW	Triplet Pearson	100	Rotation	16	36	11.60	0.857
ResNet-50	AdamW	Triplet Pearson	100	Rotation, flip	16	36	16.06	0.888
ResNet-50	**AdamW**	**Triplet Pearson**	**100**	**Rotation, flip, shift**	**16**	**36**	**20.96**	**0.920**
ResNet-152	AdamW	Triplet Euclidean	300	–	1024	6	31.45	0.789
ResNet-152	AdamW	Triplet Euclidean	300	Rotation	1024	6	44.90	0.808
ResNet-152	AdamW	Triplet Euclidean	300	Rotation, flip	1024	6	58	0.842
ResNet-152	AdamW	Triplet Euclidean	300	Rotation, flip, shift	1024	6	70.35	0.878

Table 7. Accuracy of existing researches on holdout set

Approach	**Training augmentation**	**Batch size**	**Optimizer**	**Max epochs**	**Inf. time (ms)**	**Acc**
ANN [6]	-	64	SGD, lr: 0.001	100	0.50	0.313
RepVGG [7]	Flip, blur, saturation, contrast	32	Adam, lr: 0.0001	100	13.40	0.706
RepVGG_ECA [7]	Flip, blur, saturation, contrast	32	Adam, lr: 0.0001	100	25.95	0.716
ResNet-50 [8]	**Rotation, flip, shift, shear, zoom**	**32**	**Adam, lr: 0.001**	**100**	**5.62**	**0.811**
VGG16 [9]	Rotation, flip, shear, skew, contrast	64	Adam, lr: 0.001	100	308.82	0.804
Simple CNN [9]	Rotation, flip, shear, skew, contrast	64	Adam, lr: 0.001	100	19.15	0.722

Table 5. Accuracy of baseline model on holdout set

Base model	**Optimizer (lr: 0.001)**	**Max epochs**	**Inf. time (ms)**	**Acc**
ResNet-50	**SGD**	**300**	**3.51**	**0.752**
ResNet-50	AdamW	300	3.50	0.738
ResNet-152	SGD	300	12.04	0.743
ResNet-152	AdamW	300	11.96	0.745

averaged, they may not have been as effective as in the low dimension because they failed to capture the general features due to sparsity. We can see from Table 5 that the best accuracy of our baseline deep learning classification was 0.752 on ResNet-50 with SGD optimizer, which takes 3.51 ms to classify one image. SGD optimizer performed better than AdamW in the softmax classifier. Recent studies suggest that adaptive optimizer often leads to worse generalization performance than SGD for specific tasks such as image classification and language modeling despite their faster training speed [28]. It is because adaptive optimizers may have unstable and extreme learning rates that prevent them from converging to an optimal solution [28]. One more empirical explanation is that SGD is more locally unstable than adaptive optimizer at sharp minima so that SGD can better escape to flat minima, which is flat minima often generalize better than sharp minima [29]. However, the effectiveness of these optimizers may vary depending on the task and dataset used.

4.4 Comparison with the existing researches

We also reproduced state-of-the-art methods from the literature: [6, 7, 8], and [9] using our dataset which the results show in Table 7. All methods used 224x224 RGB images, except for [6], which used 50x50 images. We compared our proposed model with the method by Lu et al. [6], which used ANN with one hidden layer and 90 nodes. Their method suffered from severe underfitting on our dataset, resulting in low accuracy. This could be due to several reasons, such as the simplicity of their network architecture (since our dataset has six classes instead of two), the sensitivity of their feature extraction methods to noise information, or the difference in image quality between our dataset and theirs. Our FADMAKA model improved upon their method in several aspects. First, we used a CNN layer to automatically learn robust features from the images without manual feature engineering. Second, we used an advanced architecture model with fine-tuning technique on pre-trained weights. Third, we used data augmentation to increase the diversity and size of our training set. The method by Lu et al. [6] had a fast inference time because of its simple architecture but the lowest accuracy among others. Ni et al. [7] used RepVGG, a deep and complex architecture and applied more data augmentation methods but still had lower accuracy than our baseline model. Their model also took a long time to infer, which means a high computational cost. Ni et al. [7] also tried RepVGG_ECA, which added an ECA layer after each block and after the head layer to enhance RepVGG. The ECA layer was a new technique proposed by Ni et al. [7] to improve the network's feature representation ability. The ECA method improved the accuracy, but it also made the inference slower. We think that their proposed method was overfitted on our dataset. Perhaps their approach could do better on our dataset by tweaking some hyperparameters for regularization to prevent overfitting.

Malathi et al. [8] used a similar approach to our baseline classifier, which used a ResNet-50 model with pre-trained weights from ImageNet. However, they used a higher image resolution of 224x224, more data augmentation techniques, and a larger batch size than us. They also added a fully connected layer with dropout after the base model and weight decay 0.001 term to the optimizer to prevent overfitting. They did not report the details of the additional layer, so we used one layer with 2048 units and a dropout rate of 0.3. The final layer had six nodes for the classification task. Their model had the highest accuracy among the state-of-the-art models we reproduced on our dataset, but it was still lower than our proposed FADMAKA model. However, this approach is better than our baseline classifier. Our baseline used the same base model with a lower image resolution 64x64, without the additional fully connected layer and dropout. Rahman et al. [9] used three techniques in their paper: without transfer learning, with transfer learning, and with fine-tuning. They said that fine-tuning had the highest accuracy. However, when we reproduced it on VGG16 with fine-tuning from ImageNet weights on all layers, the model was still underfitting even with 100 epochs with the hyperparameters they suggested. Then we tried transfer learning by freezing the VGG16 layers and only training the additional fully connected layers. We noticed the accuracy was good, but the inference computation was slow compared to other methods. The accuracy was better than our baseline could be because of the higher image resolution and more diverse data augmentation techniques. However, our FADMAKA model had higher accuracy, even with a smaller image resolution and inference time. Rahman et al. [9] also proposed Simple CNN with the same hyperparameters and augmentation settings, but the accuracy was not that high compared to our baseline classifier models.

Overall, the FADMAKA approach had high accuracy but required more computational time than the average of state-of-the-art methods, although it was still faster than RepVGG_ECA and VGG16. The computation increased because of the additional augmentation process and the averaging of the latent representation. Also, k-NN classification required high computational time because it had to calculate the similarity between the input latent and all the latents in the database. We expected that we could increase the accuracy of our model by using the following methods: enhancing the image resolution to preserve more details, applying more varied data augmentation techniques during training to prevent overfitting, adding some additional fully connected layer after the base model to extract more features, and adding dropout to regularize the model. This could be seen from the pattern of the state-of-the-art model [8], which also used ResNet-50 as the base model and achieved an accuracy of 0.811, while our baseline classifier model only got 0.752 even with longer epochs.

5. Conclusion

We propose FADMAKA, a novel paddy pest and disease identification method using DML and k-NN. Our method extracts latent representations from input images and retrieves and labels similar images from the latent space. We evaluated our method on a dataset of 600 images of paddy plants with 6 classes of pests and diseases. Our method outperforms several baselines and state-of-the-art methods that used a softmax classifier, achieving maximum accuracy of 0.920. Moreover, our method can improve accuracy by up to 19.2% by augmenting input images during inference and fusing the latent space. We deploy our model on serverless cloud computing so that users can access the platform through their online devices. Our work contributes to computer vision and agriculture by providing a novel and effective paddy pest and disease identification solution. However, our work also has limitations and challenges that need to be addressed in future research, such as model complexity, scalability, computational time, and the addition of new domains.

Nomenclature

- $\mathrm{a, b, c, d}$: the parameters for contrast stretching
- α: margin parameter for Triplet
- β_1, β_2: exponential decay rates
- C: image channel where $C \in \{R, G, B\}$
- $d(\cdot)$: distance metric
- $f(\cdot)$: DML model
- γ: angle of rotation
- K: the number of classes
- k: k nearest neighbor
- $\mathcal{L}_T$: the Triplet margin loss function
- $\mathcal{L}_X$: categorical cross-entropy loss function
- m: radius of the square blur filter
- m_{t+1}: first-moment of the gradient
- N: the number of samples in a dataset
- n: the number of images in each set
- p, q: the pixel coordinates in each channel
- s: predicted label by DL
- $T(\cdot)$: generic transformation function

- v_{t+1}: second-moment of the gradient
- $\widehat{w}, \hat{b}$: final weights and biases for DML
- $\widehat{W}, \hat{B}$: final weights and biases for DL
- x: image of size $HxWxC$
- x': transformed image of size $HxWxC$
- X: anchor input image for Triplet
- X^n: negative input image for Triplet
- X^p: positive input image for Triplet
- y: image label
- z or z': latent representation of size h
- $\bar{z}$ or $\bar{z}'$: average of the latent representation
- η: learning rate
- θ_t: weights and biases at t
- λ: weight decay coefficient
- ϵ: small constant

Conflicts of interest

The authors declare no conflict of interest.

Author contributions

Conceptualization, H.D., M.Y., and M.Z.S.H.; Methodology, H.D.; Software, H.D.; Formal analysis, H.D., M.Y., and M.Z.S.H.; Supervision, M.Y. and M.Z.S.H.; Funding acquisition, M.Y. and M.Z.S.H.

Acknowledgments

This research was supported by Direktorat Jenderal Pendidikan Vokasi through Penelitian Produk Vokasi (P2V) Scheme 2023.

References

[1] B. Wang, "Identification of Crop Diseases and Insect Pests Based on Deep Learning", *Scientific Programming*, Vol. 2022, pp. 1–10, Jan. 2022.

[2] J. Liu and X. Wang, "Plant diseases and pests detection based on deep learning: a review", *Plant Methods*, Vol. 17, No. 1, 2021,

[3] J. Yu, X. Wang, X. Chen, and J. Guo, "Automatic Premature Ventricular Contraction Detection Using Deep Metric Learning and KNN", *Biosensors*, Vol. 11, No. 3, 2021.

[4] M. Kaya and H. S. Bilge, "Deep Metric Learning: A Survey", *Symmetry*, No. 11, 2019.

[5] T. Endo and M. Matsumoto, "Aurora Image Classification with Deep Metric Learning", *Sensors*, Vol. 22, No. 17, 2022.

[6] Y. Lu, Z. Li, X. Zhao, S. Lv, X. Wang, K. Wang, and H. Ni, "Recognition of Rice Sheath Blight Based on a Backpropagation Neural Network", *Electronics*, Vol. 10, No. 23, 2021.

[7] H. Ni, Z. Shi, S. Karungaru, S. Lv, X. Li, X. Wang, and J. Zhang, "Classification of Typical Pests and Diseases of Rice Based on the ECA Attention Mechanism", *Agriculture*, Vol. 13, No. 5, 2023.

[8] V. Malathi and M. P. Gopinath, "Classification of pest detection in paddy crop based on transfer learning approach", *Acta Agriculturae Scandinavica, Section B — Soil & Plant Science*, Taylor & Francis, Vol. 71, No. 7, pp. 552–559, 2021.

[9] C. R. Rahman, P. S. Arko, M. E. Ali, M. A. I. Khan, S. H. Apon, F. Nowrin, and A. Wasif, "Identification and recognition of rice diseases and pests using convolutional neural networks", *Biosystems Engineering*, Vol. 194. pp. 112–120, 2020.

[10] X. Wu, C. Zhan, Y. K. Lai, M. M. Cheng, and J. Yang, "IP102: A Large-Scale Benchmark Dataset for Insect Pest Recognition", In: *Proc. of 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 8779–8788, 2019,

[11] H. B. Prajapati, J. P. Shah, and V. K. Dabhi, "Detection and classification of rice plant diseases", *Intelligent Decision Technologies*, IOS Press, Vol. 11, pp. 357–373, 2017.

[12] J. Chen, D. Zhang, A. Zeb, and Y. A. Nanehkaran, "Identification of rice plant diseases using lightweight attention networks", *Expert Systems with Applications*, Vol. 169, p. 114514, 2021.

[13] C. Shorten and T. M. Khoshgoftaar, "A survey on Image Data Augmentation for Deep Learning", *Journal of Big Data*, Vol. 6, No. 1, p. 60, 2019.

[14] Erwin, "Improving Retinal Image Quality Using the Contrast Stretching, Histogram Equalization, and CLAHE Methods with Median Filters", *International Journal of Image, Graphics and Signal Processing*, Vol. 12, No. 2, pp. 30–41, Apr. 2020.

[15] C. Janiesch, P. Zschech, and K. Heinrich, "Machine learning and deep learning", *Electronic Markets*, Vol. 31, No. 3, pp. 685–695, 2021.

[16] K. He, X. Zhang, S. Ren, and J. Sun, "Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification", In: *Proc. of 2015 IEEE International Conference on Computer Vision (ICCV)*, pp. 1026–1034, 2015,

[17] W. Gu, A. Tandon, Y. Y. Ahn, and F. Radicchi, "Principled approach to the selection of the embedding dimension of networks", *Nature Communications*, Vol. 12, No. 1, p. 3772, 2021.

[18] Y. Huo, P. Puspitaningayu, N. Funabiki, K. Hamazaki, M. Kuribayashi, Y. Zhao, and K.

Kojima, "Three Diverse Applications of General-Purpose Parameter Optimization Algorithm", *Algorithms*, Vol. 16, No. 1, 2023.

[19] K. Park, J. S. Hong, and W. Kim, "A Methodology Combining Cosine Similarity with Classifier for Text Classification", *Applied Artificial Intelligence*, Taylor & Francis, Vol. 34, No. 5, pp. 396–411, Apr. 2020.

[20] G. Mao, "On high-dimensional tests for mutual independence based on Pearson's correlation coefficient", *Communications in Statistics - Theory and Methods*, Taylor & Francis, Vol. 49, No. 14, pp. 3572–3584, Jul. 2020.

[21] M. Yuliana, Wirawan, and Suwadi, "An Efficient Key Generation for the Internet of Things Based Synchronized Quantization", *Sensors*, Vol. 19, No. 12, 2019.

[22] I. Loshchilov and F. Hutter, "Decoupled Weight Decay Regularization", In: *Proc. of 7th International Conference on Learning Representations, ICLR 2019, New Orleans*, LA, USA, May 6-9, 2019.

[23] H. Darmawan, M. Yuliana, and M. Z. S. Hadi, "GRU and XGBoost Performance with Hyperparameter Tuning Using GridSearchCV and Bayesian Optimization on an IoT-Based Weather Prediction System", *International Journal on Advanced Science, Engineering and Information Technology, INSIGHT - Indonesian Society for Knowledge and Human Development*, Vol. 13, No. 3, pp. 851–862, 2023.

[24] R. Gove, L. Cadalzo, N. Leiby, J. M. Singer, and A. Zaitzeff, "New guidance for using t-SNE: Alternative defaults, hyperparameter selection automation, and comparative evaluation", *Visual Informatics*, Vol. 6, No. 2. pp. 87–97, 2022.

[25] H. Darmawan, M. Yuliana, and M. Z. S. Hadi, "Realtime Weather Prediction System Using GRU with Daily Surface Observation Data from IoT Sensors", In: *Proc. of 2022 International Electronics Symposium (IES)*, pp. 221–226, 2022.

[26] Y. Kiyoki, S. Sasaki, and A. R. Barakbah, "AI-Sensing Functions with SPA-Based 5D World Map System for Ocean Plastic Garbage Detection and Reduction", *Information Modelling and Knowledge Bases XXXIV*, Vol. 364, 2023.

[27] S. Luo, Y. Shi, L. K. Chin, Y. Zhang, B. Wen, Y. Sun, B. T. T. Nguyen, G. Chierchia, H. Talbot, T. Bourouina, X. Jiang, and A. Q. Liu, "Rare bioparticle detection: Via deep metric learning", *RSC Advances, Royal Society of Chemistry*, Vol. 11, No. 29, pp. 17603–17610, 2021.

[28] M. Tang, Z. Huang, Y. Yuan, C. Wang, and Y. Peng, "A Bounded Scheduling Method for Adaptive Gradient Methods", *Applied Sciences*, 2019.

[29] P. Zhou, J. Feng, C. Ma, C. Xiong, S. Hoi, and E. Weinan, "Towards Theoretically Understanding Why SGD Generalizes Better than ADAM in Deep Learning", In: *Proc. of the 34th International Conference on Neural Information Processing Systems*, 2020.