https://doi.org/10.17221/65/2023-RAE

The effect of parameter adjustment in sago palm classification-based convolutional neural network (CNN) model

Sri Murniani Angelina Letsoin[1]*, *David Herák*[2]

[1]*Faculty of Engineering, University of Musamus, South Papua, Merauke, Indonesia*
[2]*Department of Mechanical Engineering, Faculty of Engineering, Czech University of Life Sciences Prague, Prague, Czech Republic*

**Corresponding author: letsoin@unmus.ac.id*

Citation: Letsoin S.M.A., Herák D. (2024): The effect of parameter adjustment in sago palm classification-based convolutional neural network (CNN) model. Res. Agr. Eng, 70: 123–133.

Abstract: In our study location, Merauke Regency, the easternmost city in Indonesia, the sago palm is associated with different types of ecosystems and other non-sago vegetation. During the harvesting season, the white flowers blossoming between the leaves on the tops of palm trees may be distinguished manually. Four classes were determined to address the visual inspections involving different parameters that were examined through the metric evaluation and then analysed statistically. The computed Kruskal-Wallis test found that the parameters vary in each network with a *P*-value of 0.00341, with at least one class being higher than the others, i.e., non-sago with a *P*-value of 0.044 with respect to precision, recall, and F1-score. Thus, the general linear model (GLM) was tested specifically in trained Network-15 and Network-17, which have similar parameters except for the batch size. It indicated the two networks' differences based on their prediction results, classes, and actual images. Accordingly, a combination of learning rate (Lr) and batch size improved the reliability of the training and classification task.

Keywords: deep learning; detection; model; parameter; transfer learning

The sago palm (*Metroxylon Sagu* Rottb) is one of the environmentally-conscious palm tree species that may grow wildly in the forest, primarily in Southeast Asian countries and also in Papua New Guinea (PNG). Several earlier studies revealed sago's food and non-food industry features (Karim et al. 2008; Ehara et al. 2018). Visible parts of the tree can be utilised, such as the bark, leaves, starch, and sago waste. The bark can be used for traditional flooring, walls, or craft paper. Further, the leaves are used for roofing, and the waste is for animal feed or compost. Sago's palms contain a lot of starch utilised for traditional food, cakes, and beverage industries; thus, as a vital resource for the agricultural industry, biopesticides, and the bioethanol industry (Mofu and Abbas 2015; Metaragakusuma et al. 2016; Jonatan et al. 2017; Amin et al. 2019). In our selected study location, Merauke Regency, the easternmost city in Indonesia, sago palm trees typically develop in wild stands with a height of 7–15 m with different types of ecosystems, such as peatland areas or swampy forests. Figure 1 presents the sago palm in the field work, which may live with other vegetation (Figure 1A) or non-sago palm vegetation (Figure 1B).

Supported by the Internal Grant Agency (IGA) of the Faculty of Engineering 2021 (Gant No.: 31130/1312/3105) Smart sago palm detection using Internet of Things (IoT) and Unmanned Aerial Vehicle (UAV) Imagery.

© The authors. This work is licensed under a Creative Commons Attribution-NonCommercial 4.0 International (CC BY-NC 4.0).

https://doi.org/10.17221/65/2023-RAE

Figure 1. Sago palm in Tambat district at Merauke Regency, Papua province of Indonesia: (A) sago forest with non-sago vegetation, and (B) non-sago (other vegetation)

The harvesting season may be distinguished manually by the white flowers blossoming between the leaves on the tops of palm trees. Several studies were focused on investigating the sago palm's condition, for instance, by extracting satellite imageries combined with suitable methods, such as support vector machine (SVM), object-based image analysis (OBIA), and image processing (Hidayat et al. 2018). Nevertheless, the study pointed out that morphology and similarity with other palms could affect the classification result. Moreover, the maximum likelihood as a classifier in sago palm distribution from the satellite was studied in the Philippines (Santillan and Makinano-Santillan 2016). Nevertheless, the previously related works were not applicable to our fieldwork settings. The most specific problem is the challenge of harvesting time prediction that is practically defined through the morphology of sago. Nonetheless, due to the wild stand of the sago, the height of the sago surrounded by swampy areas could influence the result. Another sago palm detection model uses the convolutional neural network (CNN) architecture, namely Alex Net, Xception, ResNet and CraunNet, to identify the maturity of sago obtained from unmanned aerial vehicle (UAV) images (Wahed et al. 2022). This related research addressed identifying the maturity of sago palms through their canopies rather than harvest time prediction. CNN, frequently referred to as a ConvNet, is a type of feed-forward neural network commonly used for analysing visual objects. Each image in CNN is represented as an array of pixel values. The CNN has many layers, generally consisting of a convolution layer, rectified linear unit (reLU) layer, pooling layer, flatten layer and fully connected layer (Kneusel 2021). Today, deep learning based on CNN study is enormously used in image classification tasks or object detection. In general, deep learning in object detection is divided into three categories: (*i*) CNN: A CNN deforms learned features according to the initial data and applies 2-D convolutional layers, ideally designed to process 2-D data, for example, images. (*ii*) Segmentation, a deep learning method that associates a label or category with every pixel in an image; and (*iii*) object detection method, which refers to using deep learning to provide a specific location of an object in an image (Zheng et al. 2021). According to a study review by (Yasir et al. 2023), the most prevalent deep learning method developed to cope with remote sensing image processing, for example, satellite data or data from an unmanned aerial vehicle (UAV), is CNN. The feature knowledge in CNN can be transferred from one domain to another using the transfer learning technique. Transfer learning (TL) can be investigated as a process of refining the target prediction function f_t (.) based on D^s and T^s , with $D^s \neq D^T$ or $T^s \neq T^T$ through knowledge transfer. Figure 2 shows the concept of transfer learning as follows.

The two components of a domain are a feature space X and a marginal distribution of probabilities P(X), where $X = \{x_1, x_2..., x_{n-1}, x_n\}$, n represents number of feature vectors in X. Similar to D, T contains two components, i.e., label space Y and a predictive function. Pairs of feature vectors and labels are used to train the predictive function f_t (.), a domain $D = \{X, P(X)\}$ and a task $T = \{Y, f(.)\}$ accordingly. Henceforth, the source domain can be described as $D^S = \{X, P_S(X)\}$ with an associated source task $T^S = \{Y, f_s(\hat{A})\}$, equally the $D^T = \{X, P_t(X)\}$ with

https://doi.org/10.17221/65/2023-RAE

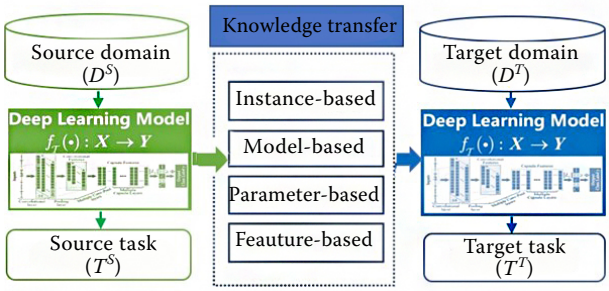

Figure 2. The concept of transfer learning with modification
Source: Li et al. 2022

a related source task $T^T = \{Y, f_t(\hat{A})\}$. In this study, transfer learning based on the CNN model was performed to distinguish four visible morphologies of sago and one class of other vegetation in the sago forest area. Different parameters were implemented in training five networks, namely epoch, batch size and learning rate, in training five networks. Moreover, the results were analysed statistically to investigate whether or not the various parameters affect the prediction results. The research dataset in this study is self-made from UAV images and ground photographs, including sago leaves, trunks, flowers and non-sago images, which aims to differentiate the visual morphology of sago from other vegetation by utilising the transfer learning technique-based CNN model.

MATERIAL AND METHODS

The images were captured from the ground and a UAV with a certain parameter. The UAV was integrated with the mission flight planner, and it flew over a sago area of 74.600 m^2 in Merauke Regency, Papua Province of Indonesia, (location: 137°38'52.9692' E–141°0'13.3233' E, 6°27'50.1456' S–9°10'1.225' S, collecting a total of 661 images in two flights. The Autel drone (EVO II Pro, Autel Robotics, China) was used to fly from 9:00 a.m. to 11:30 a.m., using a double grid with 70 and 80% of front overlap and 70 and 60% of side overlap at an altitude of 60.3 m. Afterwards, all the images were transferred to computer storage for preprocessing by using pix4dmapper (version 4.8). In this stage, all data were divided into three types, namely data for testing (103 images), training (441 images) and validation (132 images). Image segmentation, as well as a cropping process, were required to designate the region of interest (ROI). Thus, the images were labelled into four classes, namely non-sago (110 images), sago flowers (110 images), sago leaves (111 images) and sago trunks (110 images) through makesense.ai (www.makesense.ai). Finally, all the classes of images are uploaded to the MATLAB software (MatlabR2022a) directory for further processing.

In this experiment, a validation frequency of 4 was used, corresponding to each category class, *i.e.*, non-sago (NS), sago flowers (SF), sago leaves (SL) and sago trunks (ST), respectively. Other parameters were set up: momentum = 0.9 and learning rate (lr) bias coefficient = 10. Further, the weight lr factor was 1, and the bias lr factor was 10. The network layers used in each trained Network (Table 1) were developed, referring to five convolutional layers. Three fully connected layers within the activation layer, namely ReLU layers, were added after each convolution layer; further, two cross channels normalisation after ReLU layers, pooling layer with stride [2 2], and padding [0 0 0 0]. The rest of the layers consisted of 50% dropout layers, fully connected to four layers, SoftMax for probability, and cross-entropy loss function for the classification output. The total layers were 25 layers and 68 layers arranged in MATLAB environments and transferred from AlexNet and SqueezeNet architecture as a base domain; thus, all networks were trained according to each parameter, as displayed in Table 1. Network-10 and Network-15 were designed with 68 layers, while the rest networks with 25 layers. The layers designed are as follows:

Batch normalisation layers. To normalise the activations of each channel, the layer initially decreases the mini-batch mean and divides it by the mini-batch standard deviation. Following that, the layer adjusts the input by a learnable offset β and scales it by a learnable scale factor γ. β and γ are learnable parameters that are changed throughout training the networks. use batch normalisation lay-

Table 1. Parameter used in experiment

Network name	Training set up		
	epoch	learning rate	minimum batch size
Trained Network-10	10	0.001	64
Trained Network-15	10	0.0001	10
Trained Network-17	10	0.0001	64
Trained Network-19	15	0.0001	64
Trained Network-22	8	0.0001	10

https://doi.org/10.17221/65/2023-RAE

ers between convolutional layers and nonlinearities, such as ReLU Layers, to accelerate the convolutional neural network training process and reduce network initialisation risk. To normalise the elements x_i in a min batch $W=\{x_{(1,\ldots)}x_m\}$, first, the batch normalisation operation determines the mean μ_i:

$$\mu_W = \frac{1}{m}\sum_{i=1}^{m}(x_i) \quad (1)$$

Variance σ_W^2 over the independent dimensions of spatial, time, and dimension observation for each channel:

$$\sigma_W^2 = \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_W)^2 \quad (2)$$

It then calculates the normalised activations as the following equation:

$$\hat{x}_l = \frac{x_i - \mu_w}{\sqrt{\sigma_w^2 + \in}} \quad (3)$$

If parameters to be learned are described by α, β, then to accommodate the potential data inputs with a zero mean and a variance of the unit are not ideal during the processes that follow the batch normalising. The batch normalising operation scales and shifts the activations employing a conversion as shown in the equation:

$$y_i = a\hat{x}_l + \beta \quad (4)$$

Activation layers. Generally, there are three activation functions, i.e., sigmoid, Tanh activation, and ReLU. Sigmoid and Tanh activation is usually used in Recurrent neural networks, while ReLU layer is preferred in multilayer perceptron and CNN. ReLU Layer is one activation function that performs the element-wise operation; for example, it adjusts all negative pixels to zero. Otherwise, it returns the value as a rectified feature map. The relationship in this layer can be formulated as seen below:

$$RELU(x) = \max(0, x) \quad (5)$$

In multiclass classification problems, particularly the prediction of the probability of each instance, the softmax layer as an output layer is used:

$$P(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{n} e^{x_j}} \quad (6)$$

where: x – the values from layers in the output of the i-dimension; n – the size of the dimension referring to the size of classes. In a classification task, the sum of the probabilities equals 1.

Pooling layers. The pooling layer is an operation of down-sampling to downsize the dimensions of the rectified feature map. Filters and strides are also used at the pooling layer to identify various features such as corners, edges, leaves, etc. Since the pooling layer is used to decrease a certain number of parameters to train, the computation requirements are also reduced. There are two kinds of pooling layers, namely, the average pooling and the max pooling layer. The max pooling layer is most commonly employed to select the largest value in each filter region. Two factors make the pooling layer of paramount importance to CNN. First, without diminution, the computation would crash when convolutional layers capture duplicate information. Second, the duplicate features would degrade the redundant information's ability to describe features. As a result, implementing a pooling layer is necessary for reducing the dimensions of features.

Fully connected layer. Fully connected layers are often positioned near the output layers. In the image classification task, this layer acts as a classifier and the final output layer. It receives input from the pooling or convolution layer's final output before converting it to a single vector through a flattened patch. The flatten patch function is utilised for transforming all of the dimensional arrays produced by the pooling feature map into a one-dimensional linear vector. Thus, flattening the matrices provides the input to the fully interconnected layer, which classifies the object.

Dropout layers. Like other regularization techniques, the dropout is preferable for cases when the training data is insufficient and the model is prone to overfitting. During training, this layer sets a number of its output characteristics to zero by deleting the forthcoming node in the layer as defined in their dropout rate. It employs the majority of layers, encompassing convolutional, densely fully connected, and recurring layers such as the long short-term memory (LSTM) structure.

A function of loss. A good model is defined by a smaller loss function. Otherwise, the model's parameters need to be adjusted to reduce the loss. The loss during a single training process is called the loss function, whereas the average loss across all training data sets is referred to as the cost function. Loss function in deep learning network can be estimated depending on its task, for instance, regression task

https://doi.org/10.17221/65/2023-RAE

by applying the mean squared error (MSE) or the mean absolute error (MAE). A focal loss is used in the object detection task, while binary cross-entropy or category cross-entropy is used in classification. For multiclass classification, the category cross-entropy is employed within the following equation:

$$Loss = -\sum_{j=0}^{k} y_j \log(\hat{y}_J) \quad (7)$$

where: k – the number of data classes; j = 1, 2, … k.

$$COST = \frac{1}{2}\sum_{i=1}^{n}\sum_{j}^{k}\left[y_{ij}\log\hat{y}_{ij}\right] \quad (8)$$

where: k – classes; y – the actual value; $\hat{y}$ – the prediction.

Optimisation function. The optimiser is used to update the parameter in the network, and at the same time to minimise the loss function. There are some optimisers mostly used in deep learning models, such as the Adam optimiser or Stochastic gradient descent with momentum (SGDM). The Stochastic gradient descent (SGD) can oscillate throughout the path, which affects achieving the best outcome; one technique for preventing this oscillation is to incorporate the momentum parameter to minimise it. The SGDM optimises the process with this equation:

$$\theta_{l+1} = \theta - \alpha_l - \nabla E(\theta_l) + \gamma(\theta_l - \theta_l - 1) \quad (9)$$

l is the number of iterations; $\alpha > 0$ represents the Lr; θ denotes the vector of the parameter; $\nabla E(\theta)$ is the gradient of the loss function as determined by all of the training data sets. The typical gradient descent algorithms operate the full data set simultaneously. γ indicates the amount the previous gradient step contributed to the current iteration, which is expressed as momentum value. The momentum value is defined as a scalar from 0 to 1. In this study, we use SGDM with a momentum value of 0.9. After data were trained with parameters defined previously, the testing stage of 103 images was performed. The test images were cropped to 227 × 227 × 3 in size; the prediction results were then documented in a spreadsheet file. At the evaluation stage, the precision, sensitivity and F1-score were investigated, as displayed in the Table 2.

True positive indicates a positive sample which was correctly predicted, and false positive a negative sample that was incorrectly forecasted. While false negative indicates a positive case that was forecasted incorrectly, true negative indicates a negative sample that was successfully predicted.

Sample data set used in this study are displayed in the Figure 3.

Additionally, the Kruskal-Wallis H tests were performed to investigate whether they produce the same median weight. The non-parametric test applies ranks to the H_o if the parameter weight for sago palm detection in all networks are the same, while the H_a represents at least one network is differ from the other. The H test is done with the following formula (Da Silva et al. 2022; Akshit Rajesh Tayade and Safari-Katesari 2023):

$$H = \left[\frac{12}{n_T(n_T+1)}\sum_{i=1}^{k}\frac{R_i^2}{n_i}\right] - 3(n_T+1) \quad (10)$$

where: k – the number of populations; n_i – how many observations there are in sample i; $n_T = \Sigma_{(i=1)}{}^{k}$ – the sum of all observations across all samples; R_i – the total rank scores for the sample I.

RESULTS AND DISCUSSION

The training processes were performed for five network structures; some training progresses are displayed in Figure 4. The accuracy and loss values are distinguished by blue and orange, respectively. The training accuracy is represented by light blue coloured dots, while the training loss during the learning process is indicated by light orange dots. In addition, black coloured dots denote validation

Table 2. Metric evaluation used in this study

Metric	Formula	Criteria
F1-score	$\frac{2\times(recall\times precision)}{recall + precision}$	high scores support the model's validity
Precision	$\frac{true\ positive}{true\ positive + false\ positive}$	determines the model's capability to forecast a positive label.
Sensitivity (recall)	$\frac{true\ positive}{true\ positive + false\ negative}$	defines the model's capability to accurately find instances of particular classes.

https://doi.org/10.17221/65/2023-RAE

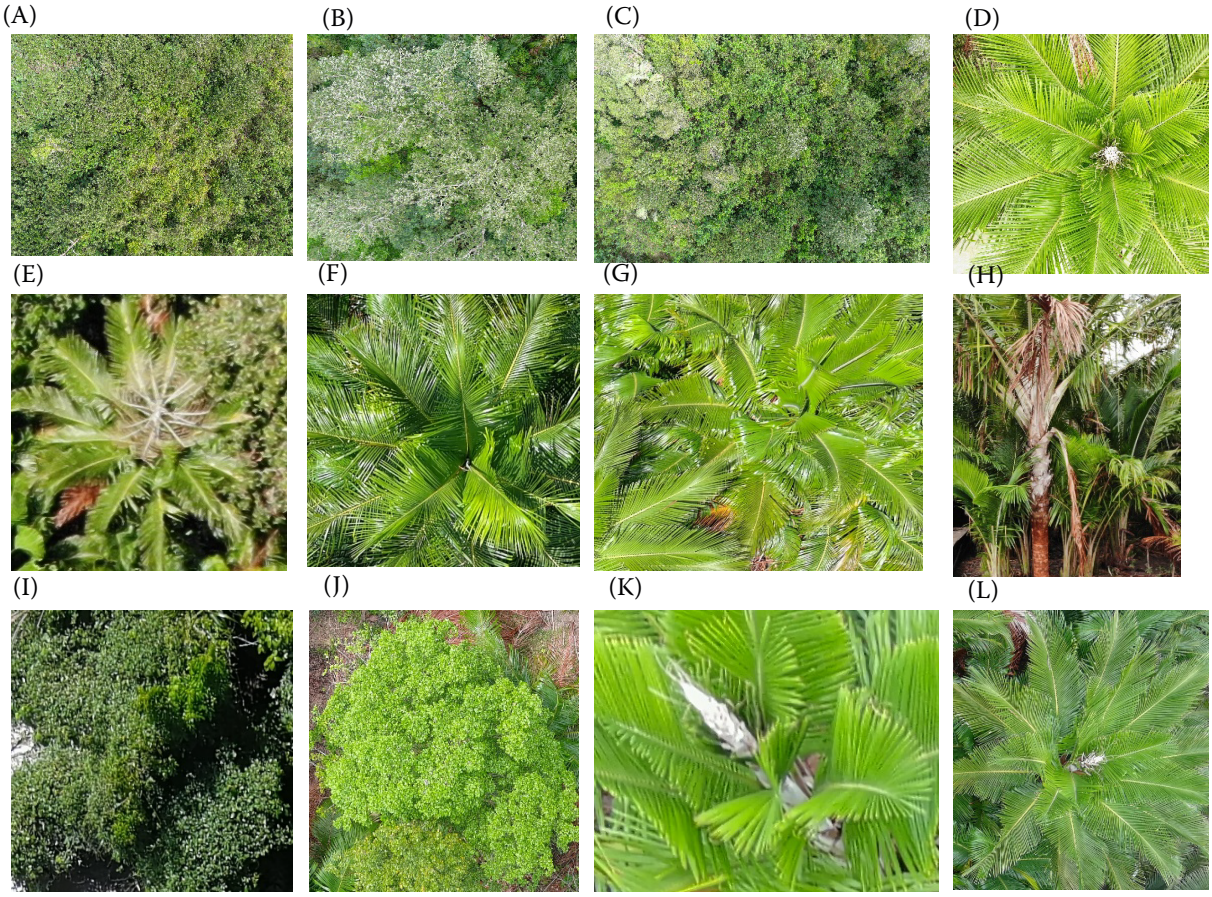

Figure 3. Sample dataset provided in this study: (A, B and C) Non sago, (D, E) sago flowers, (F, G) sago leaves, (H) sago trunk, (I, J) test image: non-sago, and (K, L) test image: sago flowers

accuracy and loss, while dark blue and dark orange colours define the sleekness of both accuracy and loss values, respectively. Training loss is calculated after each batch, while validation loss is measured after each epoch.

The learning results from each trained Network are presented in Table 3.

As seen from Table 3, Network-17 achieved less differentiation between training and validation learning results compared to Network-10 or Network-15.

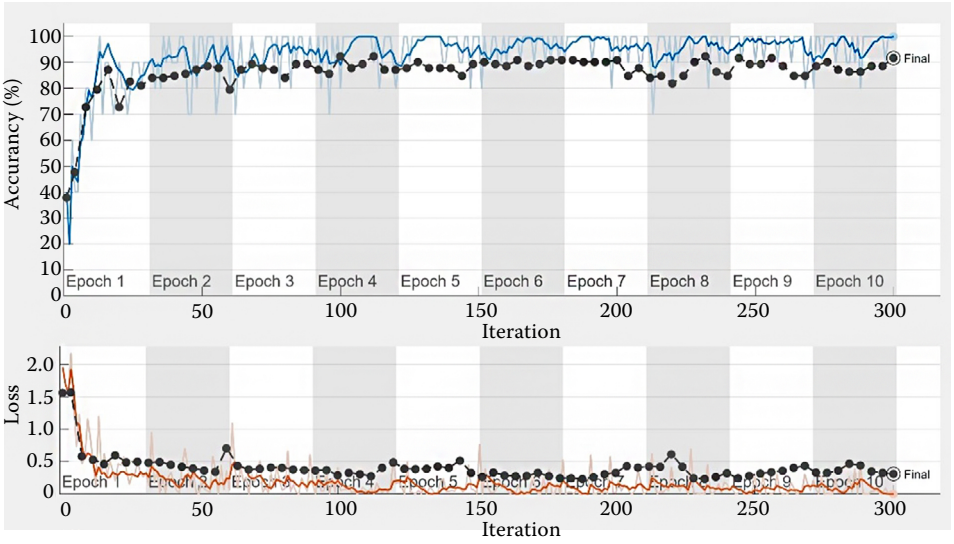

Figure 4. Sample training progress of trained Network-15

https://doi.org/10.17221/65/2023-RAE

Table 3. Learning results of five trained network

Network trained name	Training accuracy	Training loss	Validation accuracy	Validation loss	Network iteration	Elapsed time
Trained Network-10	96.87	0.1884	84.85	0.8798	40	4 min 44 sec
Trained Network-15	100.00	0.0038	91.67	0.3120	300	33 min 21 sec
Trained Network-17	89.06	0.2816	90.15	0.2766	40	3 min 44 sec
Trained Network-19	96.87	0.1601	88.66	0.4384	60	5 min 55 sec
Trained Network-22	80.00	0.3592	86.36	0.4761	240	31 min 40 sec

Training loss and validation loss in Network-19 and Network-22 differed by the range of 0.1 to 0.3, while the training and validation accuracy values differed between 6 to 8. Moreover, the number of iterations was equivalent to the elapsed time; the more networks were iterated, the more time was consumed. In the last stage, 103 test images were used to investigate the performance of each trained model in predicting and classifying the input test images. Some of the confusion matrixes are presented in Figure 5. Sago model-1 consists of 68 CNN model layers, while sago model-2 contains 25 CNN layers. For example, the precision value in trained Network-15 in detecting non-sago was about 80%, while in Network-17 and Network-22, 82.6, 85%, respectively. Accordingly, the recall or sensitivity of these three networks was about 69.6, 82.6, and 73.9%.

Furthermore, the sensitivity, precision and F1-score are measured based on values in the confusion matrix, as presented in Figure 6. The precision and the sensitivity of each class were performed better in trained Network-17, trained Network-19, and trained Network-22. Although the sensitivity (recall) in trained Network-10 was 91% for sago trunks, the precision was only about 37%. On the one hand, in trained Network-15, the precision of sago flowers was 90%, but the sensitivity was only about 56%, which means that around 56% of the network was able to detect the instances of specific classes. Network-10 was trained with parameters similar to those of Network-17, i.e., Epoch = 10, min batch size = 64. The biggest distinction between these two networks was the learning rate. The learning rate (Table 1) in Network-10 was set up faster than in trained Network-17; 0.001 and 0.0001, respectively. If the loss value changes rather than drops, the model may not be learning at all. Nevertheless, if it declines in the training set but not in the validation set (or if it declines but there is a significant difference), the model may be overfitting. To overcome this circumstance in deep learning as well as transfer learning techniques, it is necessary to merely decrease the learning rate (Lin et al. 2023; Mukoya et al. 2023), as we found in the course of our research

Passing a complete dataset through the network constitutes an epoch; thus, the total number of training samples in a single mini-batch is referred

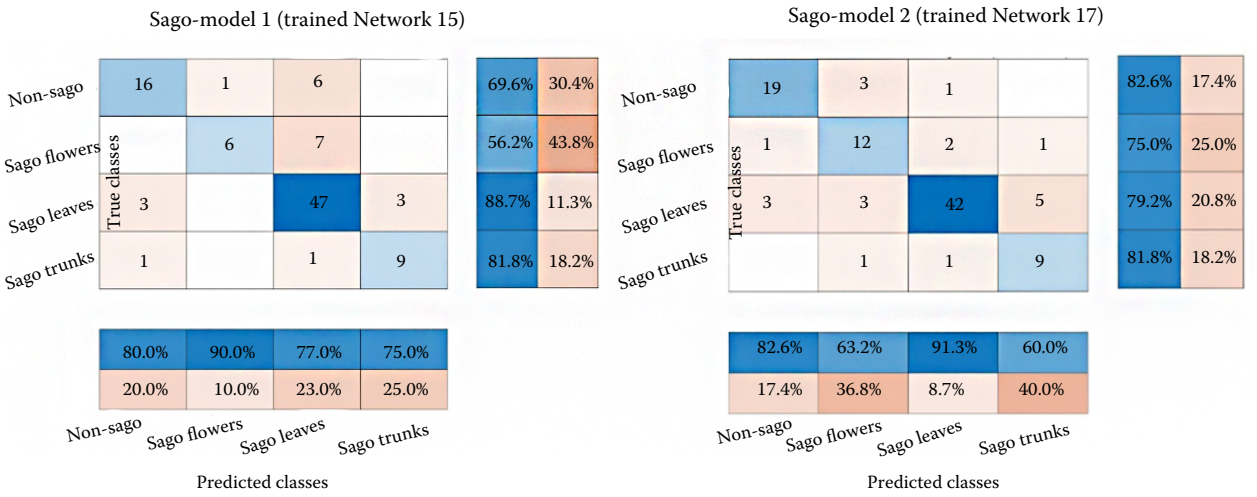

Figure 5. Sample confusion matrix results with parameters: (A) 10 epochs, learning rate was 0.0001, minimum batch size 10, and (B) 10 epochs, learning rate was 0.0001, minimum batch size 64

https://doi.org/10.17221/65/2023-RAE

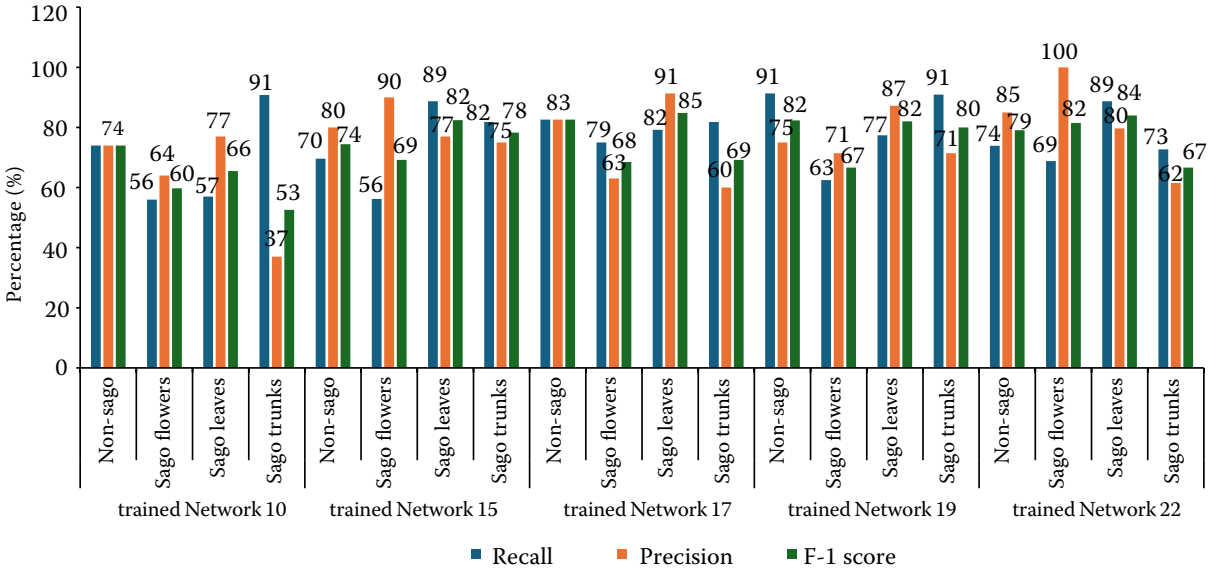

Figure 6. Metric evaluation of five trained networks

to as the batch size. Adapting a larger batch requires higher hardware processing; therefore, splitting it into mini-batch sizes is foremost. An updating of the model's weights during training is referred to as an iteration. The number of batches required to finish one epoch is equal to the number of iterations. In this experiment, we define four kinds of mini-batch size, i.e., 10, 64, combined with three groups of epochs, i.e., 8, 10, 15, and with 309 training images. For example, a trained Network-17 splits into 64 mini-batch sizes with ten epochs. Thus, 309 images divided by 64 mini-batch size turn to approximately 4.8 or around four images in one epoch. Then, it will take 40 network iterations to complete ten epochs (4 images x 10 epochs). Thus, if we set up a high lr or a fast-learning process, the model is not able to read accurately. As a result, the model fails to learn, particularly if the loss does not decrease (Kumar and Janet 2022).

Figure 7 shows the sum of ranks (R_i) produced through the Kruskal-Wallis H test process. As can be seen, the R_i of the five networks were domi-

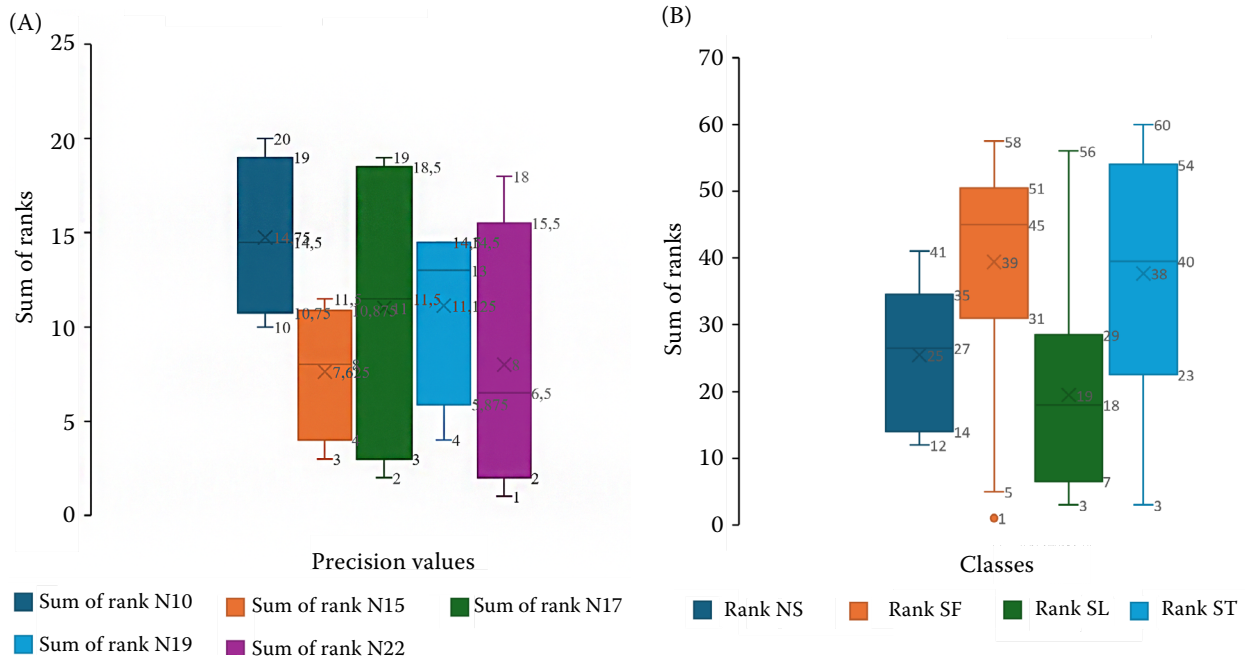

Figure 7. Sum of ranks of (A) five networks, and (B) four classes in five networks
NS – non sago; SF – sago flowers; SL – sago leaves; ST – sago trunks

https://doi.org/10.17221/65/2023-RAE

nated by trained Network-22, followed by trained Network-17, trained Network-15, and trained Network-19, while the last rank was trained Network-10.

The *H* values were calculated based on the equation, and the results for the five networks are in Table 4.

The calculation results in Table 4 were compared to the same non-parametric test Kruskal-Wallis 1-way ANOVA (*k* samples) in SPSS IBM statistics software (version 26). The overall result is shown in Table 5.

The results in Tables 4 and 5 indicated that the parameter adjustment affected the results in classification differently; at least one parameter differed from the other. Trained Network-15 and trained Network-17 were set up with the same parameters, namely learning rate (0.0001) and epoch (10); only one parameter differed between those two networks, namely batch size, i.e., 10 and 64, respectively. Table 6 shows the tests of effects on the subject from

Table 4. Kruskal-Wallis test results

Parameter	Value	Description
No. NS	15	number of observations in non-sago class
No. SF	15	number of observations in sago flowers class
No. SL	15	number of observations in sago leaves class
No. ST	15	number of observations in sago trunks class
k	4	number of groups
nt	60	total number of observations
$\frac{12}{n_T(n_T+1)}$	0.003278689	
$3(n_T+1)$	183	
$\sum_{i=1}^{k}\frac{R_i^2}{n_i}$	59.979	
H-value	13.65	*df* =3
Chi-square (χ2)	7.814727903	χ2 inverse (0.95, *df*)
P-value	0.003415348	χ2 distribution, right tail (*H*, *df*)
H-value	reject the null hypothesis	$H > \chi 2$

Table 5. Kruskal-Wallis 1-way analysis of variance (*k* samples), *N* = 15

Parameter	Test statistic	Significance level	*P*-value	H_0
Non sago	9.804	0.05	0.044	reject
Sago flowers	6.106	0.05	0.187	retain
Sago leaves	7.218	0.05	0.125	retain
Sago trunk	2.718	0.05	0.606	retain

H_0 – null hypothesis

Table 6. General linear model result of four factors

Source	Type III sum of squares	*df*	Mean square	*F*-value	Significance	Source
class * trained name * predict *actual	sphericity assumed	0.488	9	0.054	3.534	< 0.001
	greenhouse-Geisser	0.488	6.539	0.075	3.534	0.001
	Huynh-Feldt	0.488	7.500	0.065	3.534	< 0.001
	Lower-bound	0.488	3.000	0.163	3.534	0.016

df – degree of freedom

https://doi.org/10.17221/65/2023-RAE

class, trained network, prediction result and actual class using a general linear model (GLM).

The class factor in Table 6 was defined in 4 groups, i.e., non-sago, sago flowers, sago leaves and sago trunks. It can be seen that the results from the two networks differed from one class to others significantly in terms of statistics analysed. These results are in line with another earlier study that highlighted batch size as one of the influencing factors in image classification (Usmani et al. 2023) together with learning rate, as was also revealed in this study. Therefore, an adjustment between learning rate and batch size could improve the level of precision.

CONCLUSION

This study involved knowledge transfer by adjusting different parameters, namely epoch, learning rate and batch size in sago palm detection. Five networks were used to classify and predict three visible morphology classes of sago palm, *i.e.*, sago flowers, sago leaves, sago trunks and other non-vegetation or non-sago. The results showed that different parameters achieved various results. This is apparent from the metric calculation followed by the statistical analysis. Two statistical approaches were delivered to accept or reject the hypothesis, i.e., first, the Kruskal-Wallis test to analyse the differences between the five networks. As a result, the Kruskal-Wallis test found the parameters in each network are different with a *P*-value of 0.00341, while at least one class is higher than others, i.e., non-sago with a *P*-value of 0.044 with respect to precision, recall, and F1-score. Second, the GLM test specifically examined two networks, namely trained Network-15 and trained Network-17, since these two networks consisted of similar parameters, except for the batch size. The results revealed that the two networks' effects differed based on the prediction result, actual image, and classes. This research effort highlighted the importance of adjusting the parameters, such as learning rate and batch size, to achieve the expected result. The study can experiment further with different networks, various learning rates, and the batch size.

REFERENCES

Akshit Rajesh Tayade, Safari-Katesari H. (2023): A statistical analysis to develop machine learning models: Prediction of user diet type.

Amin N., Sabli N., Izhar S., Yoshida H. (2019): Sago wastes and its applications. 1841–1862.

da Silva F.G., Ramos L.P., Palm B.G., Machado R. (2022): Assessment of machine learning techniques for oil rig classification in c-band sar images. Remote Sensing, 14: 2966.

Ehara H., Toyoda Y., Johnson D.V. (2018): Sago palm: Multiple contributions to food security and sustainable livelihoods. Springer, Singapore.

Hidayat S., Matsuoka M., Baja S., Rampisela D.A. (2018): Object-based image analysis for sago palm classification: The most important features from high-resolution satellite imagery. Remote Sensing, 10: 1319.

Jonatan N.J., Ekayuliana A., Dhiputra I.M.K., Nugroho Y.S. (2017): The utilization of *Metroxylon Sago* (Rottb.) dregs for low bioethanol as fuel households needs in Papua province Indonesia. KnE Life Sciences, 3: 150.

Karim A.A., Tie A.P.L., Manan D.M.A., Zaidul I.S.M. (2008): Starch from the Sago (*Metroxylon sagu*) palm tree properties, prospects, and challenges as a new industrial source for food and other uses. Comprehensive Reviews in Food Science and Food Safety, 7: 215–228.

Kneusel R.T. (2021): Practical Deep Learning: A Python-based Introduction (1st ed.). No Starch Press, Inc., San Francisco.

Kumar S., Janet B. (2022): DTMIC: Deep transfer learning for malware image classification. Journal of Information Security and Applications, 64: 103063.

Lin K., Zhao Y., Wang L., Shi W., Cui F., Zhou T. (2023): MSWNet: A visual deep machine learning method adopting transfer learning based upon ResNet 50 for municipal solid waste sorting. Frontiers of Environmental Science & Engineering, 17: 77.

Li Z., Kristoffersen E., Li J. (2022): Deep transfer learning for failure prediction across failure types. Computers & Industrial Engineering, 172: 108521.

Metaragakusuma A.P., Katsuya O., Bai H. (2016): An overview of the traditional use of sago for sago-based food industry in Indonesia. KnE Life Sciences, 3: 119.

Mofu S.S., Abbas B. (2015): Development of sago palm research and agroindustry in University of Papua. In: Proceeding of the 12th International Sago Symposium in Tokyo, Japan. Sept 15–16, 2015. Japan Society for the Promotion Science.

Mukoya E., Rimiru R., Kimwele M., Gakii C., Mugambi G. (2023): Accelerating deep learning inference via layer truncation and transfer learning for fingerprint classification. Concurrency and Computation: Practice and Experience, 35: e7619.

Santillan J.R., Makinano-Santillan M. (2016): Recent Distribution of Sago Palms in the Philippines. In: M.J.D. Paluga (Ed.): University of the Philippines: Mindanao, Philippines, 186.

https://doi.org/10.17221/65/2023-RAE

Usmani I.A., Qadri M.T., Zia R., Alrayes F.S., Saidani O., Dashtipour K. (2023): Interactive effect of learning rate and batch size to implement transfer learning for brain tumor classification. Electronics, 12: 964.

Wahed Z., Joseph A., Zen H., Kipli K. (2022): Sago palm detection and its maturity identification based on improved convolution neural network. Pertanika Journal of Science and Technology, 30: 1219–1236.

Yasir M., Jianhua W., Shanwei L., Sheng H., Mingming X., Hossain M. (2023): Coupling of deep learning and remote sensing: A comprehensive systematic literature review. International Journal of Remote Sensing, 44: 157–193.

Zheng J., Fu H., Li W., Wu W., Yu L., Yuan S., Tao W.Y.W., Pang T.K., Kanniah K.D. (2021): Growing status observation for oil palm trees using unmanned aerial vehicle (UAV) images. ISPRS Journal of Photogrammetry and Remote Sensing, 173: 95–121.

Received: June 30, 2023
Accepted: March 22, 2024
Published online: July 16, 2024