Revue d'Intelligence Artificielle
Vol. 35, No. 5, October, 2021, pp. 375-381
Journal homepage: http://iieta.org/journals/ria

# A Precision Agricultural Application: Manggis Fruit Classification Using Hybrid Deep Learning

Putra Sumari[1*], Wan Muhammad Azimuddin Wan Ahmad[1], Faris Hadi[1], Muhammad Mazlan[1], Nur Anis Liyana[1], Rotimi-Williams Bello[1,2], Ahmad Sufril Azlan Mohamed[1], Abdullah Zawawi Talib[1]

[1] School of Computer Sciences, Universiti Sains Malaysia, Pulau Pinang 11800, Malaysia
[2] Department of Mathematics/Computer Sciences, University of Africa, Toru-Orua 561101, Bayelsa State, Nigeria

Corresponding Author Email: putras@usm.my

https://doi.org/10.18280/ria.350503

**Received:** 5 September 2021
**Accepted:** 13 October 2021

***Keywords:***
*CNN, hybrid deep learning, mangosteen fruit, Resnet, SGD, transfer learning, Xception, VGG16*

## ABSTRACT

Fruits come in different variants and subspecies. While some subspecies of fruits can be easily differentiated, others may require an expertness to differentiate them. Although farmers rely on the traditional methods to identify and classify fruit types, the methods are prone to so many challenges. Training a machine to identify and classify fruit types in place of traditional methods can ensure precision fruit classification. By taking advantage of the state-of-the-art image recognition techniques, we approach fruits classification from another perspective by proposing a high performing hybrid deep learning which could ensure precision mangosteen fruit classification. This involves a proposed optimized Convolutional Neural Network (CNN) model compared to other optimized models such as Xception, VGG16, and ResNet50 using Adam, RMSprop, Adagrad, and Stochastic Gradient Descent (SGD) optimizers on specified dense layers and filters numbers. The proposed CNN model has three types of layers that make up its model, they are: 1) the convolutional layers, 2) the pooling layers, and 3) the fully connected (FC) layers. The first convolution layer uses convolution filters with a filter size of 3x3 used for initializing the neural network with some weights prior to updating to a better value for each iteration. The CNN architecture is formed from stacking these layers. Our self-acquired dataset which is composed of four different types of Malaysian mangosteen fruit, namely Manggis Hutan, Manggis Mesta, Manggis Putih and Manggis Ungu was employed for the training and testing of the proposed CNN model. The proposed CNN model achieved 94.99% classification accuracy higher than the optimized Xception model which achieved 90.62% accuracy in the second position.

## 1. INTRODUCTION

Mangosteen fruits, also referred to as manggis are tropical fruits abundantly available in Malaysia, and they are in categories, namely manggis hutan (jungle mangosteen), manggis mesta (mesta mangosteen), manggis putih (white mangosteen), and manggis ungu (purple mangosteen) as shown in Figure 1.

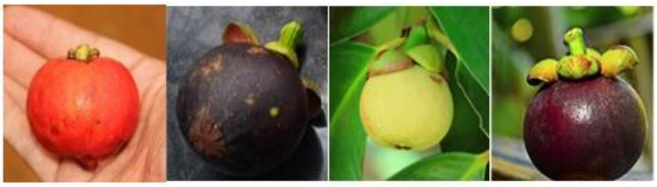


**Figure 1.** From left to right: Manggis Hutan, Manggis Mesta, Manggis Putih, and Manggis Ungu

Nowadays, fruit classification has become a necessity in the beverage industries. Fruit varies in characteristics such as sweetness, color, dimensions, shape, size, and so forth. It is common for farmers to sort their harvested fruits into individual species using traditional methods; this is to facilitate storage and sales of the produce among others.

However, the traditional methods are prone to so many challenges. Training a machine to identify and classify fruit types in place of traditional methods can ensure precision fruit classification. By taking advantage of the state-of-the-art image recognition techniques, we propose the classification of mangosteen fruits using hybrid deep learning.

The proposed model involves a proposed optimized Convolutional Neural Network (CNN) model compared to other optimized models such as Xception, VGG16, and ResNet50 using Adaptive moment estimation (Adam), RMSprop, Adagrad, and Stochastic Gradient Descent (SGD) optimizers on specified dense layers and filters numbers. The CNN model proposed in this study has three types of layers that make up its model, they are: 1) the convolutional layers, 2) the pooling layers, and 3) the fully connected (FC) layers. The first convolution layer uses convolution filters with a filter size of 3x3 used for initializing the neural network with some weights prior to updating to a better value for each iteration. The CNN architecture (structure) is formed from stacking these layers. Each model was structured uniquely based on suitability of the model on the given dataset by changing their classification layer. Parameters tuning was also conducted in this study in order to develop a reliable and accurate model. The final results of the models are then compared to one another to select the best performing model.

The rest of the work is as follows: Section 2 presents the related work, Section 3 describes the materials and methods, Section 4 presents the results and discussion, Section 5 concludes the work.

## 2. RELATED WORK

Deep learning model can be applied to build an image classification network using conventional neural networks [1, 2]. This becomes a core technology for artificial intelligence (AI) applications including fruit classification. Convolutional neural network (CNN) is one of the most popular algorithms for deep learning; it's useful for finding patterns in images to recognize the objects. They learn directly from image data, use patterns to classify images and eliminate the need for manual feature extraction. This is a kind of classification algorithm that can identify and classify fruits to their types.

Transfer learning, also referred to as knowledge transfer on the other hand transfers what the model has already learned to the new developed model for problem solving. The transfer learning consists of a previously trained model that releases some of the top layers of the fixed model foundation and attaches a new classification layer and a final layer of the base model [3]. The adaptation of this high-level feature representation in the model makes it suitable for specific tasks. This study optimized Xception transfer learning model to validate the performance accuracy of the proposed CNN model. Several works exist in the literature that addresses the problem of fruit recognition as an image segmentation problem.

Azizah et al. [4] built a system that detects Mangosteen fruits based on their color pixel. They studied Mangosteen fruit detection problems for outcome prediction. Bello et al. [5] proposed an enhanced Mask R-CNN for the segmentation of individual cow objects in the herd; the method employed CNN-based ResNet for features extraction, RPN (region proposal network) for object region proposals in the image, and FCL (fully connected layers) for the classification of the individual cow objects to their types. Femling et al. [6] proposed a classification system for a grocery store that can classify ten types of fruit. They used datasets consisting of images from ImageNet and camera-captured images. Xception [7], VGG16 [8] and ResNet50 [9] are some of the models mostly used as transfer learning in detection and classification problems.

Alkan et al. [10] reported in their paper a study which utilizes deep learning for automated detection of the symptoms of diseases on vine leaves. They proposed the improvement of disease detection accuracy in vine leaves and development of a system for Syrian and Turkish farmers and agricultural engineers to maintain the quality of grape production. The images they acquired were processed using MATLAB R2018b, Deep Learning Toolbox including CNNs with AlexNet, GoogleNet and ResNet18. A standard transfer learning algorithm was also used with CNNs, whereas a multiclass support vector machine (SVM) was used with AlexNet, whilst GPU and CUDA were used for accelerating the process of the disease detection for vine leaves.

Paranavithana and Kalansuriya [11] proposed in their study an approach based on CNN to develop a model that identifies and predicts the suitability of tea buds for the tea plucking as a solution to the myriad range of problems associated with tea picking. This was after they carried out a study on the proper procedure required for selecting tea leaves, and they found out that there was no proper procedure for carrying out such task. Moreover, the study revealed the dangers involved in having trust in tea buds picked using the conventional standards.

Nasir et al. [12] in order to classify fruits and the diseases that affect them employed the combination of deep neural network and contour feature-based approach. They employed useful features extracted from plant dataset to fine-tune and pretrain VGG19 deep learning model. This was followed by the application of pyramid histogram of oriented gradient (PHOG) to contour features extraction, and these features were added together to the deep features using serial-based approach. Their work was similar to the work of Palakodati et al. [13] who proposed CNN and transfer learning for classifying fresh and rotten fruits. However, in our work, we approach fruits classification from another perspective by introducing a high performing hybrid deep learning that can classify mangosteen fruits for precision agricultural application.

## 3. MATERIALS AND METHODS

As there are no off-the-shelf mangosteen fruit datasets available, we acquired our own dataset. We acquired 250 colored images for each of the 4 types of subspecies of mangosteen fruits making a total of 1000 images employed for the experiment. The images were manually cropped and resized to 224x224 pixels both in width and height without affecting the 3 color channels. Figure 2 shows the sample of mangosteen fruits in their respective categorical folders. For each of the 4 types of subspecies of the mangosteen fruits, 200 (80%) images were apportioned as the training dataset and 50 (20%) images were apportioned as the testing dataset. In order to increase the number of images, data augmentation was employed. Figure 3 shows the subcategories of the fruits augmented to acceptable degree of 250-image dataset collection threshold except manggis putih. Tensorflow and Keras were employed as Python libraries for both the training and testing of the proposed models on the platform of Google Colab.

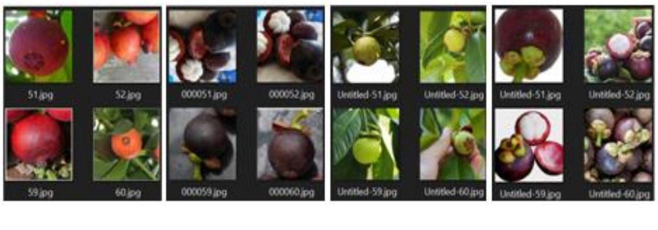


**Figure 2.** Sample of mangosteen fruits in their respective categorical folders

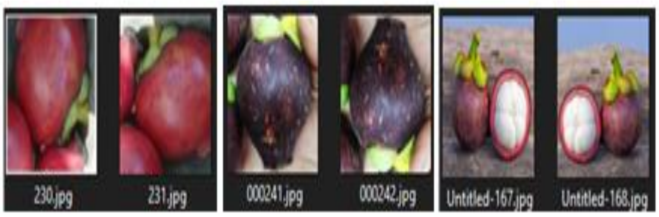
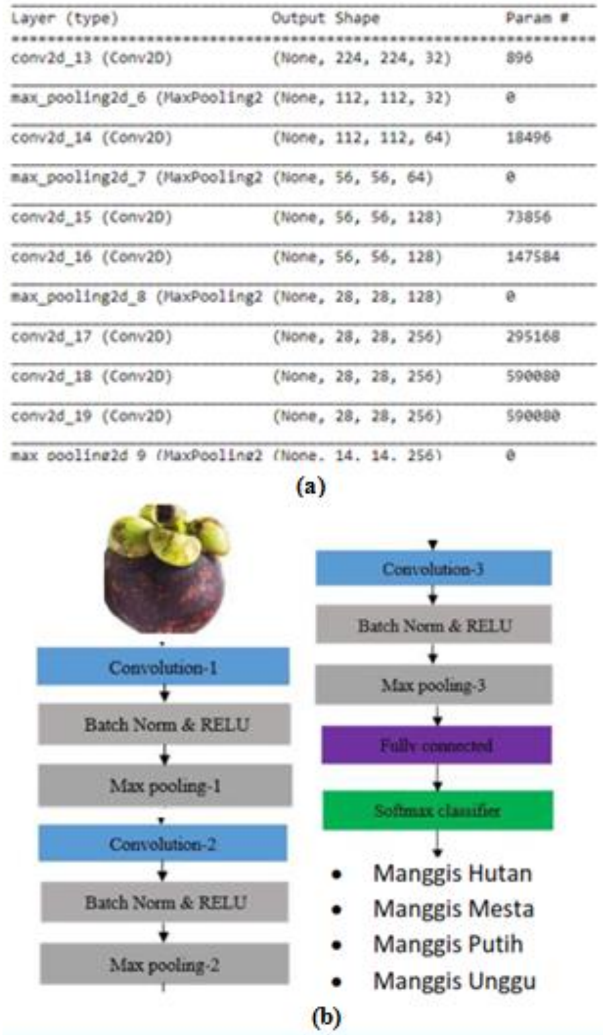


**Figure 3.** From left to right: augmentation samples showing image rotation in Manggis Hutan, horizontal axis flip in Manggis Mesta, and vertical axis flip in Manggis Ungu

### 3.1 Proposed hybrid deep learning model

We optimized Xception [7], VGG16 [8] and ResNet50 [9], and compared them to the proposed CNN model. The four optimizers used are Adam, RMSprop, Adagrad, and SGD optimizers. These models were trained on the acquired mangosteen fruit dataset, first in their base configuration, then in their optimized state, before comparing them with each other to determine the best performing model for our particular use case. While other models were optimized with variations of their optimizers, dense layers and epochs, CNN model on the other hand was optimized with variations of its optimizer, dense layers, and epochs including the filter numbers and learning rates in its optimization.

Figure 4 shows the architecture of the proposed CNN model and its application to manggis fruit classification. During the computation, the pixels of the picture were shown using a matrix. To detect a pattern, we used a filter multiplied by the pixel matrix of the image. The size of this filter may vary and the multiplication depends entirely on the size of the filter, and one can take a subset of the image pixel matrix. The convolution moved to the next pixel and this process was repeated until all the picture pixels in the matrix were complete.



| Layer (type) | Output Shape | Param # |
|---|---|---|
| conv2d_13 (Conv2D) | (None, 224, 224, 32) | 896 |
| max_pooling2d_6 (MaxPooling2 | (None, 112, 112, 32) | 0 |
| conv2d_14 (Conv2D) | (None, 112, 112, 64) | 18496 |
| max_pooling2d_7 (MaxPooling2 | (None, 56, 56, 64) | 0 |
| conv2d_15 (Conv2D) | (None, 56, 56, 128) | 73856 |
| conv2d_16 (Conv2D) | (None, 56, 56, 128) | 147584 |
| max_pooling2d_8 (MaxPooling2 | (None, 28, 28, 128) | 0 |
| conv2d_17 (Conv2D) | (None, 28, 28, 256) | 295168 |
| conv2d_18 (Conv2D) | (None, 28, 28, 256) | 590080 |
| conv2d_19 (Conv2D) | (None, 28, 28, 256) | 590080 |
| max_pooling2d_9 (MaxPooling2 | (None, 14, 14, 256) | 0 |

**Figure 4.** Architecture of the a) proposed CNN model for b) classifying manggis fruit

Overfitting and size of the output space were minimized by using the pooling layers. After normalizing Rectified Linear Unit (ReLU), the activation function was applied to each convolution. Finally, before using the dense layer, we flattened the map of the third convolution feature. Loss function of cross-entropy category and the Adam optimizer with a learning rate of 0.0001 were used to calculate adaptive learning rates for hyperparameters, and the model was trained at epochs 20, 50 and 80. The loss function calculates the loss by matching the actual value and the value predicted by the neural network. By using the loss function of cross-entropy category instead of the sum-of-squares for our classification problem leads to improved generalization as well as faster training. Our proposed model being a multi-class classification model has the task of predicting one of more than two class labels for a given example. We can, therefore, estimate the cross-entropy for a single prediction using the cross-entropy calculation as described in the equation below.

H(P, Q) = – sum x in X P(x) * log(Q(x))

where, each x in X is a class label that could be assigned to the example, and P(x) will be 1 for the known label and 0 for all other labels.

### 3.2 Optimized transfer learning models

Xception is a model that is structured into 14 modules and is made up of 36 convolutional layers [7]. Xception’s architecture can be broken up into 3 parts, namely 1) the Entry flow, 2) the Middle flow, and 3) the Exit flow as shown in Figure 5. We optimized the default input layer of Xception from 299x299x3 to 224x224x3 by importing a custom shape into the optimized model in order to fit our dataset images, and left out the top layer, thereby creating our own classifier instead of using the available default one. Moreover, our model made use of the ImageNet weights to get the benefit of a transfer model that’s already pretrained with a comprehensive dataset to avoid retraining. The Xception model was frozen once it was loaded to lock the weights and biases of the model’s layers, thereby ensuring their stability in later training.

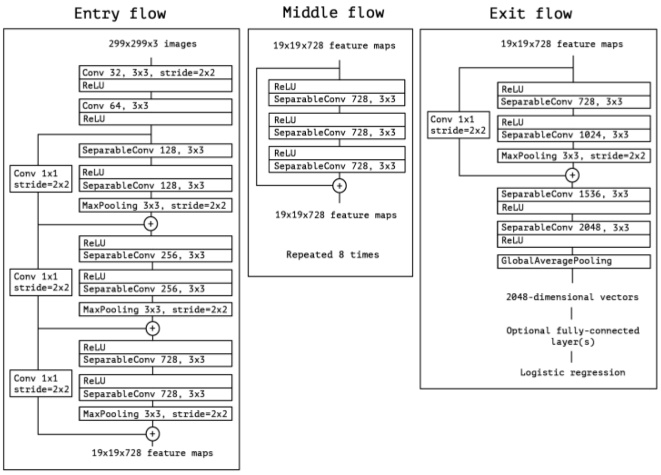
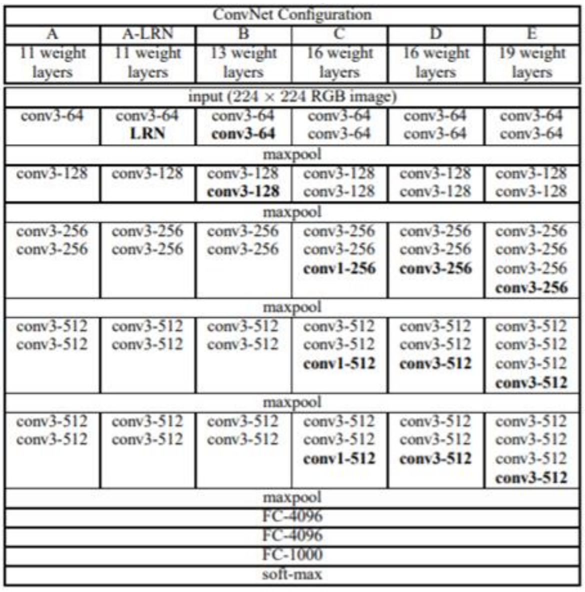


**Figure 5.** Xception's architecture

VGG16 as another model employed in this study consists of 16 layers, and the model uses 3x3 convolution filters to reduce the number of parameters. It comprises a size 2×2 Max pooling layer and a total of 5 such layers. The arrangement of VGG16 networks is such that after the last Max pooling layer is three fully connected layers. Its final layer comprises the softmax classifier, and all hidden layers undergo the application of ReLu activation. Moreover, VGG16 network gives excellent performance irrespective of the small quantity of image datasets used and this is due to the extensive training it has undergone [8]. VGG16 was employed in this study to create a transfer learning model for the Manggis recognition model. The model was imported along with its weight; however, the top layers of the model which are the two fully connected layers (FCL) with its output layers were dropped. The pixel dimension 224x224 same as the pixel dimension of the acquired dataset was used. Figure 6 shows the schematic architecture of the VGG16 network.



| ConvNet Configuration | | | | | |
|---|---|---|---|---|---|
| A | A-LRN | B | C | D | E |
| 11 weight layers | 11 weight layers | 13 weight layers | 16 weight layers | 16 weight layers | 19 weight layers |
| input (224 × 224 RGB image) | | | | | |
| conv3-64 | conv3-64<br>**LRN** | conv3-64<br>**conv3-64** | conv3-64<br>conv3-64 | conv3-64<br>conv3-64 | conv3-64<br>conv3-64 |
| maxpool | | | | | |
| conv3-128 | conv3-128 | conv3-128<br>**conv3-128** | conv3-128<br>conv3-128 | conv3-128<br>conv3-128 | conv3-128<br>conv3-128 |
| maxpool | | | | | |
| conv3-256<br>conv3-256 | conv3-256<br>conv3-256 | conv3-256<br>conv3-256 | conv3-256<br>conv3-256<br>**conv1-256** | conv3-256<br>conv3-256<br>**conv3-256** | conv3-256<br>conv3-256<br>conv3-256<br>**conv3-256** |
| maxpool | | | | | |
| conv3-512<br>conv3-512 | conv3-512<br>conv3-512 | conv3-512<br>conv3-512 | conv3-512<br>conv3-512<br>**conv1-512** | conv3-512<br>conv3-512<br>**conv3-512** | conv3-512<br>conv3-512<br>conv3-512<br>**conv3-512** |
| maxpool | | | | | |
| conv3-512<br>conv3-512 | conv3-512<br>conv3-512 | conv3-512<br>conv3-512 | conv3-512<br>conv3-512<br>**conv1-512** | conv3-512<br>conv3-512<br>**conv3-512** | conv3-512<br>conv3-512<br>conv3-512<br>**conv3-512** |
| maxpool | | | | | |
| FC-4096 | | | | | |
| FC-4096 | | | | | |
| FC-1000 | | | | | |
| soft-max | | | | | |

**Figure 6.** Architecture of VGG16 model

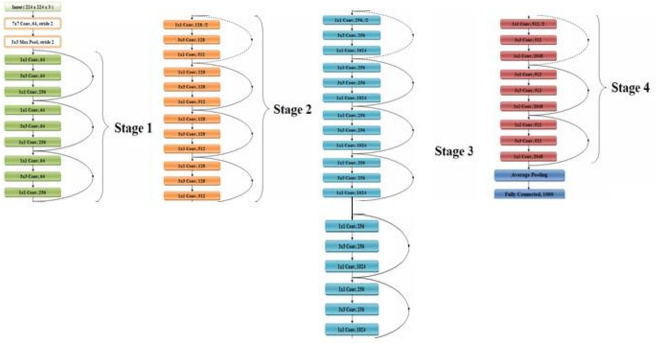


**Figure 7.** Architecture of ResNet50

ResNet50 is a 50-layer residual deep learning neural network model pre-trained for image classification [9]. We imported the input size 224x224x3 for fitting our dataset images, followed by using 64, 64, and 128 size kernels to perform the convolution process in all the three layers of stage 1 of the network which consists of three Residual blocks, each with three layers. The identity connection is represented by the curved arrows as shown in Figure 7. The dashed connecting arrow indicates that the convolution operation in the Residual block is performed with stride 2, resulting in a half-size input in terms of height and width but a doubling channel width. Advancing from one stage to another, the channel width is doubled and the size of the input is reduced to half. After training and testing the model with the training dataset and testing dataset respectively to evaluate their performance, the parameters and results obtained are presented and discussed under results and discussion.

## 4. RESULTS AND DISCUSSION

The effect of optimizer and the learning rate are shown in Figure 8 and Figure 9 respectively. The weights that are updated during the training of network are referred to as learning rate. These are important hyperparameters used in the CNN model that range between 0.0 and 1.0 [6]. In our CNN model, we used three learning rates and observed the influence of those learning rates on accuracy. The three learning rates were 0.1, 0.01 and 0.001. From Figure 9, it is observed that after lowering the learning rate from 0.1 to 0.0001, the accuracy improves, whereby clarifying the highest accuracy provided by our CNN model.

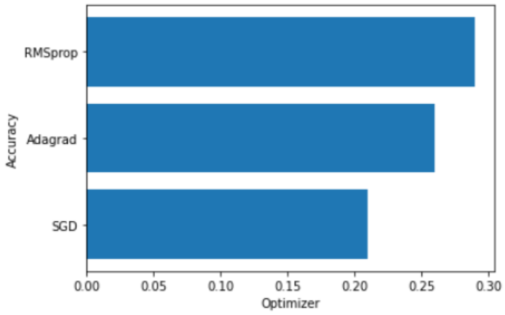


**Figure 8.** Optimizer performance

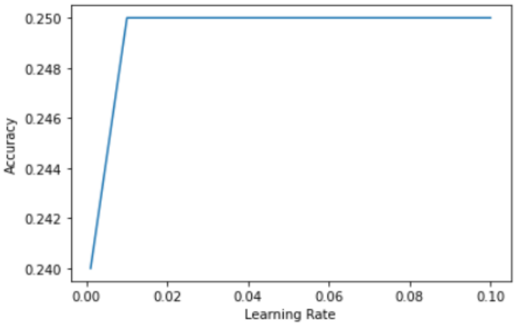


**Figure 9.** Learning rate performance

From Table 1, Adam optimizer produced 94.99% accuracy with the proposed CNN model when evaluated on the dataset. The tendency of different hyperparameters such as group size, number of times, optimizer, and learning rate to produce high accuracy values depends on the suitability of our dataset. The use of learning rates of 0.01 and 0.1 delays execution and produces poor result. The frozen Xception model was added to a new empty model as the base layer which was utilized in training the model with the training dataset, and then topped it off with a 4-unit classifier to make the default Xception model get a baseline performance benchmark.

The performance of the trained model was also tested with the testing dataset. In total, 8 experiments were conducted. The parameters and results are presented in Table 2. Based on Figure 10, comparing the effects of varying the epochs and switching between Adam and SGD optimizers, the best results were obtained from epoch 50 and Adam optimizer. While the results of the training and testing accuracies for both epoch parameters using the SGD optimizer is lower than the values obtained using the Adam optimizer for the same epoch, we can see that besides the positive influence of the Adam optimizer, increasing the epoch also stabilizes the model's accuracy as well as the losses as the 5-epoch experiment on both optimizers indicates an incomplete trend, still increasing and decreasing for both accuracy and loss respectively. The 50-epoch experiment for both optimizers on the other hand, show both the accuracy and loss values plateauing after epoch 10 or so, but for the purpose of this experiment, we prefer using epoch value 50 in order to be more accurate.

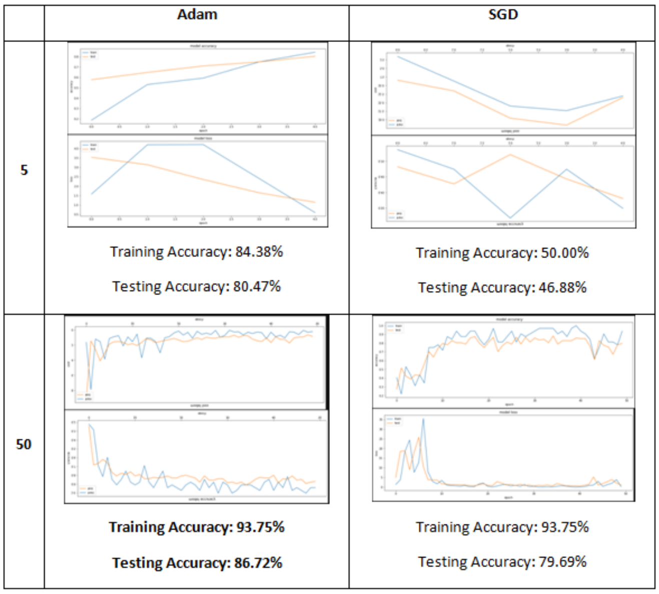
Although, the 1 layer 1024 neuron model produced testing accuracy that is less than the testing accuracy of the 2 layer 1024, 1024 neuron model, it has highest training accuracy, thereby considered as the best optimized model from the Xception transfer learning experiment. The implication of this is that, the fitting problem of base Xception model is mitigated. The result of the optimization between SGD and Adam optimizers using 5 and 50 epochs is shown in Figure 11. As shown in the Figure 11, Adam optimizer yields higher training score result than the score yielded by SGD.

**Table 1.** Accuracy by layer and optimization

| Optimizer | Adam | RMSprop | Adagrad | SGD |
|---|---|---|---|---|
| Epoch | 50 | | | |
| Number of dense layers | **1** | 2 | 1 | 1 |
| Learning rate | **0.0001** | 0.0001 | 0.001 | 0.1 |
| Validation steps/step per-epoch | **1** | 1 | 1 | 1 |
| Model accuracy | **94.99%** | 79.00% | 66.52% | 64.99% |

**Table 2.** Performance by layer and neuron optimization

| Optimizer | Adam | | | | |
|---|---|---|---|---|---|
| Epoch | 50 | | | | |
| Number of dense layers | 0 | 1 | **1** | **2** | 3 |
| Neuron configuration | 0 | 128 | **1024** | **1024, 1024** | 2048, 1024, 1024 |
| Training accuracy | 93.75% | 96.88% | **89.84%** | **78.12%** | 100.00% |
| Testing accuracy | 86.72% | 85.16% | **90.62%** | **91.41%** | 83.59% |



Training Accuracy: 84.38%

Testing Accuracy: 80.47%

Training Accuracy: 50.00%

Testing Accuracy: 46.88%

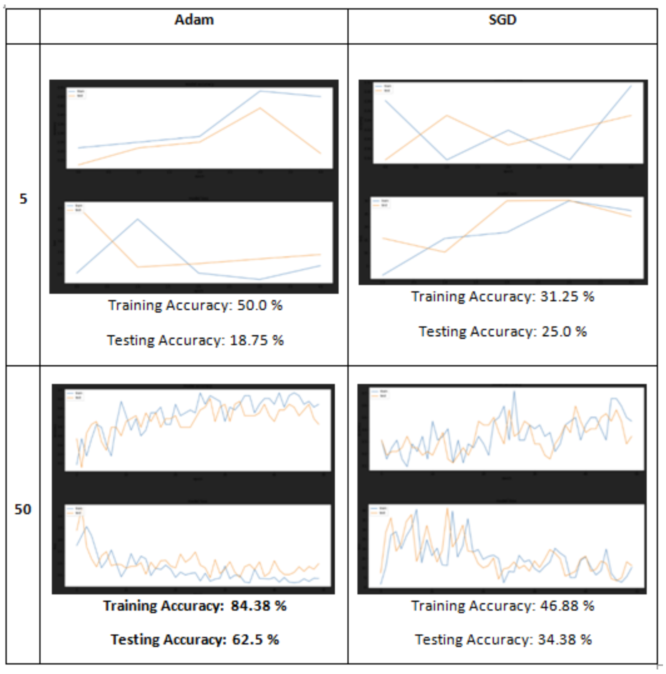
**Training Accuracy: 93.75%**

**Testing Accuracy: 86.72%**

Training Accuracy: 93.75%

Testing Accuracy: 79.69%

**Figure 10.** Comparison of the effects of epoch and optimizer



Training Accuracy: 50.0 %

Testing Accuracy: 18.75 %

Training Accuracy: 31.25 %

Testing Accuracy: 25.0 %

**Training Accuracy: 84.38 %**

**Testing Accuracy: 62.5 %**

Training Accuracy: 46.88 %

Testing Accuracy: 34.38 %

**Figure 11.** Comparison of the effects of epoch and optimizer

**Table 3.** Performance by layer and neuron optimization

| Optimizer | Adam | | | | |
|---|---|---|---|---|---|
| Epoch | 50 | | | | |
| Number of dense layers | 0 | 1 | **2** | 2 | 3 |
| Neuron configuration | 0 | 100 | **1000, 500** | 1024, 1024 | 1000, 1000, 1000 |
| Training accuracy | 84.38% | 90.62% | **90.62%** | 87.50% | 93.75% |
| Testing accuracy | 62.5% | 65.62% | **71.88%** | 71.88% | 69.75% |

Table 3 shows the result comparison of each optimization made. The addition of fully connected layers increases the result of the models' training accuracy. However, it is observed that adding more layers and neurons to the model than necessary leads to overfitting. Figure 12 shows the comparison of Adam and SGD optimizers. Based on Table 4, comparing the effects of varying the epochs and switching between Adam and SGD optimizers, the better result was obtained by Adam optimizer at epoch 5 higher than what was obtained by SGD optimizer at the same epoch of 5, making Adam optimizer at epoch 5 suitable for the experiments.

From Table 4, it is observed that increasing dense layers impacts performance negatively because the more the layers, the lower the testing accuracies. Moreover, raising the neuron values from 128 to 1024 results in immediate performance gain between the 1 layer models, however, there is fitting issue as the layers increase. With these findings, 1-layer 1024 neuron model produced acceptably high model accuracy, thereby making it considered as the best optimized model from the ResNet50 transfer learning experiment. Table 5 shows the performance across models. Out of all the models tested, the proposed CNN model produced 94.99% accuracy higher than Xception that produced 90.62% accuracy. Figure 13 shows epoch 50 obtaining 94.99% accuracy, thereby producing the highest accuracy so far. The CNN model takes quite an effort to build and optimize. The Xception model on the other hand was quite simple to import, optimize and computationally fast as it is already pretrained.

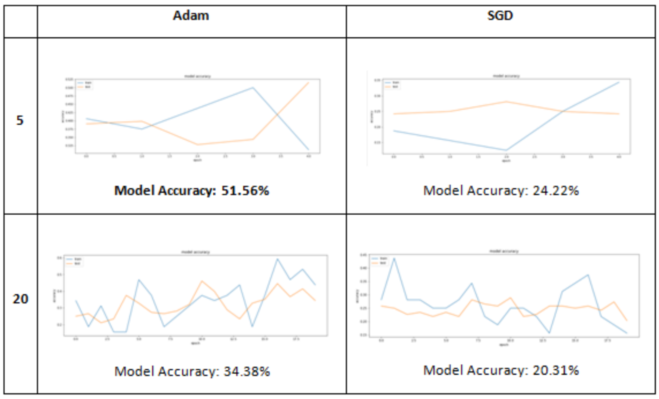


**Figure 12.** Comparison on epoch and optimizer

**Table 4.** Performance by layer and neuron optimization

| Optimizer | Adam | | | | | |
|---|---|---|---|---|---|---|
| Epoch | 50 | | | | | |
| Number of dense layers | 0 | 1 | **1** | 2 | 3 | 3 |
| Neuron configuration | 0 | 128 | **1024** | 1024, 1024 | 100, 100, 100 | 2048, 1024, 1024 |
| Model accuracy | 27.34% | 25.00% | **51.56%** | 34.38% | 45.5% | 25.00% |

**Table 5.** Performance across models

| Model Name | Proposed CNN | Xception | VGG16 | ResNet50 |
|---|---|---|---|---|
| Best configuration | **Optimizer: Adam**<br>**Epoch: 50**<br>**1 dense layer**<br>**Learning rate: 0.0001** | **Optimizer: Adam**<br>**Epoch: 50**<br>**1 dense layer**<br>**1024 neurons** | Optimizer: Adam<br>Epoch: 50<br>2 dense layers<br>1000, 500 neurons | Optimizer: Adam<br>Epoch: 50<br>1 dense layer<br>1024 neuron |
| Best performance | **94.99%** | **90.62%** | 71.88% | 51.56% |

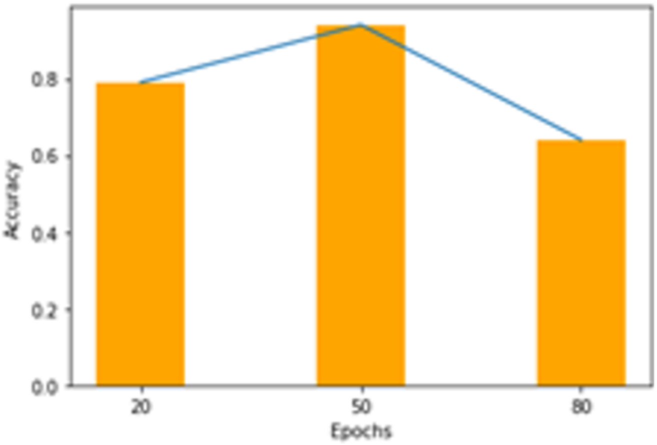


**Figure 13.** Epoch performance

## 5. CONCLUSIONS

We approach fruits classification from another perspective in this study by introducing a high performing hybrid deep learning that can classify mangosteen fruits for precision agricultural application. This involved an optimized Convolutional Neural Network (CNN) model compared to other optimized models such as Xception, VGG16, and Resnet50 using Adam, RMSprop, Adagrad, and SGD optimizers on specified dense layers and filters numbers. By using the CNN model, we can accurately classify mangosteen fruits to their types based on their color pixels. Although we achieved reasonable results from our models, two key improvements are still necessary for optimal performance. The first improvement is to increase the dataset. The second improvement is to utilize systems that have high computational power as this would improve the optimizations of the models.

## REFERENCES

[1] Mettleq, A.S.A., Dheir, I.M., Elsharif, A.A., Abu-Naser, S.S. (2020). Mango classification using deep learning. International Journal of Academic Engineering Research, 3(12): 22-29.

[2] Gullapelly, A., Banik, B.G. (2021). Classification of rigid and non-rigid objects using CNN. Revue d'Intelligence Artificielle, 35(4): 341-347. https://doi.org/10.18280/ria.350409

[3] Thenmozhi, K., Reddy, U.S. (2019). Crop pest classification based on deep convolutional neural

network and transfer learning. Computers and Electronics in Agriculture, 164: 104906. https://doi.org/10.1016/j.compag.2019.104906

[4] Azizah, L.M.R., Umayah, S.F., Riyadi, S., Damarjati, C., Utama, N.A. (2017). Deep learning implementation using convolutional neural network in mangosteen surface defect detection. 7th IEEE International Conference on Control System, Computing and Engineering, Penang, Malaysia, pp. 242-246. https://doi.org/10.1109/ICCSCE.2017.8284412

[5] Bello, R.W., Mohamed, A.S.A., Talib, A.Z. (2021). Enhanced mask R-CNN for herd segmentation. International Journal of Agricultural and Biological Engineering, 14(4): 238-244. https://doi.org/10.25165/j.ijabe.20211404.6398

[6] Femling, F., Olsson, A., Alonso-Fernandez, F. (2018). Fruit and vegetable identification using machine learning for retail applications. In: 14th International Conference on Signal-Image Technology & Internet-Based Systems, Las Palmas de Gran Canaria, Spain, pp. 9-15. https://doi.org/10.1109/SITIS.2018.00013

[7] Chollet, F. (2017). Xception: Deep learning with depthwise separable convolutions. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, pp. 1251-1258. https://doi.org/10.1109/CVPR.2017.195

[8] Simonyan, K., Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556

[9] Gupta, U. (2017). Detailed guide to understand and implement ResNets. https://cv-tricks.com/keras/understand-implement-resnets/.

[10] Alkan, A., Abdullah, M.Ü., Abdullah, H.O., Assaf, M., Zhou, H. (2021). A smart agricultural application: automated detection of diseases in vine leaves using hybrid deep learning. Turkish Journal of Agriculture and Forestry, 45: 1-13. https://doi.org/10.3906/tar-2007-105

[11] Paranavithana, I.R., Kalansuriya, V.R. (2021). Deep convolutional neural network model for tea bud(s) classification. IAENG International Journal of Computer Science, 48(3): 1-6.

[12] Nasir, I.M., Bibi, A., Sha, J.H., Khan, M.A., Sharif, M., Iqbal, K., Nam, Y., Kadry, S. (2021). Deep learning-based classification of fruit diseases: An application for precision agriculture. CMC-Computers Materials and Continua, 66(2): 1949-1962. https://doi.org/10.32604/cmc.2020.012945

[13] Palakodati, S.S.S., Chirra, V.R., Yakobu, D., Bulla, S., (2021). Fresh and rotten fruits classification using CNN and transfer learning. Revue d'Intelligence Artificielle, 34(5): 617-622. https://doi.org/10.18280/ria.340512