Article

Classification of Adulterated Particle Images in Coconut Oil Using Deep Learning Approaches

Attapon Palananda and Warangkhan Kimpan *

Department of Computer Science, School of Science, King Mongkut's Institute of Technology Ladkrabang, Bangkok 10520, Thailand; 57605017@kmitl.ac.th

* Correspondence: warangkhan.ki@kmitl.ac.th

Abstract: In the production of coconut oil for consumption, cleanliness and safety are the first priorities for meeting the standard in Thailand. The presence of color, sediment, or impurities is an important element that affects consumers' or buyers' decision to buy coconut oil. Coconut oil contains impurities that are revealed during the process of compressing the coconut pulp to extract the oil. Therefore, the oil must be filtered by centrifugation and passed through a fine filter. When the oil filtration process is finished, staff inspect the turbidity of coconut oil by examining the color with the naked eye and should detect only the color of the coconut oil. However, this method cannot detect small impurities, suspended particles that take time to settle and become sediment. Studies have shown that the turbidity of coconut oil can be measured by passing light through the oil and applying image processing techniques. This method makes it possible to detect impurities using a microscopic camera that photographs the coconut oil. This study proposes a method for detecting impurities that cause the turbidity in coconut oil using a deep learning approach called a convolutional neural network (CNN) to solve the problem of impurity identification and image analysis. In the experiments, this paper used two coconut oil impurity datasets, PiCO_V1 and PiCO_V2, containing 1000 and 6861 images, respectively. A total of 10 CNN architectures were tested on these two datasets to determine the accuracy of the best architecture. The experimental results indicated that the MobileNetV2 architecture had the best performance, with the highest training accuracy rate, 94.05%, and testing accuracy rate, 80.20%.

Keywords: coconut oil; deep learning; convolutional neural networks; image recognition; object detection

Citation: Palananda, A.; Kimpan, W. Classification of Adulterated Particle Images in Coconut Oil Using Deep Learning Approaches. *Appl. Sci.* **2022**, *12*, 656. <https://doi.org/10.3390/app12020656>

Academic Editor: Byung-Gyu Kim

Received: 10 December 2021

Accepted: 7 January 2022

Published: 10 January 2022

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Copyright: © 2022 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Coconut oil is a fatty acid-rich oil that is healthful for human consumption. The key to production is maintaining the benefits of the coconut oil. The production process that maintains the highest value of coconut oil is cold pressed, which produces virgin coconut oil. The process starts with the dehumidification of coconut meat at approximately 65–70 degrees Celsius; the meat is then compressed to extract the oil. To remove impurities, a centrifuge is used to circulate the oil through a particle filter. These impurities contribute to the turbidity of the coconut oil. Examples of such impurities are coconut scraps, coconut shell scraps, coconut coir, and dust particles that are retained after the oil compression process. Impurities that remain after the impurity filtering process will become suspended sediment that will sink to the bottom of the container until it can be seen with the naked eye. Large amounts of sediment in the coconut oil can cause turbidity. Turbidity can be checked in coconut oil in the same way as in water. Karnawat and Patil [1] photographed water and applied image processing. Many researchers have also applied image processing methods to help analyze other problems, such as digital image analysis of coarse dust particles or aerosols transmitted by laser light [2], seafloor plankton analysis using image processing and machine learning [3], automatic screening classification of diabetic retinopathy from

fuzzy image processing combined with machine learning [4], photographic analysis of airborne PM_{2.5} (particulate matter) and PM_{1.0} adhering to a microscopic glass surface [5], and three-step real-time vehicle tracking method on the road using the Adaboost algorithm together with the image segmentation method [6]. Consequently, this paper uses deep learning methods to identify adulterating objects in coconut oil is proposed. As coconut oil flowed through the pipeline to the storage tank, a microscope was used to photograph the oil in a closed environment. The front lens of the microscope inside receives light from a black-light lamp that shines through the coconut oil. The black light does not reflect light onto objects, making the objects distinct. On the basis of the steps mentioned above, microscopic photographs of the coconut oil were recorded at different times of the production process, resulting in 4725 photographs of the coconut oil.

The next step involved the process of finding and identifying impurities in the coconut oil images. A learning model with the standard neural network architecture was built when the image of the adulterating object in the coconut oil was taken. In this paper, the performances of a standard convolutional neural network (CNN), MobileNetV2, VGGNet16, VGGNet19, GoogLeNet (Inception V3, Xception, and InceptionResNetV2), ResNet50, ResNet101, and DenseNet121 were compared in terms of image processing speed and accuracy. The best-performing model was then used to detect impurities in the coconut oil.

The rest of this paper is structured as follows: Section 2 discusses the literature review. Then, the research methodology is shown in Section 3. Finally, Section 4 presents the research conclusions and future work.

2. Literature Review

2.1. Deep Learning

Deep learning is a subset of machine learning (ML) within artificial intelligence (AI). Based on artificial neural networks (ANNs), deep learning simulates the process of information processing in the human brain. An ANN consists of interconnected cells as the nervous system used for data processing. Deep learning was first presented through the implementation of ML by Rina Dechter [7,8]. It is a method that does not use data manipulation to work through predefined equations; rather, it uses basic data-related parameter settings and teaches computers to experience self-learning by processing to identify the strengths and differences in the data. Deep learning is filtered by layers according to the nature of the neural networks to process the output answers and to examine whether the output is right or wrong. If the examined results of the output data do not match reality, the learning settings will be adjusted and the data reprocessed to increase the accuracy of the output by increasing the number of samples or adding layer depth to consider more details. Therefore, deep learning can consider differences in the data and provide output answers without a need for human suggestions. Therefore, the order of the number of layers is the depth in deep learning.

2.2. Convolutional Neural Networks

The CNN method was introduced by Fukushima which is a neural network with a convolutional layer added. It performs as a filter, defining the attributes of an input image and collecting various features from each image for classification. In 2019, Fukushima [9] developed neocognitron. Unlike standard CNN, neocognitron was used to develop recognition of partly occluded patterns via the mechanism of selective attention. The backpropagation neural network was first stated by Rumelhart to help optimize prediction of data classification by internal restructuring using hidden layers. LeCun [10] tested backpropagation with a CNN on the Modified National Institute of Standards and Technology (MNIST) dataset, a handwritten numeric dataset. Then in 2019, they proposed deep learning hardware: past, present, and future which described the evolution of neural networks and deep learning affected by hardware and software improvements. In 2020, Li et al. [11] proposed improvements of backpropagation to be used for deep transfer learning. They used a

pretrained model to transfer knowledge which learned from larger datasets to the target task. In 2007, Ranzato et al. [12] first presented the use of backpropagation in collaboration with max-pooling CNNs (MPCNNs). Then, in 2018, Brinker et al. [13] proposed a CNN approach to skin cancer screenings. In 2020, Patil and Bellary [14] presented a method for identifying the growth of melanoma cancer cells using a CNN approach.

Newby et al. [15] proposed ANN approach in particle automatic tracking to analyze particle localization and track particle motion from 2D and 3D collected video images obtained through a microscope.

Khairi et al. [16] proposed a method for determining the turbidity level of water samples from the application of X-ray imaging systems. An ANN was then used to help determine the turbidity level. This proposed method was able to predict the change in turbidity level. It helped classify water quality levels and benefited industries that require high water quality.

Rong et al. [17] proposed two different convolutional neural networks using deep learning for automatically segmenting images and detecting different sizes of foreign objects that occur both naturally and man-made, such as dry leaves, scrap of papers, plastic scraps, and metal parts. This proposed structure was applied to walnut images. The experimental results had an accuracy of 99.5% in object segmentation and it was able to correctly classify 95% of the foreign objects and found that the segmentation and detection processing time of each image was less than 50 ms. Ferreira et al. [18] proposed a method for weed detection from aerial photographs of soybean fields with drones. CNN was used for weed classification in four classes: soil, soybean, broadleaf, and grass weeds. They used the CaffeNet framework based on the AlexNet architecture model. The experimental results showed that CNN had an accuracy of 99% which was better than SVM, AdaBoost, and random forest.

The core structure of a CNN consists of the three following principles: first is the input, which is used to receive information or object images as well as human vision. Second is the hidden layer, which is used to process the input data as well as the human brain and can learn to classify things. Last is the output, which shows the results obtained from the processing of the second part. The standard structure of a CNN is shown in Figure 1.

Figure 1. CNN architecture.

Figure 1 describes the general structure of a CNN, which consists of the following parts:

1. **Input layer:** reads the image input data prior to passing it to the neural network.
2. **Convolutional layer:** filters the image features from the analysis of each pixel of the image that has been read. The result is a convolutional feature map.
3. **Rectified linear unit (ReLU):** performs a nonlinear activation function.

4. Pooling layer: provides a subsample rectified feature map to reduce linear dimensions and create a feature representation.
5. Softmax layer: configures the output to display it in the form of a multiclass logistic classifier.
6. Output layer: displays the results of the classification.

The CNN concept was further developed. One method called AlexNet [19], which has eight convolutional layers, competed in the ImageNet Large-Scale Visual Recognition Challenge in 2010 (ILSVRC2010). After that achievement, researchers used the CNN concept to develop various models. For example, in 2014, Simonyan and Zisserman [20,21] presented the VGGNet approach. From 2014 to 2016, the Google team proposed the GoogLeNet, or inception method. Then, in 2017, Howard et al. [22] presented the MobileNet approach, which is a small model that can work quickly and uses fewer processing resources. This has made the MobileNet architecture popular for mobile devices, and we will focus on it in this paper.

2.3. MobileNet

MobileNet is a CNN approach introduced by Howard et al. [22]. It aims to reduce the size of the standard CNN model for use in mobile devices. It can also maintain a performance level close to that of a large-scale deep learning neural network. MobileNet takes an input image with dimensions of $224 \times 224 \times 3$ and passes it through the convolution layers. It demonstrates the ability of small artificial neural networks to recognize objects quickly without the use of a GPU. The structure of MobileNet is shown in Figure 2. In 2020, Pan et al. [23] proposed a MobileNet approach combined with a transfer learning algorithm (TL-MobileNet) for image analysis and recognition. This solved the problem of the new image classification accuracy. DropBlock and global average were added to the original MobileNet layer. The results showed that the model had 97.69% prediction accuracy on the MNIST dataset. Kerf et al. [24] presented detection of oil spills in water using MobileNet architecture which helped decision-making process and provide solutions of oil spills on the surface of water. Drone was used to explore and photograph with an infrared camera which can be used at night. Both RGB and IR (Infrared) images were processed. In the initial process, RGB images encountered with the process of extract water region press, quantify oil in water, and create mask and image resize, while IR images undergo only resize. RGB and IR images were then synchronized and calibrated. The resulting IR images and RGB masks were then used for data augmentation process to train the CNN model. For the testing method, began with the IR images were resized and then the images were taken in a pre-learning CNN on interface device process to detect oil spills on the water surface. As a result, when an oil leak was found, a GPS alerted the oil leak position to the monitoring system.

Figure 2. MobileNet architecture.

In 2021, Iqbal et al. [25] presented blockage classification of culverts by remotely applying deep learning models. They used three datasets: images of culvert openings and blockage (ICOB), visual hydrology-lab dataset (VHD), and synthetic images of culverts (SIC), to predict the blockage in the image. The performances of CNN algorithms (DarkNet53, DenseNet121, InceptionResNetV2, InceptionV3, MobileNet, ResNet50, VGG16, EfficientNetB3, NASNet) were compared in terms of image processing response times and accuracy. The results showed that the MobileNet is effective in classification performance and response times.

2.4. VGGNet

VGGNet was developed by the visual geometry group (VGG) at the University of Oxford. In 2014, Simonyan and Zisserman [20,21] proposed a VGG network architecture. This model implemented a 3×3 convolutional layer that added a step above the standard CNN model (max-pooling, fully connected, and softmax layers). The experimental results of VGG11, VGG11 (local response normalization (LRN)), VGG13, VGG16 (Conv 1), VGG16, and VGG19 showed that VGG16 and VGG19 had the lowest error values. The numbers 16 and 19 refer to the number of layers. The error value of VGG19 with 19 layers is higher than that of VGG16 with 16 layers, which means that a higher number of layers cannot reduce the error values. In addition, the size of the VGG19 model is larger than that of the VGG16 model. In 2019, Tammina [26] proposed a solution to a limited number of image classifications by training the VGG16 model with the minimum number of samples. The experiment showed the accuracy and loss after the CNN parameter was adjusted, with the lowest accuracy being 79.20%. The number of images was then increased to train the model, and the new accuracy value was 95.40%.

2.5. GoogLeNet

GoogLeNet has 22 stacked convolutional layers. It was developed by Google researchers as a form of inception network. Beginning in 2015, Inception V1 was developed by Szegedy et al. [27] to increase the ability to examine and classify individual objects. This method includes different convolutional (1×1 , 3×3 , and 5×5) and 3×3 max-pooling node sizes, resulting in better performance than that of VGGNet in the Large-Scale Visual Recognition Challenge 2014 (ILSVRC14). After this achievement in the competition, GoogLeNet improved its training performance with a batch normalization technique, known as Inception V2, and later upgraded to Inception V3. It factored the convolutional node into the inception module and made it smaller. Inception V4 improved Inception V3 by adding wider inception modules than those used in Inception V3. Performance tests showed that Inception V4 had the lowest error rate and spent the most time training the model compared to the previous 3 models.

2.6. ResNet

ResNet, or residual neural network, was first presented by He et al. [28]. In this study, deep residual learning for image recognition was used to solve vanishing gradient problems. The authors used residual block learning methods instead of learning certain properties of the image. This approach removes the properties learned from the input of that layer. If the number of layers reaches 152, a shortcut module design technique is applied to the network to cross layers to reduce errors in training the deep layer CNN model. In training ResNet models, the number of parameters is the layer used for naming, such as ResNet50 or ResNet101. The structure of ResNet50 has dimensions (3,4,6,3), which means $(3 + 4 + 6 + 3) \times 3 = 48$ layers + 2 layers, or 50 layers, as shown in Figure 3.

Figure 3. ResNet50 architecture.

In 2020, Aparna et al. [29] proposed the detection of holes from thermal images using convolutional neural networks and the comparison of each ResNet model. The results obtained from the confusion matrix table showed that ResNet101 had an accuracy of 97.08% with thermal images obtained from the FLIR ONE thermal camera.

2.7. DenseNet

DenseNet, or dense convolutional network, was proposed in 2019 by Huang et al. [30] as an innovation codeveloped by Cornwell University and Tsinghua University. In DenseNet, each layer receives input from all previous layers and passes the attributes of its own layer to the following layer. This connection allows each layer to receive data from all previous layers. The connection between the DenseNet data layers is shown in Figure 4.

In 2018, Tao et al. [31] proposed a solution to the problem of ambiguity in the classification of sensitive data that enhanced the depth and width of the deep neural network (DNN) through multiple filter sizes to create a wider neural network. Moreover, the proposed process enables the network to work more smoothly.

Later, Too et al. [32] proposed a method to classify plant diseases into 38 different groups based on images of diseased and unaffected plant leaves of 14 healthy species by comparing VGG16, InceptionV4, ResNet50, ResNet101, ResNet152, and DenseNet121. The results showed that the DenseNet architecture achieved an accuracy of 99.75%, which was the highest accuracy of all the architectures that were tested.

Figure 4. Connection in DenseNet.

3. Research Methodology

This section explains how to obtain impurities from coconut oil images flowing through a microscope. This method was proposed in a previous study by Palananda and Kimpan [33] that focused on determining coconut oil turbidity by image processing. In this paper, we relied on the same method of obtaining images used in our previous study. Coconut oil images were recorded at different time intervals, and ultimately, 4725 images were obtained. All the obtained coconut oil images underwent an image processing technique to find impurities according to the process of finding objects in coconut oil, as shown in Figure 5. The process started with converting the images from RGB (red, green, blue) to BGR (blue, green, red). Then, the images were converted from BGR to grayscale. This reduced 3 layers to 1 layer. Next, the images were converted from grayscale to black-and-white or binary images, which involves converting the image pixel value from 0–255 levels to 0 or 1. Next, the image enhancement process was performed to reduce noise with the closing method, which is one of the commonly used morphological methods. Then, we quantified the total pixels, which meant gathering more than 30 pixels in an image and creating a line around the group of pixels or objects using the Sobel method, which searches for differences in the pixel intensity. This avoids the image noise left over from the image enhancement process. Finally, we used the segmentation method to crop the resulting object, meaning the adulterating object in the coconut oil image that we were interested in.

Figure 5. Object image acquisition in coconut oil.

As shown in Figure 5, after the coconut oil images were taken through the process of finding adulterants in coconut oil, a total of 14,784 images were captured, some of which were apparently incomplete. These images were not filtered by experts. Individual object images and indistinguishable or incomplete object images were filtered out, resulting in 7861 images of adulterating objects in the coconut oil.

After that, we classified the image of the impurity objects to new 10 categories as: (1) FiberT1 represents the group of impurity objects derived from yarn fibers from bags of grated coconut pulp in the process of compressing coconut oil, (2) FiberT2 represents representing the group of impurity object derived from the fibers of the coconut pulp, (3) FiberT3 represents the fiber images of coconut coir fibers, (4) FiberT4 represents the fiber images of the inner shell of the coconut shell, (5) ParticleT1 represents the group of impurity object derived from coconut coir, (6) ParticleT2 represents the group of impurity object derived from coconut shell fragments, (7) ParticleT3 represents the group of impurity object from the coconut pulp, (8) ParticleT4 represents the group of impurity object derived from coconut shell dust, (9) Air, and (10) Tissue.

All images in these datasets are the images each type of objects in coconut oil with different motions of objects and different perspectives of the coconut oil images according to the actual conditions arising from each production cycle of coconut oil. As a result, the images from these datasets are similar to the images which used the data augmentation process, including horizontal flip, random shift, rotation, zoom, and brightness of the images.

Next, the data were separated into two parts: the first part was used for the training model, and the other part was used for the test model to determine the model performance after the training model was completed.

3.1. Datasets

From the data of 7861 images of adulterating object particles obtained from the previous process, the image data of the impurities were divided into 10 classes: FiberT1, FiberT2, FiberT3, FiberT4, ParticleT1, ParticleT2, ParticleT3, ParticleT4, Air, and Tissue, as shown in Table 1. The classes of the impurity images in each group were separated into two parts, which were used to create datasets to train the model and test its performance. Each dataset is described as follows.

The first dataset, “Particles in Coconut Oil_V1 (PiCO_V1)”, was used for model performance testing. The total of 1000 particle images was determined to consist of 135 air images, 186 FiberT1 images, 72 FiberT2 images, 111 FiberT3 images, 76 FiberT4 images, 63 ParticleT1 images, 91 ParticleT2 images, 64 ParticleT3 images, 86 ParticleT4 images, and 116 Tissue images, as shown in Table 1.

The second dataset, “Particles in Coconut Oil_V2 (PiCO_V2)”, was used for the training model. The total number of particle images was 6861, consisting of 1529 air images, 1107 FiberT1 images, 311 FiberT2 images, 398 FiberT3 images, 391 FiberT4 images, 516 ParticleT1 images, 661 ParticleT2 images, 483 ParticleT3 images, 898 ParticleT4 images, and 567 tissue images. Table 1 shows sample images of some particle objects.

The two datasets, PiCO_V1 and PiCO_V2, contain grayscale images that are scaled to meet the training model requirements. In the first step, we set the image resolution to 224×224 pixels for use with the standard CNN, MobileNetV2, ResNet50, ResNet101, DenseNet121, VGG16, and VGG19 architectures. In the second step, we set the image resolution to 299×299 pixels for use with the Xception, InceptionV3, and InceptionResNetV2 architectures, which were later used in the experimental section.

Table 1. 10 classes of particles in coconut oil.

Class	Example of Particle Image for Model Training								
Air									
FiberT1									
FiberT2									
FiberT3									
FiberT4									
ParticleT1									
ParticleT2									
ParticleT3									
ParticleT4									
Tissue									

3.2. Experimental Settings and Results

The CNN modeling experiments were conducted on a laptop with a 2.60 GHz Intel i7-9750H processor and 16 GB of main memory with a Jupyter Notebook and Python 3.7 kernel with TensorFlow, Keras, and OpenCV libraries. After the proposed model was developed, it was tested with the PiCO_V1 and PiCO_V2 datasets. The results were evaluated using a confusion matrix for the correctness measurement. The values in the confusion matrix were used to determine the efficiency of the classification model, which can be calculated from Equations (1)–(4).

$$\text{Accuracy} = \frac{TP + TN}{TP + FP + TN + FN} \quad (1)$$

$$\text{Precision} = \frac{TP}{TP + FP} \quad (2)$$

$$\text{Recall} = \frac{TP}{TP + FN} \quad (3)$$

$$\text{F1}_{\text{Score}} = 2 \times \left[\frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \right] \quad (4)$$

where: *TP* is true positive, which means that the predicted data match the actual data in the class under consideration; *FP* is false positive, which means that the predicted data are do not match the actual data in the class under consideration; *TN* is true negative, which means that the predicted data match the actual data in a class that is not considered; and *FN* is false negative, which means mispredicted data in an unconsidered class.

Equation (1) is used to determine accuracy, meaning the number of times the prediction data match the actual data and, combined with the prediction data, match the actual data in a class that is not considered. When that number is divided by the total number of data, the efficiency of accuracy prediction is obtained.

Precision value can be calculated from Equation (2), which means the number of times the prediction data match the actual data. Then, it is divided by the number of times the prediction data match the actual data plus the incorrect predictive data in the class under consideration. The precision value can also be called a positive predictive value (PPV).

The recall value, which means the number of times the prediction data match the actual data, can be calculated from Equation (3). Then, the number obtained is divided by the number of times that the predicted data match the actual data plus the wrong predicted data in the class that is not considered. Finally, the F1_{Score} value, meaning the value obtained by combining precision and recall values, can be calculated from Equation (4). It is the harmonic mean between the precision and recall values. These four values were used to determine the performance of all models of the CNN architectures. The test results were based on the PiCO_V1 dataset, which will be discussed in the next section.

For the training model, 10 standard CNNs were used. The preliminary testing of the model's accuracy began with the input of images from the PiCO_V2 dataset that had a resolution size that met the requirements of each standard CNN architecture. The process of training the model started by importing one image at a time until every image of the PiCO_V2 dataset had been imported. The experiment started with the MobileNetV2 architecture, and the results were then compared with those of other CNNs.

3.2.1. Experiments with MobileNetV2

In the first part of our experiment, MobileNetV2 was performed to test the differences between two hyperparameters: width multipliers and fixed resolution. The MobileNetV2 performance obtained from the PiCO_V2 dataset showed that the width multiplier values affected the accuracy and size of the resulting models, as is clearly illustrated in Table 2. There is a relation between the width and the size of the model. When the width multipliers are reduced in value, the size of the model is also reduced. Because the size of the model is smaller, the accuracy performance continuously decreases from 94.05% to only 71.54%.

Table 2. Performance of MobileNets on different width multipliers (fixed resolution = 224).

Width Multipliers	Accuracy (%)	Size Model (Mb)
1.00	94.05	12.52
0.75	88.72	8.38
0.50	71.54	5.60
0.25	74.25	3.88

A test was then conducted to differentiate one hyperparameter from another, in terms of resolution, to see how the findings differed from those of the prior test. In this experiment, the best results were obtained from the previous step with a resolution of 244×244 pixels,

which was set to a width multiplier value of 1.0 as a constant throughout the testing process. Moreover, different image resolution values were used. Initially, the test was performed with the same image resolution size of 224×224 pixels. The image resolution was scaled down three times in the PiCO_V2 dataset, to 192×192 , 160×160 , and 128×128 , and retrained to a new training model. The results of the image resolution scaling test are shown in Table 3. Scaling the resolution had little effect on the accuracy performance. The accuracy ranged from 94.05% to 91.22%, and changes in the resolution scaling had no effect on the model size.

Table 3. Performance of MobileNets on different resolution multipliers (fixed width = 1.0).

Resolution	Accuracy (%)	Size Model (Mb)
224×224	94.05	12.52
192×192	93.40	12.52
160×160	92.56	12.52
128×128	91.22	12.52

The hyperparameters affecting the experiments on MobilenetV2 from Tables 2 and 3 were combined, and the results are presented in Table 4. The resolution of the image at 224×224 pixels and a width multiplier value of 1.0 were optimized for the MobileNetV2 network layer for the PiCO_V2 dataset.

Table 4. Accuracy of MobileNets on every combination of resolution and width multipliers.

Resolution	Width Multipliers			
	1.00	0.75	0.50	0.25
224×224	94.05	88.72	71.54	74.25
192×192	93.40	88.77	72.35	74.44
160×160	92.56	88.21	71.23	72.87
128×128	91.22	87.83	70.96	73.23

The performance of the MobileNetV2 model was then investigated using the 1000 images PiCO_V1 dataset, the results of which were already known. First, the images were exported into MobileNetV2 for testing. The confusion matrix values were used to describe the model's accuracy performance. The test results are shown in Table 5.

Table 5. Confusion matrix of the MobileNetV2 architecture.

Predicted Class (Accuracy)											
Actual Class	Class	Air	FiberT1	FiberT2	FiberT3	FiberT4	ParticleT1	ParticleT2	ParticleT3	ParticleT4	Tissue
	Air	0.81	0.02	0.02	0.00	0.03	0.00	0.07	0.01	0.04	0.00
	FiberT1	0.02	0.83	0.05	0.01	0.01	0.03	0.03	0.00	0.02	0.00
	FiberT2	0.00	0.01	0.92	0.00	0.00	0.01	0.04	0.00	0.01	0.00
	FiberT3	0.04	0.09	0.03	0.72	0.04	0.01	0.05	0.00	0.02	0.01
	FiberT4	0.01	0.12	0.05	0.04	0.62	0.04	0.08	0.00	0.04	0.00
	ParticleT1	0.05	0.03	0.00	0.00	0.00	0.73	0.08	0.03	0.03	0.05
	ParticleT2	0.03	0.03	0.02	0.00	0.00	0.00	0.87	0.04	0.00	0.00
	ParticleT3	0.03	0.02	0.02	0.00	0.00	0.00	0.09	0.78	0.00	0.06
	ParticleT4	0.02	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.98	0.00
	Tissue	0.05	0.04	0.02	0.01	0.00	0.01	0.11	0.02	0.00	0.74

According to the prediction results, MobileNetV2 had the highest number of errors in the FiberT1 class, with 32 false predictions out of 186 images, followed by the tissue class, with 30 prediction errors out of 116 images. The errors occur because the shape of the object is similar to the image. The results in the confusion matrix were used to calculate the efficiency of the accuracy of the image classification of the PiCO_V1 dataset using Equations (1)–(4). The results indicated that the accuracy, precision, recall, and $F1_{Score}$ were 80.20%, 71.61%, 81.15%, and 76.08%, respectively.

3.2.2. Comparison of the MobileNet and Other CNN Architectures

In this section of the experiments, we compared the results of MobileNet with those of other CNN architectures based on the best width multiplier and resolution parameters obtained from the previous experiment with the MobileNetV2 architecture. The resolution was set to 224×224 pixels, and the width multiplier was 1.0. We tested two types of imported data: (1) data without resizing the resolution, which was 224×224 ; the standard CNN, VGG16, VGG19, ResNet50, ResNet101, and DenseNet121 were tested; and (2) the data with a scalable resolution of 299×299 based on the input model specification; Xception, InceptionV3, and InceptionResNetV2 were tested. We trained the entire model on the PiCO_V2 dataset with 15 epochs. The results are shown in Table 6.

Table 6. Evaluation of the runtime performances of CNN networks on our dataset at epochs = 15.

No	Model	Input Resolution	Accuracy (%) Model	Training Time (Hour)	Test Time (Minute)	Size Model (Mb)
1	MobileNetV2	$224 \times 224 \times 3$	94.05	5.13	18.20	12.52
2	Standard CNN	$224 \times 224 \times 3$	89.70	1.17	18.35	8.27
3	Xception	$299 \times 299 \times 3$	92.82	30.08	22.44	842.33
4	InceptionV3	$299 \times 299 \times 3$	93.23	15.38	20.22	96.88
5	ResNet50	$224 \times 224 \times 3$	91.65	16.04	19.27	689.54
6	ResNet101	$224 \times 224 \times 3$	90.21	25.42	19.37	795.23
7	DenseNet121	$224 \times 224 \times 3$	93.53	21.47	20.10	827.27
8	VGG16	$224 \times 224 \times 3$	91.95	8.39	18.56	689.41
9	VGG19	$224 \times 224 \times 3$	91.12	12.19	19.55	722.58
10	InceptionResNetV2	$299 \times 299 \times 3$	90.93	30.25	20.37	956.36

The results shown in Table 6 indicate that MobileNetV2 significantly outperformed all CNN models. The accuracy of MobileNetV2 was 94.05%, followed by DenseNet121, InceptionV3, Xception, VGG16, ResNet50, VGG19, InceptionResNetV2, ResNet101, and Standard CNN with accuracies of 93.53%, 93.23%, 92.82%, 91.95%, 91.65%, 91.12%, 90.93%, 90.21%, and 89.70%, respectively. The results also show that MobileNetV2 outperformed the standard CNN by approximately 4.35%.

In terms of the model size, MobileNetV2 was still the smallest model architecture. The total size of the MobileNetV2 network was only 12.52 Mb, making it approximately 6.6 and 7.7 times smaller than DenseNet121 and InceptionV3, respectively. In terms of network size, VGG16 outperformed VGG19. VGG16 and VGG19, on the other hand, required enormous network sizes. The same problem occurred for the DenseNet, Inception, Xception, and ResNet networks with larger model sizes, and the accuracy was less than that of MobileNetV2. As a result, those models were not used in our research to solve the problem.

Table 6 illustrates the time it took to train all models at 15 epochs. The model that took the least amount of time in training was the standard CNN approach, which took 1.17 h. MobileNetV2, VGG16, VGG19, InceptionV3, ResNet50, DenseNet121, ResNet101, Xception, and InceptionResNetV2 took approximately 5.13, 8.39, 12.19, 15.38, 16.04, 21.47, 25.42, 30.08, and 30.25 h, respectively. The results are shown in Table 6. MobileNetV2 outperformed the other models and performed well in recognizing and classifying impurities in coconut oil.

3.2.3. Experiments with Data Augmentation

In this experiment, we compared the MobileNetV2 model's performance to the performance of each of the other models using the confusion matrix. The accuracy, precision, recall, and $F1_{\text{Score}}$ can be determined using Equations (1)–(4). The models used the PiCO_V1 dataset with 1000 images, and the test results are shown in Table 7.

Table 7. Results of model accuracy from the confusion matrix of the PiCO_V1 dataset.

No	Model	Image Input	Test Model with Confusion Matrix Form PiCO_V1 Dataset (%)			
			Accuracy	Precision	Recall	$F1_{\text{Score}}$
1	MobileNetV2	$224 \times 224 \times 3$	80.20	71.61	81.15	76.08
2	Standard CNN	$224 \times 224 \times 3$	54.82	45.09	61.01	51.86
3	Xception	$299 \times 299 \times 3$	72.95	64.62	76.24	69.95
4	InceptionV3	$299 \times 299 \times 3$	75.13	68.19	77.14	72.39
5	ResNet50	$224 \times 224 \times 3$	68.88	62.12	72.61	66.95
6	ResNet101	$224 \times 224 \times 3$	66.44	66.56	67.29	66.92
7	DenseNet121	$224 \times 224 \times 3$	68.55	62.75	69.38	65.90
8	VGG16	$224 \times 224 \times 3$	68.52	60.55	73.02	66.20
9	VGG19	$224 \times 224 \times 3$	70.02	69.38	71.89	68.58
10	InceptionResNetV2	$299 \times 299 \times 3$	55.55	55.25	58.79	55.44

In the first experiment, we measured the standard CNN, and the accuracy, precision, recall, and $F1_{\text{Score}}$ were 54.82%, 45.09%, 61.01%, and 51.86%, respectively. The results of the accuracy testing showed that the errors in this model were due largely to incorrect selections. The model selected ParticleT4 instead of FiberT3, and ParticleT4 was selected instead of FiberT4, which represented a large number of incorrect selections. There were 452 misclassified images, or a 45.18% error rate.

In the second experiment, we measured the Xception model, and the accuracy, precision, recall, and $F1_{\text{Score}}$ were 72.95%, 64.62%, 76.24%, and 69.95%, respectively. Errors in the model occurred when images were classified as FiberT1 instead of FiberT3. In this case, 52 images were mistakenly selected. Another mistake occurred when images were classified as FiberT2 and ParticleT1 instead of FiberT1. There were 12 incorrect images from FiberT2 and 19 incorrect images from ParticleT1, resulting in 271 misclassified images, or a 27.05% error rate.

In the third experiment, we measured the InceptionV3 model, and the accuracy, precision, recall, and $F1_{\text{Score}}$ were 75.13%, 68.19%, 77.14%, and 72.39%, respectively. Errors in the model occurred when images were classified in the incorrect fiber class. The InceptionV3 model incorrectly predicted 53 images in the FiberT1 class that should actually have been in the FiberT3 class. Furthermore, the model predicted that 59 images belonged to FiberT2; however, the validation revealed that FiberT1 was the proper class. In the air, particle, and tissue classes, a total of 249 images were misclassified for a 24.87% error rate.

In the fourth experiment, we measured the ResNet50 model, and the accuracy, precision, recall, and $F1_{\text{Score}}$ were 68.88%, 62.12%, 72.61%, and 66.95%, respectively. The most misclassified images were those from FiberT1. The model determined that 31 and 18 images should be classified as FiberT2 and ParticleT2, respectively. A total of 312 images were misclassified for a 31.12% error rate.

In the fifth experiment, we measured the ResNet101 model, and the accuracy, precision, recall, and $F1_{\text{Score}}$ were 66.44%, 66.56%, 67.29%, and 66.92%, respectively. The most misclassified class was FiberT3. The model determined that 20 and 16 images should be FiberT1 and ParticleT4, respectively. A total of 336 images were misclassified for a 33.44% error rate.

In the sixth experiment, we measured the DenseNet121 model, and the accuracy, precision, recall, and $F1_{\text{Score}}$ were 68.55%, 62.75%, 69.38%, and 65.90%, respectively. The

most misclassified class was FiberT2. The model determined that 26 images should be FiberT3. A total of 315 images were misclassified for a 31.45% error rate.

In the seventh experiment, we measured the VGG16 model, and the accuracy, precision, recall, and $F1_{\text{Score}}$ were 68.52%, 60.54%, 73.02%, and 66.20%, respectively. The most misclassified class was FiberT3. The model determined that 15 images should be FiberT1. A total of 315 images were misclassified for a 31.48% error rate.

In the eighth experiment, we measured the VGG19 model, and the accuracy, precision, recall, and $F1_{\text{Score}}$ were 70.02%, 69.38%, 71.89%, and 68.58%, respectively. The most misclassified class was FiberT3. The model determined that 20 images should be FiberT1. A total of 299 images were misclassified, representing an error rate of 29.98%.

Finally, we examined the InceptionResNetV2 model. The accuracy, precision, recall, and $F1_{\text{Score}}$ were 55.55%, 55.25%, 58.79%, and 55.44%, respectively. The most misclassified class was FiberT3. The model determined that 16 and 110 images should be FiberT2 and ParticleT4, respectively. A total of 445 images were misclassified, representing an error rate of 44.5%. The results of the confusion matrix for each model are shown in Table 7.

3.3. Discussion

The overall results of this research demonstrate the high efficiency of the MobileNetV2 architecture in recognizing classes of impurities contained in coconut oil. MobileNetV2 spent the second-lowest amount of time training the model on both datasets, PiCO_V1 and PiCO_V2, while the standard CNN consumes the shortest time. By comparing the accuracy with that of each CNN model, the models were trained from the beginning without any preconfigured parameters, and only 15 training epochs were used for speed comparison. This problem may affect accuracy values in the future because the number of epochs affects the model performance. If the number of epochs is too small, it can lead to incorrect grouping errors and a lack of resemblance to the actual answer, a condition known as underfitting. In contrast, if the number of epochs is too high, it can cause problems in training the model. Then, even though the model can precisely classify objects, it is impractical to use, a problem that is called overfitting. This problem can be solved by adjusting the parameter dropout, so that the data is distributed among each class. To obtain a good compromise, the parameter adjustments should be made accordingly.

4. Conclusions

In this paper, experiments were conducted using a deep learning model to solve the problem of classifying adulterating objects in coconut oil. We selected ten CNN models, MobileNetV2, Standard CNN, Xception, InceptionV3, ResNet50, ResNet101, DenseNet121, VGG16, VGG19, and InceptionResNetV2, to benchmark the performance of their architectures. The experimental results indicated that MobileNetV2 was the best method for recognizing adulterating objects in coconut oil. It had the best accuracy compared to other architectures. The FiberT3 test results were the worst in the classification due to the similarity in the shape of FiberT3 objects to those of FiberT4 and FiberT2 objects. As a result of this classification, the performance values were reduced. Moreover, the MobileNetV2 architecture is suitable for all platforms, including mobile devices with low processing speed and low memory.

When MobileNetV2 was tested against the PiCO_V1 dataset, which is a previously existing dataset, changing the width and resolution parameters had no effect on the model accuracy. However, such changes can greatly reduce the size of the model. The MobileNetV2 architecture has a model size of 12.54 Mb., which is smaller than that of other models. Additionally, MobileNetV2 achieved an accuracy of 80.20% in testing on the PiCO_V1 dataset, which was better than the results of the other CNN models. Finally, we utilized MobileNetV2 with the coconut oil impurity detection program for further integration with the coconut oil production line.

Precaution for working in coconut oil images is the heat of the lamp used to shine through the coconut oil. If the lamp is overheating, this causes the light from the lamp

to flicker. As a result, the coconut oil images are unevenly bright and unable to show the objects contained in the coconut oil. Therefore, cooling of the lamp is a necessary factor to detect impurities in the coconut oil.

Author Contributions: Conceptualization, A.P. and W.K.; Methodology, A.P. and W.K.; Software, A.P.; Validation, A.P. and W.K.; Formal analysis, A.P. and W.K.; Investigation, A.P.; Resources, A.P.; Data curation, A.P.; Writing—original draft preparation, A.P.; Writing—review and editing, A.P. and W.K.; Visualization, A.P.; Supervision, W.K.; Project administration, W.K. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by School of Science, King Mongkut's Institute of Technology Ladkrabang, Thailand.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Acknowledgments: The authors would like to thank Tropicana Oil Co., Ltd. for providing resources and supported in testing part.

Conflicts of Interest: The authors declare no potential conflict of interest.

References

1. Karnawat, V.; Patil, S. Turbidity detection using image processing. In Proceedings of the 2016 International Conference on Computing, Communication and Automation (ICCCA), Greater Noida, India, 29–30 April 2016.
2. Berg, M.; Videen, G. Digital holographic imaging of aerosol particles in flight. *J. Quant. Spectrosc. Radiat. Transf.* **2011**, *112*, 1776–1783. [\[CrossRef\]](#)
3. Pastore, V.P.; Zimmerman, T.; Biswas, S.; Bianco, S. Annotation-free learning of plankton for classification and anomaly detection. *Sci. Rep.* **2020**, *10*, 12142. [\[CrossRef\]](#) [\[PubMed\]](#)
4. Versaci, M.; Calcagno, S.; Jia, Y.; Morabito, F.C. Fuzzy Geometrical Approach Based on Unit Hyper-Cubes for Image Contrast Enhancement. In Proceedings of the 2015 IEEE International Conference on Signal and Image Processing Applications (ICSIPA), Kuala Lumpur, Malaysia, 19–21 October 2015.
5. amirberAgain, Python and openCV to Analyze Microscope Slide Images of Airborne Particles. Available online: <https://publiclab.org/notes/amirberAgain/01-12-2018/python-and-opencv-to-analyze-microscope-slide-images-of-airborne-particles> (accessed on 25 May 2021).
6. Oheka, O.; Chunling, T. Fast and Improved Real-Time Vehicle Anti-Tracking System. *Appl. Sci.* **2020**, *10*, 5928. [\[CrossRef\]](#)
7. Dechter, R. Learning while searching in constraint-satisfaction-problems. In Proceedings of the Fifth AAAI National Conference on Artificial Intelligence (AAI'86), Philadelphia, PA, USA, 11–15 August 1986.
8. Schmidhuber, J. Deep learning in neural networks: An overview. *Neural Netw.* **2015**, *61*, 85–117. [\[CrossRef\]](#)
9. Fukushima, K. Recent advances in the deep CNN neocognitron. *Nonlinear Theory Appl. IEICE* **2019**, *10*, 304–321. [\[CrossRef\]](#)
10. LeCun, Y. 1.1 Deep Learning Hardware: Past, Present, and Future. In Proceedings of the 2019 IEEE International Solid-State Circuits Conference—(ISSCC), San Francisco, CA, USA, 17–21 February 2019.
11. Li, X.; Xiong, H.; An, H.; Xu, C.; Dou, D. RIFLE: Backpropagation in Depth for Deep Transfer Learning through Re-Initializing the Fully-connected Layer. In Proceedings of the 37th International Conference on Machine Learning (ICML2020), Vienna, Austria, 12–18 July 2020.
12. Ranzato, M.A.; Huang, F.; Boureau, Y.; LeCun, Y. Unsupervised learning of invariant feature hierarchies with applications to object recognition. In Proceedings of the 2007 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Minneapolis, MN, USA, 17–22 June 2007.
13. Brinker, T.J.; Hekler, A.; Utikal, J.; Grabe, N.; Schädendorf, D.; Klode, J.; Berking, C.; Steeb, T.; Enk, A.; Kalle, C.V. Skin Cancer Classification Using Convolutional Neural Networks: Systematic Review. *J. Med. Internet Res.* **2018**, *20*, e11936. [\[CrossRef\]](#)
14. Patil, R.; Bellary, S. Machine learning approach in melanoma cancer stage detection. *J. King Saud Univ. Comput. Inf. Sci.* **2020**, in press. [\[CrossRef\]](#)
15. Khairi, M.T.M.; Ibrahim, S.; Yunus, M.A.M.; Faramarzi, M.; Yusuf, Z. Artificial Neural Network Approach for Predicting the Water Turbidity Level Using Optical Tomography. *Arab. J. Sci. Eng.* **2016**, *41*, 3369–3379. [\[CrossRef\]](#)
16. Newby, J.M.; Schaefer, A.M.; Lee, P.T.; Forest, M.G.; Lai, S.K. Convolutional neural networks automate detection for tracking of submicron-scale particles in 2D and 3D. *Proc. Natl. Acad. Sci. USA* **2018**, *15*, 9026–9031. [\[CrossRef\]](#) [\[PubMed\]](#)
17. Rong, D.; Xie, L.; Ying, Y. Computer vision detection of foreign objects in walnuts using deep learning. *Comput. Electron. Agric.* **2019**, *162*, 1001–1010. [\[CrossRef\]](#)
18. Ferreira, A.D.S.; Freitas, D.M.; Silva, G.G.D.; Pistori, H.; Folhes, M.T. Weed detection in soybean crops using ConvNets. *Comput. Electron. Agric.* **2017**, *143*, 314–324. [\[CrossRef\]](#)

19. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. ImageNet classification with deep convolutional neural networks. *Commun. ACM* **2017**, *60*, 84–90. [[CrossRef](#)]
20. Simonyan, K.; Vedaldi, A.; Zisserman, A. Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps. *arXiv* **2014**, arXiv:1312.6034.
21. Simonyan, K.; Zisserman, A. Very Deep Convolutional Networks for Large-Scale Image Recognition. In Proceedings of the 2015 International Conference on Learning Representations (ICLR 2015). *arXiv* **2014**, arXiv:1409.1556.
22. Howard, A.G.; Zhu, M.; Chen, B.; Kalenichenko, D.; Wang, W.; Weyand, T.; Andreetto, M.; Adam, H. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. *arXiv* **2017**, arXiv:1704.04861.
23. Pan, H.; Pang, Z.; Wang, Y.; Wang, Y.; Chen, L. A New Image Recognition and Classification Method Combining Transfer Learning Algorithm and MobileNet Model for Welding Defects. *IEEE Access* **2020**, *8*, 119951–119960. [[CrossRef](#)]
24. Kerf, T.D.; Gladines, J.; Sels, S.; Vanlanduit, S. Oil Spill Detection Using Machine Learning and Infrared Images. *Remote Sens.* **2020**, *12*, 4090. [[CrossRef](#)]
25. Iqbal, U.; Barthelemy, J.; Li, W.; Perez, P. Automating Visual Blockage Classification of Culverts with Deep Learning. *Appl. Sci.* **2021**, *11*, 7561. [[CrossRef](#)]
26. Tammina, S. Transfer learning using VGG-16 with Deep Convolutional Neural Network for Classifying Images. *Int. J. Sci. Res. Publ.* **2019**, *9*, 143–150. [[CrossRef](#)]
27. Szegedy, C.; Liu, W.; Jia, Y.; Sermanet, P.; Reed, S.E.; Anguelov, D.; Erhan, D.; Vanhoucke, V.; Rabinovich, A. Going deeper with convolutions. In Proceedings of the 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Boston, MA, USA, 7–12 June 2015.
28. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016.
29. Aparna; Bhatia, Y.; Rai, R.; Gupta, V.; Aggarwal, N.; Akula, A. Convolutional neural networks based potholes detection using thermal imaging. *J. King Saud Univ. Comput. Inf. Sci.* **2019**, *in press*. [[CrossRef](#)]
30. Huang, G.; Liu, Z.; Pleiss, G.; Maaten, L.V.; Weinberger, K.Q. Convolutional Networks with Dense Connectivity. *IEEE Trans. Pattern. Anal. Mach. Intell.* **2019**, 1–12. [[CrossRef](#)] [[PubMed](#)]
31. Tao, Y.; Xu, M.; Lu, Z.; Zhong, Y. DenseNet-Based Depth-Width Double Reinforced Deep Learning Neural Network for High-Resolution Remote Sensing Image Per-Pixel Classification. *Remote. Sens.* **2018**, *10*, 779. [[CrossRef](#)]
32. Too, E.C.; Yujian, L.; Njuki, S.; Yingchun, L. A comparative study of fine-tuning deep learning models for plant disease identification. *Comput. Electron. Agric.* **2019**, *161*, 272–279. [[CrossRef](#)]
33. Palananda, A.; Kimpan, W. Turbidity of Coconut Oil Determination Using the MAMoH Method in Image Processing. *IEEE Access* **2021**, *9*, 41494–41505. [[CrossRef](#)]