Fruit Classification using Colorized Depth Images

Dhong Fhel K. Gom-os
Department of Computer Science
University of the Philippines Cebu
Cebu, Philippines 6000

Abstract—**Fruit classification is a computer vision task that aims to classify fruit classes correctly, given an image. Nearly all fruit classification studies have used RGB color images as inputs, a few have used costly hyperspectral images, and a few classical ML-based have used colorized depth images. Depth images have apparent benefits such as invariance to lighting, less storage requirement, better foreground-background separation, and more pronounced curvature details and object edge discontinuities. However, the use of depth images in CNN-based fruit classification remains unexplored. The purpose of this study is to investigate the use of colorized depth images in fruit classification with four CNN models, namely, AlexNet, GoogleNet, ResNet101, and VGG16, and compare their performance and computational efficiency, as well as the impact of transfer learning. Depth images of apple, orange, mango, banana and rambutan (*Nephelium Lappaceum*) were manually collected using a depth sensor with sub-millimeter accuracy and subjected to jet, uniform, and inverse colorization to produce three sets of dataset. Results show that depth images can be used to train CNN models for fruit classification with ResNet101 achieving the best accuracy of 96% on the inverse dataset. It achieved 100% accuracy after transfer learning. GoogleNet showed the most significant improvement after transfer learning on the uniform dataset, at 12.27%. It also exhibited the lowest training and inference times. The results show the potential use of depth images for fruit classification and similar computer vision tasks.**

Keywords*—*Fruit classification; depth image; depth colorization; CNN; transfer learning

I. Introduction

The depth output of a depth sensor is converted into a three-dimensional RGB image to provide a colorized depth image. To create colorized images, specific colorization procedures are used to the depth data. A colorized depth image stores the depth information per pixel as opposed to a color image, which typically stores red, green, and blue intensity values per pixel. As a result, in a colorized depth image, the intensity of each pixel indicates how far an object is from the camera. Depth image is also known as range image.

The use of colorized depth images has continuously gained attention in the research community. They are used in soybean canopy analysis through 3D point clouds [1], action recognition [2], human posture analysis [3], 3D semantic segmentation [4], object recognition [5], and kangaroo detection [6]. In these studies, depth images were used either exclusively or in combination with RGB through fusion. These studies have shown that colorized depth images are helpful for solving complex computer vision problems.

Certain colorized depth-image characteristics are beneficial for both simple and complex computer-vision problems. For example, depth images are good at separating 3D objects from the horizontal plane [4], which is beneficial for object detection. They can provide an outline of the strong discontinuities at the edges of an object, which is advantageous for object classification [5]. Curvature information is also more prevalent in depth images than in color images [5]. In addition, they require less storage than colored images (see Table II). They are more useful in edge cases of machine learning problems [7], such as differentiating between a hotdog food and a hotdog balloon. They are also invariant to extreme variations in lighting conditions and scale [7]. However, it is widely known that depth images require a lot of pre-processing as opposed to color images because of their tendency to contain missing depths [5], as well as their low contrast property [6]. Additional processing is required to improve the contrast and accuracy in highly complex and wide-area applications.

For less complex and more constrained applications such as fruit image classification, the benefits of using depth images can be leveraged. Fruit image classification is a computer vision task in which fruit images are classified according to their class. Here, it is assumed that the images do not contain more than one class of fruit. Applications for fruit image classification include supermarket self-checkouts and fruit sorting in factories.

Several studies on fruit image classification have been conducted. However, most of these studies used color images as inputs. Only a few studies have explored the use of depth images for fruit classification. In particular, one study [8] trained six machine learning algorithms, including Sequential Minimum Optimization (SMO), k-nearest neighbors (KNN), bagging based on REPTree, Decision Trees (DT), and Random Forests (RF) in the Waikato Environment for Knowledge Analysis (Weka) using visual features from color images and object shape representations from depth images. The shape descriptors extracted from the depth images include compactness, symmetry, local convexity, smoothness, and image moments. The results showed that RF trained on a combination of scalable color and edge histogram descriptors yielded the best performance at 99% accuracy. The problem with this classical approach is the need to perform segmentation in color images and manually extract features from the depth images. Depth images cannot be processed without their corresponding color images.

Another study [9] used depth images to render a 3D point cloud of fruits for classification. Similar to [8], [9] developed a multi-feature classification framework utilizing both color and depth images. It uses a color layout descriptor, viewpoint feature histogram, and point feature histogram as descriptors in the classification problem. Similarly, this approach is labor intensive and requires manual extraction of features from images.

Based on [8] and [9], depth images can be beneficial for fruit-image classification. However, no study has explored the use of depth images yet, particularly in CNN-based fruit classification. Therefore, the goal of this study is to explore the use of purely depth images based on simple colorization techniques in fruit classification using CNN. The researcher used three different types of depth images, namely color-jet, uniform, and inverse hue colorization, and compared the performance of each type in four CNN models, namely, AlexNet, GoogleNet, RestNet101 and VGG16. The effect of transfer learning on the type of depth image with the lowest error rate in each CNN model was also investigated. Because depth sensors are gaining popularity in the sensor market, it would be beneficial to explore the use of depth images in computer vision problems, particularly in fruit classification.

The remainder of this paper is organized as follows. After the introduction, Section II discusses the background of the study. Section III discusses the methodology of the study. The findings are presented in Section IV followed by future work and conclusions in Sections V and VI, respectively.

II. Background

A. Fruit Classification

Fruit image classification is the process of identifying a specific type of fruit in an image. This task is typically performed using convolutional neural networks (CNNs), a type of deep learning model that has become dominant in various computer vision tasks. CNNs are trained on large datasets of labeled images and learn to recognize features that are relevant to the task of fruit classification. The trained model can then be used to classify new images of fruits based on learned features.

Almost all studies tackling fruit classification using CNN use RGB color images as their dataset, except for a few that use hyper-spectral images. Hyperspectral images are captured using expensive hyper-spectral imaging which is a technique that collects and processes information from across the electromagnetic spectrum to obtain the spectrum for each pixel in an image of a scene. Table I shows a summary of the dataset types used in training the CNN models for fruit classification from 2015. [10] summarized the CNN-based fruit classification studies conducted from 2015 to 2020. This summary was manually checked by the researcher, and the outcome was plotted in the table mentioned above. For 2021-2023, the researcher manually searched the Scopus database using the keyword "fruit classification" for relevant papers. As shown in Table I, there is only one paper [11] that used the other type of dataset, i.e. hyper-spectral, from 2015-2020 and another one [12] in 2022. Evidently, the research community on fruit classification has extensively used RGB color images, and has not substantially explored other image types, including depth images.

The most popular benchmarks used for fruit classification are ImageNet, VegFru [13] and Fruit 360 [14]. ImageNet is a large visual database designed for use in visual object recognition software. It contains over 14 million images that have been hand-annotated to indicate what objects are pictured, and bounding boxes are provided in at least one million images. VegFru is a domain-specific dataset for fine-grained visual categorization of vegetables and fruits based on their eating characteristics. Each image in the dataset contained at least one edible part of vegetables or fruits with the same cooking usage, and all images were labeled hierarchically. It is closely related to the daily lives of people and is aimed at domestic cooking and food management. Fruit 360 is a dataset of images containing fruits. It is a high-quality dataset that includes 131 fruits and vegetables. The images are color (RGB) and 100 pixels × 100 pixels in size, with three values for each pixel. It contains a total of 90,380 images, with 67,692 images in the training dataset and 22,688 images in the test dataset.

TABLE I. Number of CNN-based Fruit Classification Studies per Data Type

Year	Color (RGB)	Other (Hyper-spectral)
2015-2020	20	1
2021	27	0
2022	29	1
2023*	5	0

Similar to other computer vision tasks, fruit classification tasks extensively use color images, perhaps because of ubiquitous color sensors. However, RGB images obtained from color sensors have issues when used in fruit classification. There is a high rate of misclassification of fruits that are of similar colors, such as avocado and watermelon, banana and papaya, orange and carrot, as shown in [15]; passion fruit and blackberries, red grapes and passion fruits in [16]; and peach and apple red, pear and apple green, and pomegranate and apple in [14]. It was also found in [17] that shape feature also resulted in high misclassification between apples and oranges. A similar result was found in [18] which suggested that the color feature alone does not provide a good classification outcome. One very recent study [19] used MobileNetV2 with attention module in the classification. The attention module worked well in non-smoothed fruits but provided low precision in smoothed fruits like orange (at 81.75%). The researcher argues that it might be beneficial to explore other types of images for fruit classification, one that is independent of color. This is especially because fruit classification task is constrained and not complex, where special imaging sensors such as a depth sensor can easily be set up.

B. Depth Sensing

Depth sensing refers to the process of measuring the distance between a device and an object. Depth-sensing cameras are used for this purpose, and they automatically detect the presence and measure the distance of an object within its field of view. There are three types of depth-sensing cameras based on their method of calculating depth: (a) structured light and coded light, (b) stereo depth, and (c) time of flight and LIDAR [20].

(a) is a type of technology that uses projected light, usually infrared light, in the scene for a sensor to obtain its pattern and estimate its depth. This type of technology is the best indoors and within a short range. However, it is vulnerable to interference from nearby devices that emit infrared light. As opposed to (a), which uses projected light, (b) uses any light to

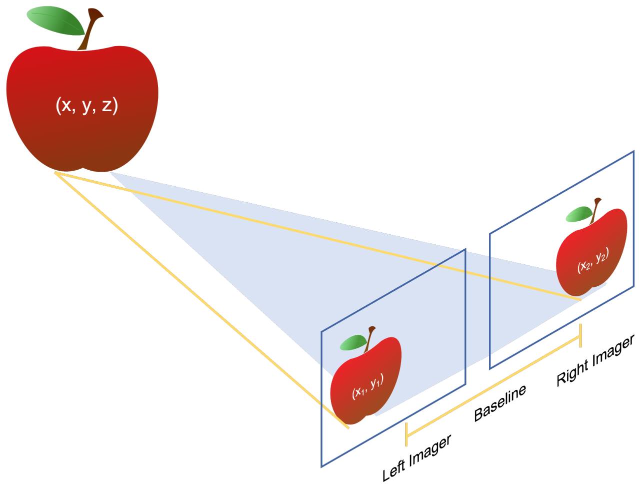

Fig. 1. Basic principle of stereo vision.

estimate depth but uses two sensors at a small distance apart. This technology works well both indoors and outdoor. Fig. 1 shows the basic operation of the stereo vision system. Similar to (a), (c) also emits light in the scene and calculates the time it returns to the sensor by which the depth is computed. As in (a), it is vulnerable to interference and is not ideal for outdoor conditions. The most common depth sensor uses a stereo vision mechanism for depth sensing. The depth sensor used in this study was a stereo-vision camera.

C. Depth Colorization

Depth colorization is a subset of image colorization [21] which is the process of estimating RGB colors for grayscale images to enhance perceptual quality. In the context of depth images, a grayscale image is a 2D depth map, where each pixel contains depth measurements from the depth sensor. Depth colorization is a method that adds colors to a depth map. This involves compression and coding [22]. It is not yet as developed as in compression and encoding in color images, but the ultimate goal is essentially the same: efficient storage, reduced artifacts, and reduced system bandwidth. To achieve this goal, various approaches have been developed. [23] suggested that there are two primary categories of representing colorized depth images, namely, hand-crafted depth colorization and ML-based depth colorization.

Some hand-crafted approaches include depth-to-surface normals [5], geocentric embedding (a.k.a. HHA encoding) [24], rendered mesh [25], quadtree decomposition & plane approximation [26], color-jet [27], and uniform and inverse colorization [22].

The surface normals [5] approach uses two cross-multiplied orthogonal tangent vectors and is normalized using the Euclidean norm, but introduces a recursive median filter to estimate the missing depth values and a bilateral filter to reduce noise. The geometric embedding approach encodes the height above ground and the angle with gravity for each pixel on top of the horizontal disparity [24]. The rendered mesh approach [25] first performs tabletop segmentation to extract the relevant depth map, missing depth values are filled in, the mesh is extracted from the point cloud, and the mesh is re-projected to a canonical camera pose. All the three approaches are computationally expensive. Despite this, they only result in minimal to no improvement in some benchmarks for object recognition tasks. The quadtree decomposition and plane approximation approaches [26] achieved a low bit rate. However, this approach requires proprietary software for encoding and decoding and does not take advantage of the hardware acceleration modules present in modern computers.

A more advanced colorization technique, ML-based depth colorization, was developed in [23]. It uses a CNN architecture that is pre-trained on ImageNet. However, despite the use of neural networks, the results show that the model did not significantly improve the classification accuracy compared to color-jet and surface normals [27]. In fact, it performs worse in some models and benchmarks. It can be deduced from [23] that the encoding used does not significantly impact performance and high accuracy in computer vision tasks is still achievable even with a simple colorization approach.

It is for this reason that simple colorization approaches have been adopted in this study, namely, color-jet [27], uniform and inverse colorization [22].

1) Color-Jet: This is a common approach in depth colorization [27] where depth data is applied with a jet colormap to transform it from a single channel to a three-channel 2D depth image. This approach was found to be effective and computationally inexpensive and was shown to outperform HHA for object recognition. In this approach, the depth values are first normalized between 0 and 255, and then a jet color map is applied to the one-channel image to make it a three-channel image. A jet is a colormap [28] used for data visualization. It is a rainbow map that is commonly used to create false color images. Typically, the depth image is derived for each pixel (i, j) by mapping the distance to color values ranging from red (near) over green to blue (far). Sometimes, this mapping is reversed, i.e., blue is near, which is the case in this study.

2) Uniform and Inverse Colorization: [22] also developed a similar approach to [27] which uses the Hue colorspace with 6 gradations and 1529 discrete levels. It has two variations: uniform and inverse. The former directly encodes the depth value whereas the latter encodes the disparity value (reciprocal of the depth value). The latter is suitable for closer distances because it can capture finer details and information. The equations below show the mapping between the normalized depth (d_n) to the Red (p_r), Green (p_g) and Blue (p_b) channels respectively in case of uniform colorization.

$$d_n = \frac{d - d_{min}}{d_{max} - d_{min}}$$

$$p_r = \begin{cases} 255, & 0 \leq d_n \leq 255 \cup 1275 < d_n \leq 1529 \\ 255 - d_n, & 255 < d_n \leq 510 \\ 0, & 510 < d_n \leq 1020 \\ d_n - 1020, & 1020 < d_n 1275 \end{cases}$$

$$p_g = \begin{cases} d_n, & 0 < d_n \leq 255 \\ 255, & 255 < d_n \leq 510 \\ 765 - d_n, & 510 < d_n \leq 765 \\ 0, & 765 < d_n 1529 \end{cases}$$

$$p_b = \begin{cases} d_n, & 0 < d_n \leq 765 \\ d_n - 765, & 765 < d_n \leq 1020 \\ 255, & 510 < 1020 \leq 1275 \\ 1529 - d_n, & 1275 < d_n 1529 \end{cases}$$

In case of inverse colorization, the mapping is done on the disparity value ($disp$) which is the reciprocal of depth.

$$disp = \frac{1}{d}, disp_{min} = \frac{1}{d_{min}}, disp_{max} = \frac{1}{d_{max}}$$

$$d_n = \frac{disp - disp_{min}}{disp_{max} - disp_{min}}$$

It is imperative that a simple and computationally efficient colorization approach be employed especially for real-time classification.

D. CNN Models

Convolutional neural networks (CNNs) is a deep learning network that automatically learns from visual data. In contrast to classical machine learning algorithms, CNN offers end-to-end model development without the need to manually extract features. CNN models learn patterns from input data via the convolution and pooling of multidimensional matrices. Four common models were considered in this study: AlexNet, VGG16, GoogleNet, and ResNet101.

1) AlexNet: AlexNet [29] is a convolutional neural network (CNN) architecture that was introduced in 2012 by Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton. It was designed to compete in the ImageNet Large-Scale Visual Recognition Challenge (ILSVRC) and achieved a significant improvement in accuracy over previous methods. AlexNet consists of five convolutional layers and three fully connected layers, with multiple convolutional kernels extracting features from the images. The architecture also includes max-pooling layers and ReLU activation functions to improve performance. AlexNet's success in ILSVRC helped popularize deep learning and CNNs, leading to many more papers and applications in computer vision.

2) GoogleNet: GoogleNet [30], also known as Inception v1, is a convolutional neural network architecture that was introduced in 2014 by researchers at Google. The architecture was designed to improve the performance of neural networks by making them deeper while avoiding the complications that arise with an increasing number of layers. GoogleNet uses a unique architecture called the inception module that consists of multiple convolutional layers with different filter sizes and pooling operations. The Inception module allows the network to capture features at different scales and resolutions, thereby improving its ability to recognize objects in images. GoogleNet also includes auxiliary classifiers that help combat the vanishing gradient problem and improve training performance.

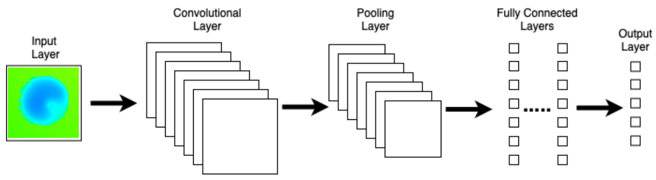

Fig. 2. General CNN architecture with depth image input.

3) ResNet101: ResNet101 [31] is a convolutional neural network architecture that was introduced in 2015 by researchers in Microsoft. The architecture is 101 layers deep and includes a unique feature called the "identity shortcut connection," which allows the model to skip one or more layers. This approach helps to combat the vanishing gradient problem and allows the network to be deeper without sacrificing its performance. ResNet101 was designed to improve the accuracy of image classification tasks and it achieved high performance on the ImageNet dataset at the time of its introduction. The architecture has since been widely used and studied in the field of computer vision.

4) VGG16: VGG16 [32] is a convolutional neural network architecture introduced in 2014 by researchers at the University of Oxford. The architecture is unique in that it has only 16 layers with weights, as opposed to relying on a large number of hyper-parameters. VGG16 was designed to improve the accuracy of image recognition tasks and it achieved high performance on the ImageNet dataset at the time of its introduction. The architecture consists of five blocks of convolutional layers, each followed by a max-pooling layer, and three fully connected layers. The convolutional layers use small 3 × 3 filters, which help to reduce the number of parameters in the model.

E. Transfer Learning

Transfer learning is a machine learning technique that involves reusing a pre-trained model as the starting point for a new model on a different task. The pre-trained model has already been trained on a large dataset and has learned to recognize a wide range of features. By using the pre-trained model as a starting point, we can save time and computational resources that would otherwise be required to train a new model from scratch. Transfer learning is particularly useful when we have limited data for the new task, as the pre-trained model can provide a good starting point for learning the new task. Transfer learning has become a popular technique in deep learning, and it has been used in a wide range of applications, including image classification, object detection, and natural language processing. By leveraging the knowledge learned from pre-trained models, we can improve the performance of our models and reduce the time and resources required for training.

III. Materials and Methods

The general architecture of the CNN model in this study is illustrated in Fig. 2.

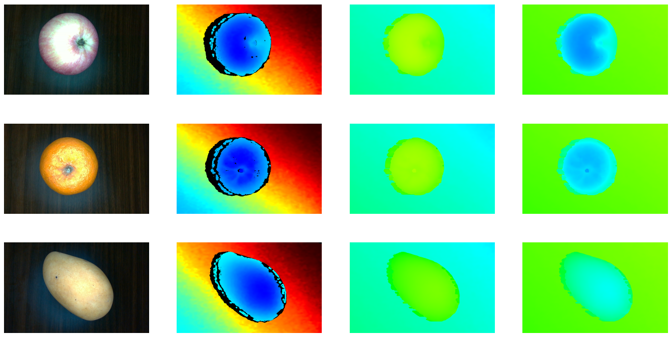

Fig. 3. Dataset samples: (left-right) RGB, jet (D1), uniform (D2), inverse (D3).

A. Dataset

In this study, there was a need to gather depth images of fruits from scratch. A depth camera based on stereo vision matching, Intel RealSense D405, was used to collect depth images in an indoor environment at an 848 × 480 resolution (30 fps). This depth camera is ideal for close-range applications, providing sub-millimeter accuracy to capture small features in an object that is suitable for this study. It was attached to a tripod ∼10 cm from the platform, and the surroundings were artificially lit with a 14 W LED ring light at 1600 lumens. The camera was placed in the middle of the ring light for even lighting. Fig. 3 shows a set of dataset samples with RGB and depth images in three colorization: jet, uniform and inverse.

One important aspect of this study is the choice of colorization for the depth images. Based on [23], there is no significant benefit from using computationally expensive approaches, as there is no evidence that this translates to high performance. Therefore, three simple colorization approaches were used. The researcher refers to these as D1, D2, and D3 for the color-jet, uniform, and inverse colorization, respectively. These datasets have one-on-one correspondence in samples, i.e., each sample in each dataset was taken at the same time, with the same fruit object and resolution, just a different colorization. In this way, we can also make fair comparisons of the performance of the three colorization methods.

The fruits considered in this study were apple, orange, mango, banana, and rambutan (*Nephelium Lappaceum*). These fruits were selected for the following reasons: (a) apples and oranges are different in color and similar in shape; (b) mangoes and bananas are similar in color and different in shape, and apples and rambutans are similar in color and have different shapes and textures.

B. Post-Processing Filters

To improve the depth quality and accuracy, the depth data from the depth camera underwent a series of post-processing filters before colorization, except for jet. Both the uniform and inverse colorized depth images underwent a series of filters, namely, decimation, spatial, temporal, and hole filling. Depth-to-disparity transformation and vice versa are required for certain filters.

Fig. 4 presents a summary of the post-processing filters used for each dataset. A decimation filter was used to minimize depth scene complexity. This was achieved by running an N × N median filter. A spatial filter was used to perform a 1D edge-preserving filter using a high-order domain transform in both the horizontal and vertical directions. A temporal filter was applied to enhance the persistence of depth data by pixel value manipulation over a number of previous frames. This is done by implementing a single pass on the data and updating the depth values while keeping track of the historical values. The hole-filling filter is intended to complete missing depth values and is performed by selecting neighborhood pixels to replace the missing depth.

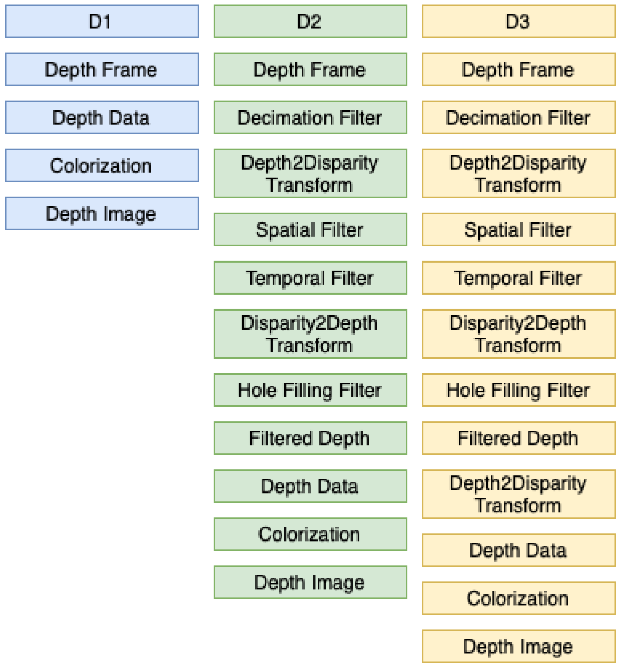

Fig. 4. The post-processing pipeline for each dataset.

TABLE II. FILE SIZE STATISTICS (IN BYTES)

	Min/Max	Mean	Median	Mode
D1	111,255/187,947	153,930	162,560	152,809
D2	76,041/111,565	96,124	99,273	93,907
D3	**68,367/110,659**	**88,971**	**90,198**	**89,373**
RGB	659,071/836,928	759,140	762,023	722,643

After colorization, the depth images were saved using the PNG format, a type of lossless compression. Table II summarizes the statistics of the sizes of the different depth images, including their corresponding RGB color images in bytes. As shown, the inverse colorization requires the least storage among the three datasets used in this study. Generally, depth images require less storage than RGB images do. The RGB color image was almost 10 times the size of the inverse depth image. This is one benefit of using depth images over color images in computer vision tasks.

C. Experimental Flow

The activities of this study include data preprocessing, data augmentation, data splitting, training and validation, and performance evaluation as shown in Fig. 5. Each step is described in the following subsections.

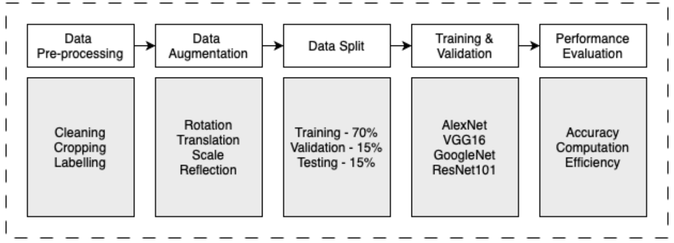
Data Pre-processing
Data Augmentation
Data Split
Training & Validation
Performance Evaluation
Cleaning Cropping Labelling
Rotation Translation Scale Reflection
Training - 70% Validation - 15% Testing - 15%
AlexNet VGG16 GoogleNet ResNet101
Accuracy Computation Efficiency

Fig. 5. Flowchart of data processing and analysis.

TABLE III. Dataset Distribution per Class in each Dataset Type

	Raw	Cleaned	Augmented	Cleaned + Augmented
apple	1,068	456	1,368	1,824
banana	1,057	534	1,068	1,602
mango	1,075	861	861	1,722
orange	1,077	301	1,204	1,505
rambutan	1,066	882	882	1,764
	5,343	3,034	5,383	8,417

1) Data Pre-processing: A total of 5,343 depth images were collected for each dataset type. However, all these cannot be used because some are extremely noisy such as images with large regions with missing depths. To clean the dataset, a MATLAB program was developed to display and manually check each image in color and depth formats. As a result of the cleaning process, there are only 3,034 valid samples in the dataset, as shown in Table III. Orange and mango produced the most and least number of invalid samples, respectively.

After the dataset was cleaned, the images were cropped ∼80 pixels from the left border because of invalid depth [33]. Consequently, the dimensions of the image were reduced. This is a known issue for stereo vision algorithms that utilize the left imager as a reference because of the non-overlapping region in the camera's field of view.

2) Data Augmentation: Because the resulting dataset was reduced after cleaning, there was a need to perform data augmentation to increase the sample size. Transformations used in data augmentation include rotation, translation, scaling, and reflection. It is important to note that exactly the same augmentation was performed on the same depth image of each type to ensure uniformity across the different dataset. For instance, the same transformation is applied to sample image X across D1, D2, and D3. This ensures uniform transformation across depth images and provides fairer performance comparisons later. The dataset (both the cleaned and augmented) now totals 8,417 per type, i.e., 25,251 depth images across all three datasets.

3) Data Split: To avoid possible overfitting in model training, the dataset was augmented and divided into 70% training, 15% validation, and 15% testing. To provide better and fairer comparisons across types, the split was performed uniformly across types and evenly between the cleaned and augmented samples, i.e., each split contained a proportional distribution of cleaned and augmented samples. To do this, each sample in the cleaned and augmented dataset was numbered sequentially, and a MATLAB program was developed to uniformly and evenly divide the dataset.

4) Training and Validation: Model training was performed after the dataset was processed and split. The CNN models used in the training were AlexNet, VGG16, GoogleNet, and ResNet101, with a batch size of 32 and an epoch of 20. The Adam optimizer was used, and the loss type was the categorical cross-entropy. The training was run using the TensorFlow framework on Macbook Pro M2 with 16GB memory, 8 CPU, and 10 GPU cores. Prior to training, the images were rescaled between 0 and 1 and resized to 224 × 224 for AlexNet and 227 × 227 for the rest. In addition, a random seed was set, and TensorFlow op was enabled for deterministic output. A total of 12 training sessions were performed, i.e., four models were trained for each dataset. Retraining of the best-performing dataset per model was performed to determine the effect of transfer learning.

5) Performance Metrics and Evaluation: To evaluate the performance of the trained model, we ran the trained models on the test dataset and utilized standard performance measures. The metrics used in this study were the average per-class accuracy (A), precision (P), recall (R), macro-average F1-score ($F1_M$), weighted-average F1-score ($F1_\mu$), Kappa score (k), training time, and inference time. A confusion matrix was derived from each model after testing to compute the metrics.

$$A = \frac{\sum_{a=1}^{L} \frac{tp_a + tn_a}{tp_a + tn_a + fp_a + fn_a}}{L}$$

$$P = \frac{\sum_{a=1}^{L} \frac{tp_a}{tp_a + fp_a}}{L}$$

$$R = \frac{\sum_{a=1}^{L} \frac{tp_a}{tp_a + fn_a}}{L}$$

where tp_a, tn_a, fp_a, and fn_a represent true positive, true negative, false positive and false negative for class a.

$$F1_M = \frac{1}{|L|} \sum_{a \in L} F1_a$$

$$F1_\mu = \frac{1}{\sum_{a \in L} Supp(a)} \sum F1_a \times Supp(a)$$

where $Supp(a)$ denotes the number of samples in class a, and $F1_a$ is the F1-score of class a

$$F1_a = 2 \times \frac{P_a \times R_a}{P_a + R_a}$$

where P_a and R_a are precision and recall for class a respectively.

$$k = \frac{p_0 - p_e}{1 - p_e}$$

where p_0 is the observed agreement ratio and p_e is the hypothetical probability of change agreement. k is a statistical measure of inter-rater agreement for categorical data.

Training and inference times are the times required for the model to train and test, respectively. These are important metrics for verifying the calculation efficiency of models trained on depth images.

IV. Results and Discussion

This section presents and discusses the findings of this study. Here, the researcher aims to demonstrate the performance of the models trained using different datasets. First, the performance by model with the use of the three datasets during training and validation is discussed, followed by the performance of each model by dataset. Subsequently, the performance of each trained model on the test dataset is presented. The calculation efficiency of each model is also discussed. Finally, the effect of transfer learning on model performance is presented.

A. Comparison of Performance during Training and Validation

In this section, the performances of the model in terms of training and validation accuracy and loss are compared. A desirable CNN model should rapidly improve accuracy and maintain stability as the number of epochs increases. Fig. 6 shows the training and validation performance of each model on the three datasets. With AlexNet, the training performance for all datasets showed a rapid and stable trend. This was also evident from the training loss trend of the model. Its validation accuracy was slightly lower than that of D2, showing an early increase as opposed to D1 and D3. Both D1 and D3 tended to oscillate in their validation performances during the early epochs. GoogleNet had a slower increase in training accuracy compared to AlexNet in all three datasets, although it stabilized well in later epochs. Its validation performance was more stable than that of AlexNet, even at earlier epochs. Compared with GoogleNet, ResNet101 showed a more rapid increase in training accuracy, but was still slower than AlexNet. It also exhibited a stable trend as the number of epochs increased. However, it showed very unstable validation performance across the three datasets. VGG16 has a better training performance than GoogleNet, particularly for D1. Its validation performance is comparable to that of GoogleNet, which registered high validation at earlier epochs and stabilized onwards. It showed the most stable validation performance for all datasets among all models.

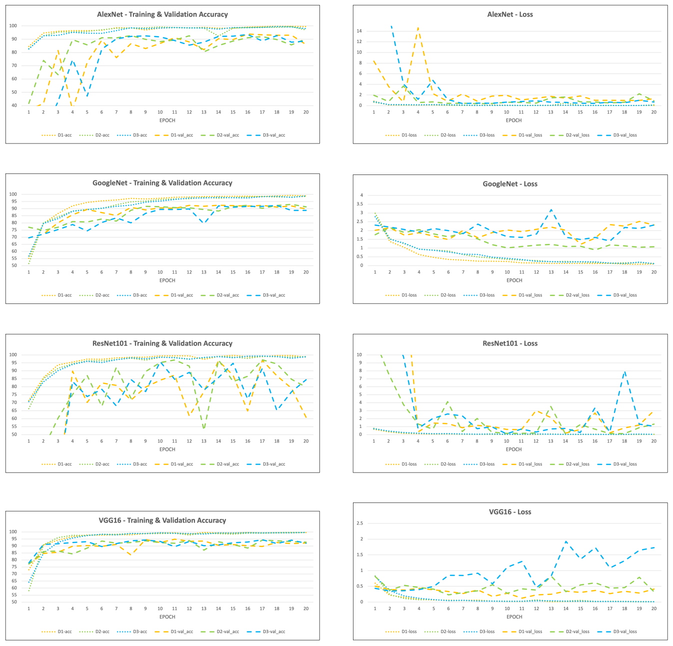

Fig. 6. Training & validation accuracy and loss by model.

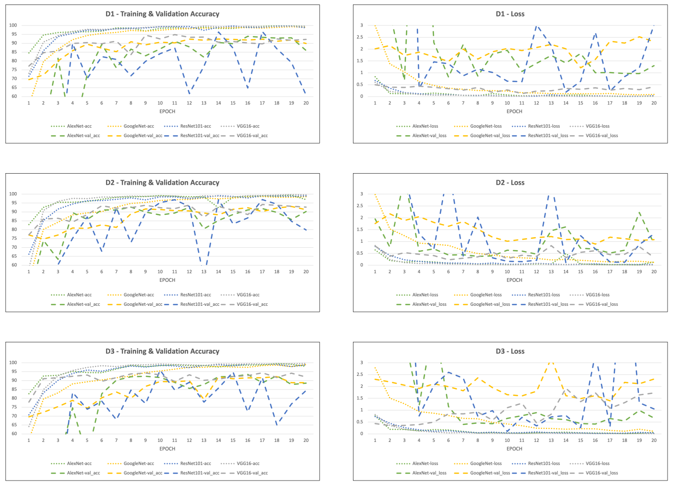

Fig. 7. Training & validation accuracy and loss by dataset.

Next, we look at the performance of the models by dataset as shown in Fig. 7. On D1, we can see that the quickest to rapidly increase in training accuracy is AlexNet followed by VGG16 and ResNet101. GoogleNet is the slowest. Both GoogleNet and VGG16 performs well in the validation set with more stability compared with the other two. ResNet101 is the worst to perform in the validation set with very unstable trend. Only VGG16 has stable trend in the validation loss compared to the rest with ResNet101 being the worst. In terms of D2, AlexNet still leads in terms of rapid increase in training accuracy with GoogleNet still trailing behind the rest. ResNet101 still has the worst validation performance. The validation loss of GoogleNet tend to be more stable in D2. In terms of D3, AlexNet still is the quickest to rapidly increase in training accuracy still with GoogleNet the worst. The trend of validation accuracy of ResNet101 still oscillate. We can say that all models perform well in the training set across the three datasets but the validation performance of ResNet101 which is quite unstable.

B. Comparison of Performance on the Test Dataset

In this section, the performances of the models on the test datasets are presented. Table IV summarizes the performance measures for each model using different datasets. It can be observed that ResNet101 has the best performance across the three datasets and in all performance metrics. It had the highest accuracy, precision, recall, kappa-score, $F1_M$ and $F1_\mu$ on D3, followed by D2 and D1. On D1, it was followed by VGG16 and GoogleNet with AlexNet, with the poorest performance in all metrics. It has the lowest performance, with an average per-class accuracy of 0.76 and k of 0.7. Note that D1 did not pass through the post-processing pipeline, which may have contributed to this result. In terms of D2, ResNet101 was followed by VGG16 and AlexNet, with GoogleNet performing the worst. It should be noted that AlexNet and GoogleNet had the same accuracy, precision, and recall values. They only differ in terms of k, $F1_M$ and $F1_\mu$ with AlexNet higher by 0.01 only. In terms of D3, next to ResNet101 are VGG16 and AlexNet, with GoogleNet trailing behind. It can be seen that VGG16 performs the second best overall with ResNet101. Overall, all models performed well in the three test datasets, except for AlexNet on D1, which registered $< 80\%$ across all metrics. The top dataset is D3, which is based on inverse colorization and registered 96% accuracy using ResNet101.

C. Training and Inference Duration

In this section, the computational efficiency of these models is discussed. The number of CNN parameters and the computational complexity are vital for the development of deep-learning applications. These variables contribute to the training and inference durations of the CNN models. An ideal CNN model is one that is less complex, yet produces good results at low training and inference times.

The number of layers and parameters in each CNN model, including the FLOPs, is listed in Table V. These variables define a complex CNN structure. The deeper the layers in a network, the more complex image processing properties that it can perform. Consequently, the hardware requirements for processing are greater. In essence, computational efficiency is determined by the amount of layers in the network and the training time, whereas computational difficulty is evaluated by the number of network parameters and FLOPs. The training and inference times of the models were also presented.

The AlexNet model has only 11 layers, which is the smallest among all the models considered in this study. It also had the lowest number of FLOPs. However, it did not have the lowest training time. It was only next to GoogleNet across all three datasets. This is due to the number of parameters, which is 60M compared with 6.8M of GoogleNet. Notably, VGG16 had the longest training time compared to ResNet101. Both also had approximately the same inference time. This can be attributed to the number of parameters VGG16 has including its massive amount of FLOPs. Among all three datasets, D3 requires the least amount of computation using GoogleNet in both training and inference with 41.07 and 0.53 minutes for training and inference, respectively. Note that D3 required the least storage which may have contributed to this outcome.

D. Transfer Learning

The researcher compared the accuracy of the four models with and without transfer learning. Only the dataset with the best accuracy during training from scratch for each model was chosen for the transfer learning experiment. D2 was used for AlexNet, GoogleNet, and VGG16, whereas D3 was used for ResNet101. The weights of the four models trained on ImageNet dataset were used.

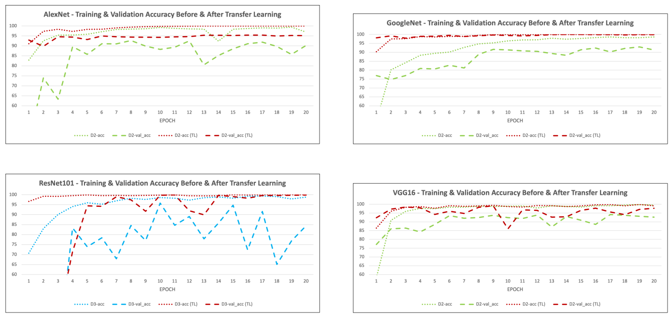

Fig. 8. Training & validation accuracy before and after transfer learning.

Fig. 8 shows the accuracy trend of each of the four models with and without transfer learning. The red lines indicate the accuracy of the model with transfer learning, and the other color indicates the accuracy without transfer learning. It is evident that the training accuracy of each model rapidly increased at earlier epochs with transfer learning compared to the accuracy without transfer learning. It also has a more stable rate than without transfer learning. It can be observed that the training accuracy is considerably higher with transfer learning, particularly for AlexNet, GoogleNet, and ResNet101. The training accuracy for VGG16 remained unchanged in later epochs.

In terms of validation accuracy, Fig. 8 shows a better overall performance in all models. Of note is ResNet101, which registered a more stable validation performance with fewer oscillations compared with no transfer learning. Both AlexNet and GoogleNet had very stable validation performances compared with no transfer learning and even with transfer learning for the ResNet101 and VGG16 models. Remarkably, GoogleNet's validation accuracy with transfer learning surpassed its training accuracy without transfer learning. These results indicate that transfer learning substantially increases both the training and validation accuracy of the CNN models. The extent of improvement varied from model to model.

To understand the effect of transfer learning on the four CNN models, the researcher gathered statistics on the training, validation, and testing performance of these models. The results are listed in Table VI. Here, the difference in accuracy is between the best accuracy with and without transfer learning. The increment, on the other hand, is the ratio between the accuracy difference and the accuracy of the CNN model without transfer learning. This measures the improvement provided by the use of transfer learning.

As shown in Table VI, GoogleNet exhibited the highest improvement in all aspects, including training, validation, and testing. It had the highest improvement in testing and the lowest improvement in training. This indicates that the novel GoogleNet architecture is suitable for applying transfer learning using depth images. It should be noted that its training accuracy has already reached 100% with the use of transfer learning. VGG16 was next to GoogleNet, with the highest improvement in the test dataset. This is despite having the lowest increase in training accuracy. It should be noted that VGG16 already has a high accuracy rate, even without transfer

TABLE IV. Performance of the Different CNN Models on the Three Datasets

	D1						D2						D3					
	A	P	R	k	$F1_M$	$F1_\mu$	A	P	R	k	$F1_M$	$F1_\mu$	A	P	R	k	$F1_M$	$F1_\mu$
AlexNet	0.76	0.78	0.75	0.70	0.75	0.76	0.89	0.9	0.88	0.87	0.88	0.89	0.87	0.86	0.86	0.83	0.86	0.87
GoogleNet	0.87	0.88	0.86	0.83	0.85	0.86	0.89	0.90	0.88	0.86	0.87	0.88	0.86	0.86	0.85	0.83	0.84	0.85
ResNet101	**0.91**	**0.91**	**0.91**	**0.89**	**0.90**	**0.91**	**0.95**	**0.95**	**0.95**	**0.94**	**0.95**	**0.95**	**0.96**	**0.96**	**0.96**	**0.95**	**0.96**	**0.96**
VGG16	0.90	0.89	0.90	0.87	0.89	0.90	0.91	0.91	0.90	0.88	0.90	0.90	0.88	0.88	0.87	0.85	0.87	0.88

TABLE V. Training and Inference Duration (Minutes)

	Layers	No. of Parameters (M)	FLOPs (M)	D1		D2		D3	
				Training	Inference	Training	Inference	Training	Inference
AlexNet	11	60	727	84.92	0.70	84.37	0.72	85.65	0.73
GoogleNet	87	6.8	2000	42.68	0.57	41.18	0.55	41.07	0.53
ResNet101	101	44	7600	199.58	0.95	199.00	0.95	198.67	0.97
VGG16	16	138	16000	215.90	0.97	211.87	0.97	221.38	0.95

TABLE VI. Effect on Model Accuracy Before and After Transfer Learning

	Dataset	Training Accuracy Difference (%)	Validation Accuracy Difference (%)	Test Accuracy Difference (%)	Training Increment (%)	Validation Increment (%)	Test Increment (%)
AlexNet	D2	0.56	2.70	3.46	0.56	2.91	3.89
GoogleNet	D2	1.37	6.98	10.92	**1.39**	**7.50**	**12.27**
ResNet101	D3	0.66	4.13	4.00	0.66	4.31	4.17
VGG16	D2	0.03	5.00	8.13	0.03	5.31	8.93

learning. AlexNet registered the least improvement in both validation and test datasets. This is an indication that the traditional CNN structure has no significant effect in testing accuracy. In addition, its architecture was less affected by transfer learning. AlexNet, GoogleNet, and ResNet101 achieved 100% training accuracy with transfer learning. These findings are consistent with the belief that the two nonlinear structures of GoogleNet and ResNet are more suitable for certain classes of inputs, i.e., the accuracy increases as the image classes change. On the other hand, the single-channel classic CNN architectures of AlexNet and VGG16 are thought to be costlier for varying inputs.

The different CNN models respond differently to transfer learning due to their diverse structures. In addition, there is a large difference in terms of the dataset used to train the parameters of these models for transfer learning, i.e., ImageNet, and the actual dataset used in the classification problem, i.e. colorized depth images. Overall, it was shown that transfer learning can significantly affect the classification accuracy of colorized depth images of fruits. This is despite the fact that the pre-trained models were not previously trained on depth images. The extent of improvement depended on the model structure.

V. Limitations and Future Work

This study has shown that it is possible to use colorized depth images in fruit classification with a high rate of accuracy, aided by CNN and transfer learning. However, this study is limited in multiple aspects, such as the dataset used and CNN models considered. The dataset used in this study was limited, with only five classes. In future work, this can be increased to include other types of fruits, including those that are very similar in form, shape, color, and texture. Only four CNN models are used in this study. Future work could include other models, such as MobileNetV2, as well as other colorization approaches. It is also useful to explore the fusion of RGB and colorized depth images in fruit classification problems.

VI. Conclusion

In this study, the researcher investigated the use of colorized depth images in CNN-based fruit classification using AlexNet, GoogleNet, ResNet101 and VGG16 and examined their performance and the impact of transfer learning application. The primary findings are as follows: (1) All four models performed well during training and validation with both GoogleNet and VGG16 having desirable trends in all of the three datasets. ResNet101 is the least ideal. (2) ResNet101 exhibited the best test accuracy with 96% rate on D3, 95% on D2 and 91% on D1. AlexNet performed the least on D1 at 76%. (3) The post-processing filters applied to D2 and D3 contributed to the performance of the models. (4) Transfer learning considerably improved the performance of the models with GoogleNet registering the largest increase on the test set at 12.27%. (5) Transfer learning could provide better validation performance in ResNet101 whose validation performance was very unstable without transfer learning.

Acknowledgment

The researcher would like to thank the University of the Philippines Cebu Central Visayas Studies Center for the 2022 Research Grant and for Kaye and Gabby for the inspiration. To God be the glory!

References

[1] X. Ma, K. Zhu, H. Guan, J. Feng, S. Yu, and G. Liu, *"High-Throughput Phenotyping Analysis of Potted Soybean Plants Using Colorized Depth Images Based on A Proximal Platform,"* Remote Sensing, vol. 11, no. 9, Art. no. 9, Jan. 2019, doi: 10.3390/rs11091085.

[2] Y. Htet, T. T. Zin, H. Tamura, K. Kondo and E. Chosa, *"Action Recognition System for Senior Citizens Using Depth Image Colorization,"* 2022 IEEE 4th Global Conference on Life Sciences and Technologies (LifeTech), Osaka, Japan, 2022, pp. 494-495, doi: 10.1109/LifeTech53646.2022.9754900.

[3] A. Abobakr, D. Nahavandi, J. Iskander, M. Hossny, S. Nahavandi and M. Smets, *"RGB-D human posture analysis for ergonomie studies using deep convolutional neural network,"* 2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC), Banff, AB, Canada, 2017, pp. 2885-2890, doi: 10.1109/SMC.2017.8123065.

[4] T. Amemiya and T. Tasaki, *"Design of Class in Unknown Object Segmentation Focusing on 3D Object Detection in Depth Image,"* 2021 IEEE/SICE International Symposium on System Integration (SII), Iwaki, Fukushima, Japan, 2021, pp. 706-707, doi: 10.1109/IEEECONF49454.2021.9382606.

[5] A. Aakerberg, K. Nasrollahi, C. B. Rasmussen, and T. B. Moeslund, *"Depth Value Pre-Processing for Accurate Transfer Learning Based RGB-D Object Recognition,"* International Joint Conference on Computational Intelligence, pp. 121–128, 2017, doi: 10.5220/0006511501210128.

[6] K. Saleh, M. Hossny, and S. Nahavandi, *"Effective Vehicle-Based Kangaroo Detection for Collision Warning Systems Using Region-Based Convolutional Networks,"* Sensors, vol. 18, no. 6, Art. no. 6, Jun. 2018, doi: 10.3390/s18061913.

[7] I. RealSense, *"What does depth bring to Machine Learning?,"* Intel® RealSense™ Depth and Tracking Cameras, May 02, 2019. https://www.intelrealsense.com/machine-learning-and-depth-cameras/ (accessed Apr. 13, 2023).

[8] L. Jiang, A. Koch, S. A. Scherer, and A. Zell, *"Multi-class fruit classification using RGB-D data for indoor robots,"* in 2013 IEEE International Conference on Robotics and Biomimetics (ROBIO), Shenzhen, China: IEEE, Dec. 2013, pp. 587–592. doi: 10.1109/ROBIO.2013.6739523.

[9] M. E. Rachmawati, M. I. Supriana, and D. M. L. Khodra, *"Toward a New Approach in Fruit Recognition using Hybrid RGBD Features and Fruit Hierarchy Property,"* 2017.

[10] C. C. Ukwuoma, Q. Zhiguang, M. B. Bin Heyat, L. Ali, Z. Almaspoor, and H. N. Monday, *"Recent Advancements in Fruit Detection and Classification Using Deep Learning Techniques,"* Mathematical Problems in Engineering, vol. 2022, p. e9210947, Jan. 2022, doi: 10.1155/2022/9210947.

[11] J. Steinbrener, K. Posch, and R. Leitner, *"Hyperspectral fruit and vegetable classification using convolutional neural networks,"* Computers and Electronics in Agriculture, vol. 162, pp. 364–372, Jul. 2019, doi: 10.1016/j.compag.2019.04.019.

[12] T. Arumuga Maria Devi and P. Darwin, *"Hyper Spectral Fruit Image Classification for Deep Learning Approaches and Neural Network Techniques,"* International Journal of Uncertainty, Fuzziness and Knowldege-Based Systems, vol. 30, no. 3, pp. 357–383, 2022, doi: 10.1142/S0218488522400116.

[13] S. Hou, Y. Feng, and Z. Wang, *"VegFru: A Domain-Specific Dataset for Fine-Grained Visual Categorization,"* in 2017 IEEE International Conference on Computer Vision (ICCV), Venice: IEEE, Oct. 2017, pp. 541–549. doi: 10.1109/ICCV.2017.66.

[14] H. Mureşan and M. Oltean, *"Fruit recognition from images using deep learning,"* Acta Universitatis Sapientiae, Informatica, vol. 10, no. 1, pp. 26–42, Aug. 2018, doi: 10.2478/ausi-2018-0002.

[15] M. S. Hossain, M. Al-Hammadi, and G. Muhammad, *"Automatic Fruit Classification Using Deep Learning for Industrial Applications,"* IEEE Trans. Ind. Inf., vol. 15, no. 2, pp. 1027–1034, Feb. 2019, doi: 10.1109/TII.2018.2875149.

[16] Y. Zhang, S. Wang, G. Ji, and P. Phillips, *"Fruit classification using computer vision and feedforward neural network,"* Journal of Food Engineering, vol. 143, pp. 167–177, Dec. 2014, doi: 10.1016/j.jfoodeng.2014.07.001.

[17] H. M. Zawbaa, M. Hazman, M. Abbass, and A. E. Hassanien, *"Automatic fruit classification using random forest algorithm,"* in 2014 14th International Conference on Hybrid Intelligent Systems, Kuwait, Kuwait: IEEE, Dec. 2014, pp. 164–168. doi: 10.1109/HIS.2014.7086191.

[18] J. L. Rojas-Aranda, J. I. Nunez-Varela, J. C. Cuevas-Tello, and G. Rangel-Ramirez, *"Fruit Classification for Retail Stores Using Deep Learning,"* in Pattern Recognition, K. M. Figueroa Mora, J. Anzurez Marín, J. Cerda, J. A. Carrasco-Ochoa, J. F. Martínez-Trinidad, and J. A. Olvera-López, Eds., in Lecture Notes in Computer Science. Cham: Springer International Publishing, 2020, pp. 3–13. doi: 10.1007/978-3-030-49076-8_1.

[19] T. B. Shahi, C. Sitaula, A. Neupane, and W. Guo, *"Fruit classification using attention-based MobileNetV2 for industrial applications,"* PLoS ONE, vol. 17, no. 2 February, 2022, doi: 10.1371/journal.pone.0264586.

[20] I. RealSense, *"Beginner's guide to depth (Updated),"* Intel® RealSense™ Depth and Tracking Cameras, Jul. 16, 2019. https://www.intelrealsense.com/beginners-guide-to-depth/ (accessed Apr. 13, 2023).

[21] S. Anwar, M. Tahir, C. Li, A. Mian, F. S. Khan, and A. W. Muzaffar, *"Image Colorization: A Survey and Dataset."* arXiv, Jan. 26, 2022. Accessed: Apr. 22, 2023. [Online]. Available: http://arxiv.org/abs/2008.10774

[22] *"Depth image compression by colorization for Intel® RealSense™ Depth Cameras,"* Intel® RealSense™ Developer Documentation. https://dev.intelrealsense.com/docs/depth-image-compression-by-colorization-for-intel-realsense-depth-cameras (accessed Apr. 13, 2023).

[23] F. M. Carlucci, P. Russo and B. Caputo, *"(DE) 2 CO: Deep Depth Colorization,"* in IEEE Robotics and Automation Letters, vol. 3, no. 3, pp. 2386-2393, July 2018, doi: 10.1109/LRA.2018.2812225.

[24] S. Gupta, R. Girshick, P. Arbeláez, and J. Malik, *"Learning Rich Features from RGB-D Images for Object Detection and Segmentation,"* in Computer Vision – ECCV 2014, D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, Eds., in Lecture Notes in Computer Science. Cham: Springer International Publishing, 2014, pp. 345–360. doi: 10.1007/978-3-319-10584-0_23.

[25] M. Schwarz, H. Schulz, and S. Behnke, *"RGB-D object recognition and pose estimation based on pre-trained convolutional neural network features,"* in 2015 IEEE International Conference on Robotics and Automation (ICRA), Seattle, WA, USA: IEEE, May 2015, pp. 1329–1335. doi: 10.1109/ICRA.2015.7139363.

[26] Y. Morvan, D. Farin, and P. H. N. de With, *"Novel coding technique for depth images using quadtree decomposition and plane approximation,"* presented at the Visual Communications and Image Processing 2005, Beijing, China, Beijing, China, Jul. 2005, p. 59603I. doi: 10.1117/12.631647.

[27] A. Eitel, J. T. Springenberg, L. Spinello, M. Riedmiller and W. Burgard, *"Multimodal deep learning for robust RGB-D object recognition,"* 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Hamburg, Germany, 2015, pp. 681-687, doi: 10.1109/IROS.2015.7353446.

[28] G. Ulander Voltaire, *Influence of different colormaps on the perceptual interpretation of numerical values produced by a self-organizing feature map.* 2021. Accessed: Apr. 22, 2023. [Online]. Available: https://urn.kb.se/resolve?urn=urn:nbn:se:kth:diva-296348

[29] A. Krizhevsky, I. Sutskever, and G. E. Hinton, *"ImageNet Classification with Deep Convolutional Neural Networks,"* in Advances in Neural Information Processing Systems, Curran Associates, Inc., 2012. Accessed: Apr. 24, 2023. [Online]. Available: https://papers.nips.cc/paper_files/paper/2012/hash/c399862d3b9d6b76c8436e924a68c45b-Abstract.html

[30] C. Szegedy et al., *"Going deeper with convolutions,"* 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Boston, MA, USA, 2015, pp. 1-9, doi: 10.1109/CVPR.2015.7298594.

[31] K. He, X. Zhang, S. Ren and J. Sun, "Deep Residual Learning for Image Recognition," 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 2016, pp. 770-778, doi: 10.1109/CVPR.2016.90.

[32] K. Simonyan and A. Zisserman, *"Very Deep Convolutional Networks for Large-Scale Image Recognition."* arXiv, Apr. 10, 2015. doi: 10.48550/arXiv.1409.1556.

[33] *"Intel RealSense D400 series product family Datasheet,"* Intel® RealSense™ Developer Documentation. [Online]. Available: https://dev.intelrealsense.com/docs/intel-realsense-d400-series-product-family-datasheet. [Accessed: 14-Apr-2023].