Bachelor of Artificial Intelligence Thesis

Fruit type and weight recognition for payment purposes using computer vision

Proponents:
Nguyễn Ngọc Bách
Lê Minh Chí
Bùi Văn Phúc

Research Supervisor:
Mr. Bùi Văn Hiệu

Artificial Intelligence Department
Hoa Lac campus - FPT University
April 2023

Abstract

Nowadays, retailers and supermarkets are still selling fruit manually, employees must memorize fruit types, scale weight, and then print out a QR (quick response) code containing price and other details. Having to remember every type of fruit will lead to sellers having some mistakes and giving the wrong estimation. In this work, a novel solution is proposed to solve that problem using computer vision together with the connectivity of digital scales and a webcam. The objective is to make the process of giving prices be done with minimum human effort. To achieve that, Computer vision is used in checkout systems for scanning fruit. The idea of this work is to calculate the price of fruit by recognizing the fruit categories by scanning one or multiple objects of the same type in a specific condition and together with weight value from the scale. Fruit recognition in retail is challenging work due to features such as intensity, color, shape, and texture. To predict fruit, a CNN network is used with DenseNet201, the purpose is to make the application process faster. To process the weight directly, an IoT weight sensor, which can connect and send values to other devices namely computers, is needed in combination with a camera for identifying purposes. Unfortunately, such a type of scale is not available in the common marketplace and we have to build our own using a load cell, an HX711 module, and an Arduino Uno. The model reached the best accuracy score of 99.62% and the application takes around 300ms for the result.

Keywords— Fruit recognition, Computer vision, Weight processing, Image classification, Convolution Neural Network, IoT

1 Introduction

Fruit is nutritious. Fruits are an excellent source of essential vitamins and minerals, and they are high in fiber. Fruits also provide a wide range of health-boosting antioxidants, including flavonoid compounds. Eating a diet high in fruits and vegetables can reduce a person's risk of developing heart disease, cancer, inflammation, and diabetes. In the modern world, fruit consumption has been highly increasing due to the need for a good diet and a healthy body, fruit is an important commodity that can be traded everywhere and greatly impacts the economy all over the world, especially in agricultural countries like Viet Nam. Viet Nam earned 3.26 billion USD from exporting fruits and vegetables in 2020 Viet Nam also spent 1.29 billion USD on importing fruits and vegetables [1]. Fruit consumption all over the world reached 206,064 kilotonnes in 2020, and Viet Nam is one of the most fruit consumption with about 3,000 kilotonnes [2]. Coming with that is hundreds of thousands of tons of fruit that need to be categorized and hundreds of thousands of tons that need to be recognized and sold to consumers. The Weights and Measures program [3] is essential to ensure confidence and fairness in the marketplace. The buyer is assured that they get what they pay for and the business owner is assured of a marketplace based on fair competition. Be that important but getting weight, recognizing, and calculating price are all done separately and required much human labor.

Recently, in the process of selling fruit in the supermarket, it still requires staff to weigh them separately before taking them to the checkout. Recognizing different kinds of vegetables and fruits is a difficult task in supermarkets since the cashier must point out the categories of a particular fruit to determine its price. A solution is issuing codes for every fruit. The use of barcodes has mostly ended this problem for packaged products given that most customers want to pick the products by themselves and skip other steps, they'll need to buy the exact amount/number of the prepackaged products. A solution is issuing codes for every single fruit [27], but the work demands large human effort if done manually or costs money to buy an automaton. Another solution is to issue the cashier or customers to select from a lookup table (LUT), however, a selection from a LUT involves significant human factors, and flipping over the booklet not only costs labor but also efficiency is considerably low.

To overcome these problems, many supermarkets have started using fruit recognition systems, which use computer vision technology to automate the process of weighing and pricing fruits[20][27]. These systems use a combination of cameras, sensors, and machine learning algorithms to identify different types of fruits, determine their weight, and calculate their price based on predefined pricing rules. One of the main advantages of using fruit recognition systems in supermarkets is their speed and accuracy. They can process fruits quickly and accurately, reducing the time it takes to weigh and price each item, and reducing the likelihood of errors. This allows supermarket staff to focus on other tasks, such as serving customers and managing inventory, instead of spending time manually weighing and labeling fruits. Another advantage of fruit recognition systems is their ability to reduce waste and improve inventory management. By accurately weighing and pricing fruits, these systems can help supermarkets optimize their inventory levels, ensuring that they have the right amount of stock on hand to meet customer demand. This can help reduce waste by minimizing the amount of unsold or spoiled fruits, and can also help supermarkets save money by reducing their inventory holding costs. Fruit recognition systems also offer a range of benefits to customers. By accurately weighing and pricing fruits, these systems ensure that customers are charged the correct price for each item, reducing the likelihood of overcharging or undercharging. This can help improve customer satisfaction and loyalty, as customers are more likely to return to a supermarket that offers accurate and fair pricing.

The proposed technique also has significant environmental benefits by reducing the use of lightweight plastic packaging and shrink warps, which are currently used to locate barcodes. This plastic waste is an exponentially growing problem all over the world. For instance, approximately 3.5 million tonnes of plastic waste is produced in Australia annually and 0.6 million tonnes was produced via packaging in 2016–2017 [17]. Most of this plastic is not recycled but goes into landfills, and a large amount of this waste makes its way to the sea.[4]. The Environmental Protection Authority (EPA) of Australia recently reported that approximately 75% of low-weight plastic is produced by plastic bags and packaging in supermarkets. Overall, fruit recognition systems are a valuable tool for supermarkets looking to improve their weighing and pricing processes, reduce waste, and improve customer satisfaction. As computer vision technology continues to evolve and improve, these systems are likely to become even more advanced and widespread, offering even greater benefits to supermarkets, retailers, and their customers.

With all the advantages it brings, fruit classification is a complex task involving significant challenges. [13][10][6]The challenges for vision-based classification result from the highly varied physical features of fruit i.e., size, shape, and color. There are so many types of fruits and some of them are very similar which traditional image processing can not overcome. Although many fruits are of distinguishing features, it is a very tough job to recognize them individually with a machine. At a higher level of abstraction, these challenges can be categorized as the Classification of different fruits and the classification of different varieties of fruits or vegetables. However, the classification of fruit at supermarket checkouts presents additional challenges such as variable environmental lighting conditions, and human elements in the scanning process. The changes in the position on the scale and the number of fruits the recognition is also a challenging issue.

To solve all those problems, we propose a novel solution to calculate the price of fruits by capturing the type of fruit from a camera using convolutional neural networking(CNN) and weight the value from the scale at the same time. A convolutional Neural Network is a Deep Learning algorithm mainly used for applications in image recognition. Its convolution layers network reduces the dimension of the image to prevent information loss, thus, it solves the problem of multiple features traditional image processing meet. And, by capturing weight and getting type at the same time, the misinformation will be reduced. The work consists of 3 parts: training the CNN model, building a machine that contains a webcam and scale, and user interface for interaction. First, the value of the frame and weight would be received from the scale and the webcam. The CNN model will recognize the fruit type search for the unit price, the price then be calculated by multiplying it with the captured weight.

2 Literature review

This section will analyze methods to obtain weight value for payment and billing purposes:

The retail industry is comprised of multiple products such as electronic devices, clothing, and fresh produce. While those machines and clothes are already priced, fruit and vegetables are however products namely meats, fruits, and vegetables are required to be weighed and priced before being sold. There is a variety of traditional methods for gaining the weight value of fresh produce as mentioned in the 1. Introduction. However, to eliminate all manual requirements in acquiring weight values, an OCR (Optical Character Recognition) system or an IoT weight sensor is capable of the task. The OCR technology is the electronic conversion of handwritten, digital text images into machine-encoded text. For this particular research, digital numbers representing weight value from a scale is the goal. OCR technology has several outstanding advantages, including increased productivity and efficiency, improved cost-effectiveness, and data accuracy. However, there are also some drawbacks to OCR technology that need consideration, such as limited OCR software, depending greatly on the quality of the input, and especially, high initial costs. The use of OCR technology is widespread in various industries such as the economy, education, retail, and healthcare for its automation. However, for small retail businesses, this method is considered unsuitable for its cost. A smart weighing machine is on the other side cost-effective and easy to obtain its components and equipment.

This section will review previous work related to fruit recognition using computer vision and machine learning:

Datasets play an important role in research as it is expensive and hard to create, companies and other existing researchers probably will not give away what they have invested in. There is still a wide range of datasets available on the Internet, but it is harder to choose which set of data is compatible with the ongoing research than generating a new one. However, [15] has provided a comprehensive review of the targeting problem. The author has presented an overview of publicly available fruit image database and their characteristics, also discusses various CNN architectures used for fruit recognition and segmentation, and highlights their strengths and limitations. Table 1 shows a summary of datasets analyzed by the authors.

Dataset	Data
typeVegFru [25]	RGB Images 256 x 256 x 3
Originality by [22]	Hyperspectral images 56 x 256 x 3
Originality by [11]	RGB Images 150 x 150 x 3
Fruits-360 [19]	RGB Images 100 x 100 x 3
Fruits-360 [14]	RGB Images 100 x 100 x 3
Originality by [9]	RGB Images 150 x 150 x 3
Supermarket Data [18]	RGB Images 48 x 64 x 3

Table 1: Datasets' images size

The study for classifying fruit from [25][22][18] mainly concentrated on identifying the fruit types. However, images contain multiple distinct backgrounds, which required a segmentation process to eliminate

the background and only capture the fruits themselves. For [19][14], these papers use a publicly available database on Kaggle; this dataset's images are already segmented, but its objective for harvesting, resulting in images containing unripe or spoiled fruit. Whereas for [11], Katarzyna and Pawel's goal is to carry out fruit recognition for application in retail, one limitation was that this research only developed on Apple images. Whereas [9], the dataset used for this paper contains a variety of fruits, and the objective of the research was for a commercial source trade system. Therefore, [11][9]'s datasets were only containing fruit images that were ripe and adequate for selling and trading purposes, and the images were taken on an unchanged background. Regardless of the above datasets, the purposes are similar, fruit recognition. Therefore it is crucial to identify and analyze those approaches.

Fruit recognition methodology could be performed in various ways. Zhang et al. [26] performed a study of fruit classification with 3 methods to augment data and max-pooling techniques for accuracy. They have obtained 94.94% accuracy, which compared with other machine learning techniques is much higher, namely PCA + kSVM of 88.29% [27], FSCABC + PCA of 89.11% [26].

Yudong Zhang and Lenan Wu [27] present a computer vision-based approach for fruit classification using a multi-class support vector machine (SVM). The authors use color and texture features extracted from fruit images to train the SVM classifier. They evaluate their approach on a dataset of 60 fruit classes and achieve an overall classification accuracy of 92.67%.

Hameed et al. [6] present a coarse-to-fine classification approach for fruits and vegetable recognition at a supermarket self-checkout. The authors propose an AdaBoost algorithm combined with a CNN to improve the classification performance. They evaluate their approach on a dataset of 16 fruits and vegetables classes and achieve an accuracy of 98.05%.

For Steinbrenner et al. [22], the author uses a modified dataset consisting of 2700 fruits and vegetables images from 13 fruit types captured by a 16-band hyperspectral camera. The images were then reorganized into 3-D matrices with 2-D cuts for each spectral band. The intermediate NN model was adjusted by 3 models of network architecture: Pseudo-RGB, convolutional kernel, and linear combination. The convolutional neural network average accuracy was reported at 88.15%, 92.23%, and 85.93% respectively.

Mure¸san and Oltean [14] present a new database of fruit images named Fruit-360 [16]. Also, they introduce the evaluation result of 3 different approaches, a basic CNN, AlexNet, and GoogLeNet models for recognizing fruits. Finally, the author concluded that basic CNN achieves very high accuracy results and is less time-consuming during processing. Sakib et al [19] were also using Fruit-360 [16] for designing and evaluating a number of CNN architectures for classifying fruit. They used multiple fusions of hidden layers and epochs for distinct cases and compared them. The initial model of basic CNN consisted of two convolutional layers, each came with a pooling layer and 2 FC layers. Surprisingly, Mure¸san and Oltean witnessed a training accuracy of 99.79% and achieved 100% accuracy.

Paper [11][9] proposed a similar architecture on the algorithm based on Convolutional Neural Network (CNN). However, while Katarzyna and Pawel [11] evaluated two 9-layer CNNs, Hussain et al. [9] showed an algorithm based on Deep Convolutional Neural Network (DCNN). The two 9-layer methods have the same design, though different weights, the first classify fruit images with an unchanged background, and the second uses images where the background was eliminated. The account for original images was 6161, which was then segmented by using a recognition algorithm for identifying a single apple in the origin image. Each segmented part was then generated into a new image, thus producing a dataset consisting of 23,662 images from 6 types of the apple family. While the author for the DCNN method used a database with 15 distinct categories comprising 44,406 images. The network contains three convolutional-pooling layers, an FC layer, and a dense output at the end. Both kinds of research resulted in significant outcomes where [11] accounted for 99.78% of the evaluation output, and [9] showed a high accuracy of 99%.

In conclusion, CNN whether from basic to different network layouts resulted in higher accuracy estimation than other methods mentioned above.

3 Dataset preparation

The databases used in this study were originally from a database consisting of 44406 fruit images [5]. During collecting this database, the source collectors created all kinds of challenges, which were expected in a real-time scenario at a supermarket or a food stall such as light, sunshine, shadow, and pose variation making the model robust when coping with illumination variation, the camera capturing objects, specular reflection shading and shadows. We tested our model's robustness in all scenarios and it performed quite well.

The images were gathered at various times of the day and on different days for the same category. These features increase the dataset variability and represent a more realistic scenario. The images showed variations in light settings and quality, and one of those variations in imagery is illumination. Illumination can make two images of the same fruitless similar to two images of different kinds of fruits. The dataset is comprised of fruit images under relatively unconstrained conditions. There are also images with the room light on and room lights off, moving the camera and intelligent weight machine closer to the windows of our lab than open windows, closed windows, open window curtains, and closed curtains. It is necessary to cope with the camera capturing objects, specular reflection shading, and lighting and shadow variation for a real application in a supermarket.

Figure 1 below shows an example of the "Apple" Class. As you can see, all the images have various lighting conditions as well as a number of items being shown in the pictures. Moreover, this class is a special case that has many sub-classes under it. Each subclass shows several stages of an apple with different colors and shapes. All of this is crucial in the training process, in order to provide an accurate prediction.

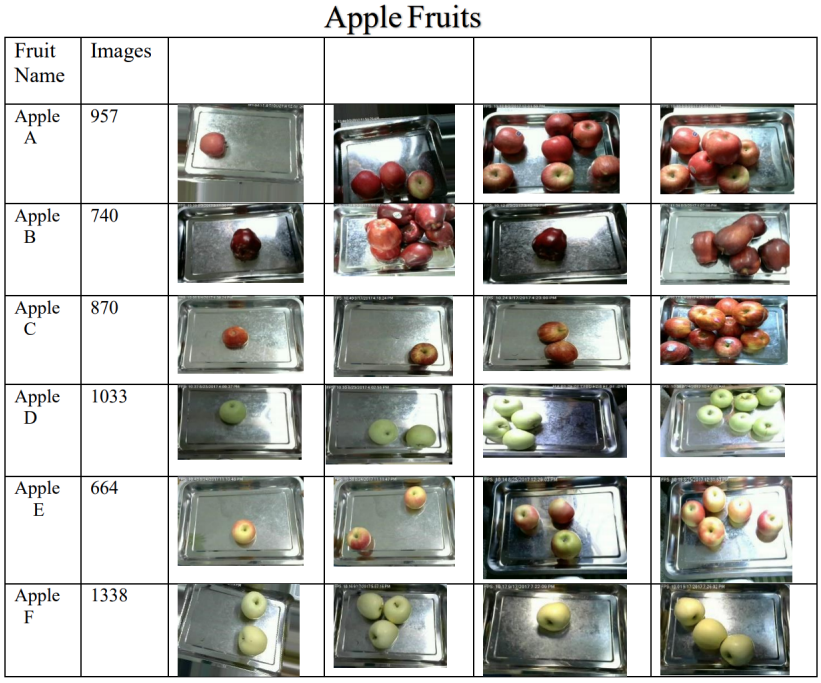


Figure 1: Some Images of Apple Class

While Figure 2 shows the data distribution of the datasets. Even though some classes like guava or apple have a noticeable difference in data size, it is important because they have more various kinds of fruit compared to the others. That's why we still keep them without reducing them. Still, this will cause some problems in bias and overfitting which will be discussed later.

.

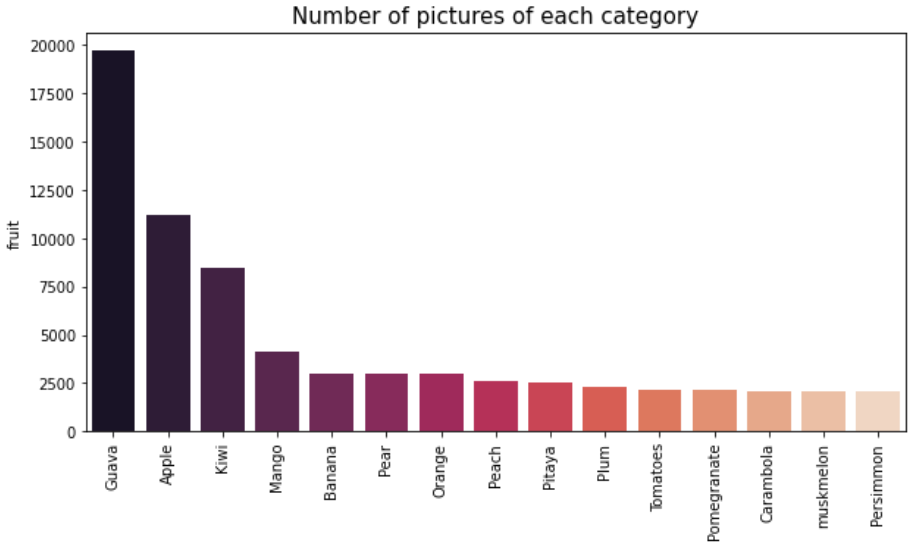


Figure 2: Data distribution

Since the pictures come in various sizes, they need to be normalized and resized. In this study, we downsized all pictures to 150x150 pixels. Moreover, unlike many traditional methods, we keep the RGB color without converting to grayscale as colors are another important factor in recognizing different kinds of fruits in which many have similar shapes where training with greyscale could be more difficult.

4 Methodology

Feature learning is the core of image recognition in general. However, general computer vision methods or even basic neural networks can pick up tons of features from the image, making it hard and taxing for the computer to predict based on those alone. The goal of this section is to choose the most suitable model to find the most representative features, reducing computing time in the process. Thus, out of many considering models, we believe that Convolutional Neural Network (CNN) is the most suitable for the task.

4.1 Basic Convolutional Neural Network (CNN)

4.1.1 Background, objective of using a CNN

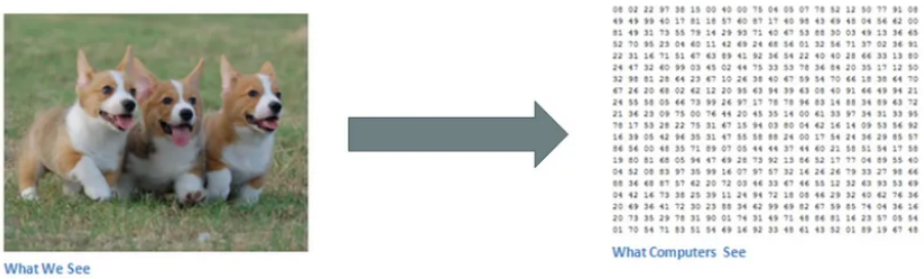


Figure 3: How computer see image

As in Figure 3, let's say we have a color image in. JPG form with a size of 480x480, the computers will interpret it as a matrix of 480x480x3. Each number on the matrix can be a number from 0 to 255 which describes the RGB intensity of a pixel at that point. And the computers will process the image based on these numbers, not how the image looks if seen through human eyes.

So the idea is, just like we identify a dog with features like 4 paws, the shape of the ears, the heads, etc., we will have the computer look at low-level features such as curves or edges and build up to more sophisticated and abstract concepts. These can be obtained by running multiple convolutional layers, and from those initial features that we have extracted to create more high-level detail like paws or heads, it is easier to identify an object.

4.1.2 A Classic CNN

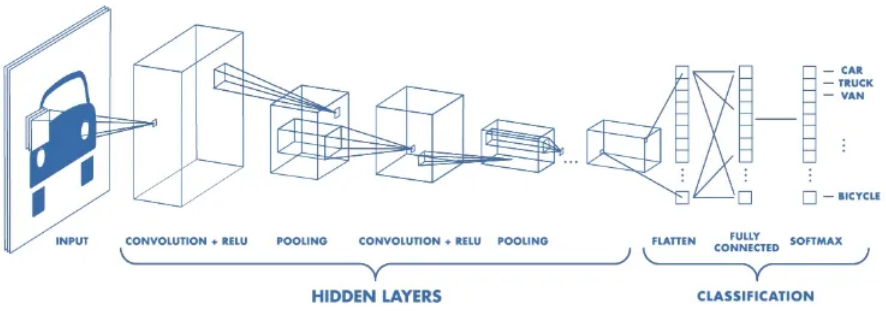


Figure 4: Classic CNN

A classic CNN normally contains 2 blocks: the hidden/convolution block and the classification block.

The convolution block consists of many sequences of convolution and pooling layers. These sequences will have:

*** A convolution layer:**

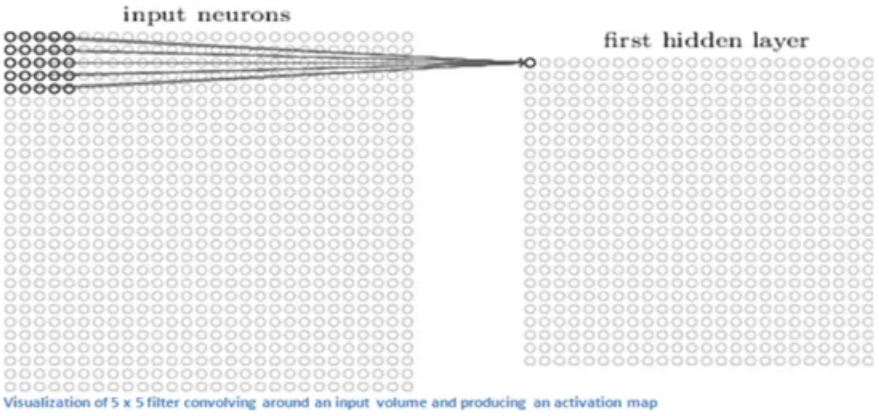


Figure 5: Convolution layer

In the convolution layer, we will have a kernel slide through the entire image, thus multiplying itself with the original pixels of the image into one number. After finishing going through all the pixels, we will have a 2 dimension matrix known as a feature map.

Then the feature map can go through an activation or correction layer like ReLU to correct any negative values to zeros.

*** The pooling layer:**

These layers are often placed within two convolution layers. They process the results of the previous convolution layers into the input for the next ones. Their main function is reducing the size of the feature maps and preserving the most important characteristics. The most popular method of pooling is max pooling, mainly by cutting the maps into 2x2 cells with no overlap or 3x3 cells 2 pixels apart from each other thus 1 row overlapped. Then we get the maximum value of each cell and put it into a new map, resulting in a much smaller map. This is crucial for computing speed and efficiency.

The resulting map may not be as accurate as possible compared to the original one. But it need not be. To identify objects, we do not need to know the exact location of each feature. We only need to know their relative position to other features.

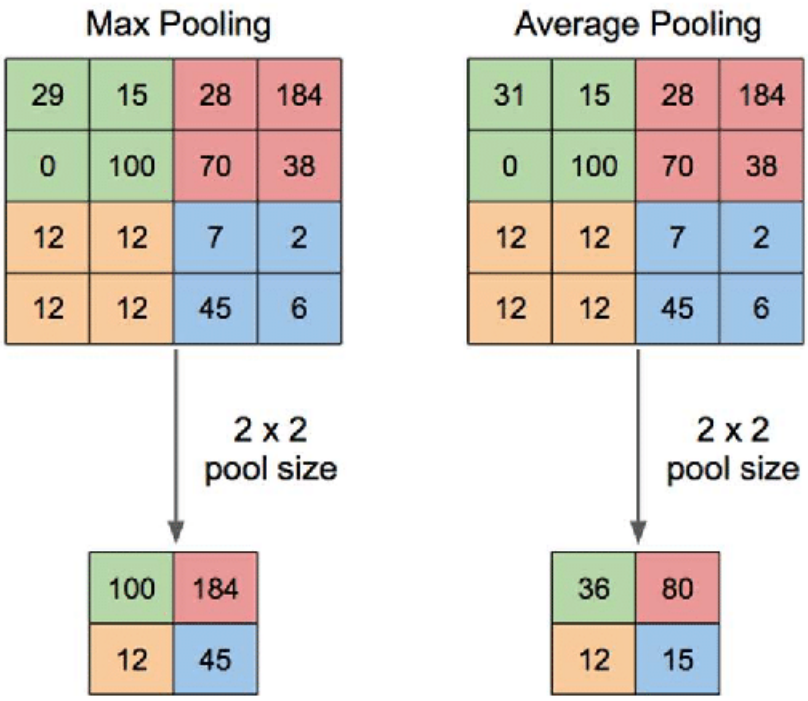


Figure 6: Example of pooling

The convolution block may only have one or many of these consequences. After these sequences, we will have a final feature map that will be used in the classification process.

The classification block is not a feature of the CNN model. It is just a normal classification process that uses the result from the convolution block as an input feature.

In conclusion, unlike the traditional methods that have features predetermined by a particular formalism, CNN models learn and produce the features by training through the model itself. They are initialized and updated through back-propagation with gradient descent.

4.2 Transfer Learning and Pre-trained Models

Transfer learning is a learning method where a pre-trained model is reused to train with a brand new dataset[24]. The basics of how this method works can be seen in the Figure 7 below. The foundation idea here is to use an existing model that has been trained before, normally from a huge amount of data like the ImageNet, and that knowledge that has been learned will be "transferred" to a new model to classify a much smaller task like the one in this study.

Three attributes are important to use the transferred knowledge efficiently: (i) The successfully trained model helps to reduce or get rid of hyper tuning parameters; (ii) The couples of first layers of the pre-trained architecture can be used as feature extractors for low-level ones like blobs, edges, tints, shades or textures; (iii) The only things left to do is re-trained some of the last layers because we believe that the later layers carry out complex identification tasks for the specific classes needed to be classified.

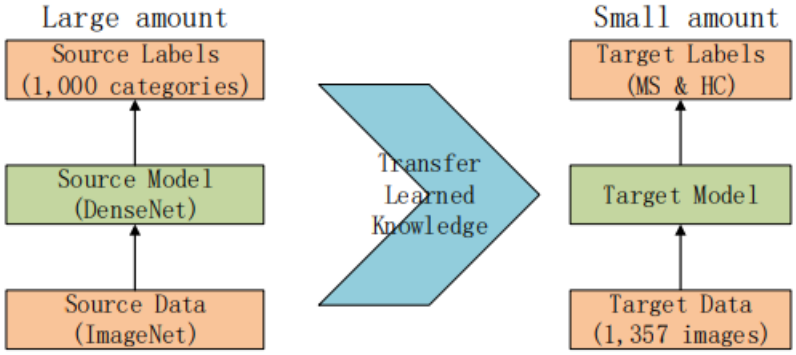


Figure 7: Basic of transfer learning

Pretrained Models are great tools to help create a new deep-learning model based on a pre-established framework. Because of time or hardware limitations, it's unrealistic to build a model from scratch, especially a complex one with a fair amount of data. Nowadays, it's a common sight to use a pre-trained model to build our own model for more specific uses as there are now a few well-developed CNN architectures such as AlexNet, LeNet, VGGNet, and so on. One of those was AlexNet, of which Krizhevsky et al. (2012)[12]brought to the scene in 2012 when it was in a contest with the other CNN models and ended first in the ImageNet Large Scale Visual Recognition (ILSVRC 2012). Even though AlexNet has some similarities to LeNet, it did introduce some new methods like using max pooling and ReLU in a nonlinearity way. Karen Simonyan (2015)[21] offered a new CNN model called VGGNet in 2014, the model was a neural network with 19 layers deep and ended in 2nd place in the ILSVRC2014 competition. Also in the same year, the champion of the contest ILSVRC2014, GoogleNet was introduced by Szegedy et al. (2015) [23] consists of some new unique techniques including using 1x1 filter in convolution instead of traditional 2x3 or 3x3, which greatly reduce the feature map size as well as deepen and widen the network overall. As for the ILSVRC 2015, the winner ResNet [7] provided an even deeper network with an impressive number of 152 layers, which introduced the new method of skipping 2 layers at least or shortcut connection. Then in 2016, Huang et al. (2017)[8] developed a new idea named DenseBlock, which reuses all the feature maps of previous convolutional layers throughout the entire model.

In this paper, we adopted the method of DenseNet for our own prediction tasks as it provided the best performance based on our quick testing using 1/10 of this study's dataset as shown in Figure 8. All the available pre-trained architectures from the kera-tensorflow library were used in this testing round and the result was sorted by accuracy after one epoch. The results show that DenseNet201 surpassed many available applications in the library, with many other DenseNet models also in the top score.

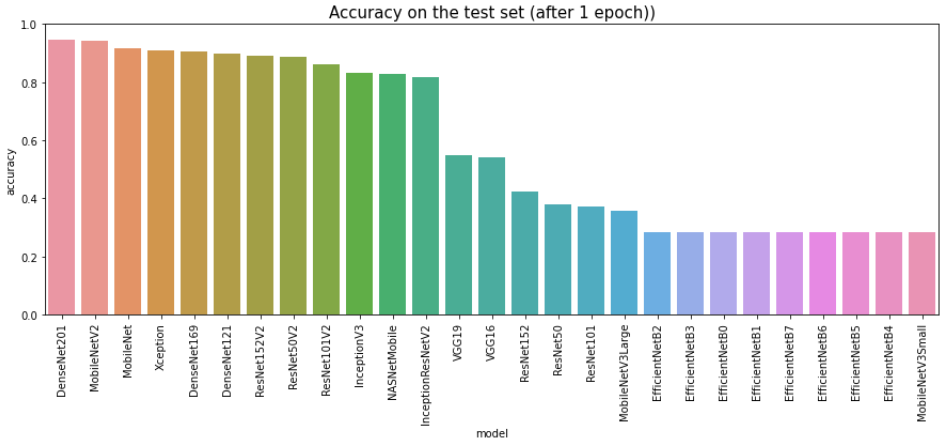


Figure 8: Comparison of pre-trained models

4.3 Densely Connected Convolutional Networks

The basics and classic model of CNN have been discussed above. However, in the traditional CNN, the layers are often connected slowly as in the (1) formulation below. This makes the task of deepening and widening the resulting network very hard, as it may encounter either exploding or gradient vanishing in the process. Later, ResNet introduced the idea of skipping at least 2 layers in order to employ a shortcut connection. The structure was shown below in Figure 9, of which the input is x_{l-1} and the output after two conv layers $H_l(x_{l-1})$ are added together through the shortcut to the input layer x_{l-1}, and thus the $l-th$ layer is the resulting summation, as written in Equation 2.

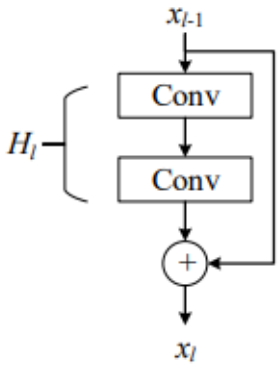


Figure 9: Basic structure of Resnet block

Then from the foundation proposed above, the more advanced DenseNet model was improved further by combining all of the previously existing feature maps one by one, replacing the only final map in the

traditional methods, as shown in Equation 3

$$x_l = H_l(x_{l-1}) \tag{1}$$

$$x_l = H_l(x_{l-1}) + x_{l-1}) \tag{2}$$

$$x_l = H_l([x_0, x_1, x_2, ..., x_l - 1]) \tag{3}$$

Where l s is the index of each layer, and Hstands for the non-linear operation, whilex_l is the feature from the l_{th} layer.

Considering Equation 3, DenseNet provides the combination of all previous layers through concatenating, which means, all the computed feature maps pass on to the later layers and are connected to the currently generated feature map. This newly developed DenseNet model introduces some pros compared to older architectures like reusing features, which reduces the previous concerning problem of either exploding or gradient vanishing. However, to make the structure of DenseNet a viable solution, some changes should be made, like down-sampling the feature maps to make concatenating possible as the size of the feature maps keeps changing and makes the operation impossible to be implemented. And then, The dense blocks structure was introduced to address the down-sampling process. The idea is that between the dense blocks, there will be transition layers, which include these operations: batch normalization, convolution, and pooling operations as shown in Figure 11. On the other hand, Figure 10 illustrates a case of how a Dense Block evolves, in which the layer number is 5 and the growth rate is set as k. As shown below, the blocks increase in size over time and each feature map includes the previous ones.

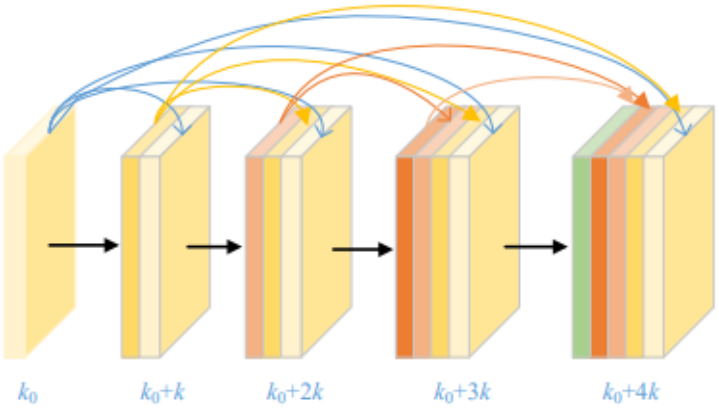


Figure 10: Structure of Dense blocks

From Figure 8, we can find that k feature maps are generated for each operation. Since the example shows that there are 5 layers to be presented, we can get $k_0 + 4k$ feature maps combining with each other after the process is done. k_0 is the number of feature maps in the previous layers. In this paper, we have chosen 32 as the default value of k.

However, due to a huge amount of feature maps as well as data being processed, a final reducing layer must also be added, which is done by implementing a 1x1 convolution kernel before a 3x3 one, which helps size down the feature maps and saves the computation cost. Then, as the model became more and more compacted, a transition layer is required in order to reduce the map in the following ways: we suppose

m feature maps are generated by a DenseBlock and assume the compression factor as $\theta \in (0, 1]$ Then the feature maps will be reduced to $\lfloor \Theta m \rfloor$. If $\theta = 1$, it will not affect the final number of feature maps.

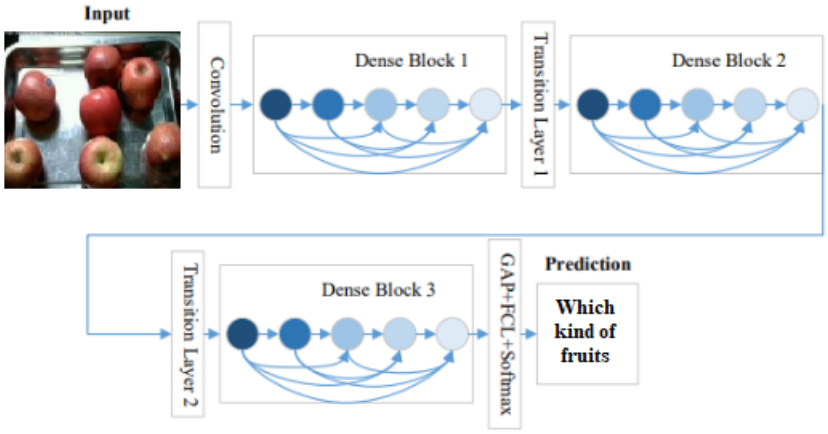


Figure 11: Dense blocks and Transition Layers

Figure 11 illustrates the core structure of a simple DenseNet, including 3 Dense Blocks, an input layer, various transition layers, and global average pooling (GAP) layer at the end. The transition layers consist of a batch normalization layer, which has a 1x1 convolution layer, and a 2×2 average pooling layer with a stride of 2. Particularly, the global average pooling (GAP) is similar to the normally used pooling layers. Still, GAP performs a more drastic feature map reduction, that reduces a feature map of W x W x c down to 1 x 1 x c. It means that the GAP layer will cut down the feature to a single digit.

5 Implementation

5.1 Building model

5.1.1 Building a basic CNN

We are using the keras-tensorflow library for this model. The basic model will consist of these layers below:

Layers	Details
Input	(150,150,3)
Convolution + Relu	number of filers: 32; kernel: 3x3
Max Pooling	2x2
Convolution + Relu	number of filers: 64; kernel: 3x3
Max Pooling	2x2
Convolution + Relu	number of filers: 64; kernel: 3x3
Max Pooling	2x2
Convolution + Relu	number of filers: 64; kernel: 3x3
Max Pooling	2x2
Convolution + Relu	number of filers: 64; kernel: 3x3
Max Pooling	2x2
Convolution + Relu	number of filers: 64; kernel: 3x3
Max Pooling	2x2
Flatten	
Dense layer	size: 256 + reLu
Dense layer	size: 15
Softmax	
Output	

Table 2: layers of basic CNN model

The resulting model is 6 convolution blocks deep. Due to hardware limitations, we will run this model through 100 epochs but with only one-fifth of our dataset.

5.1.2 Implement DenseNet

As we already established, we will use transfer learning following a pre-trained DenseNet model in our final model. After installing the dense block structures, we needed to modify the later layers in order to perform our particular task. The last dense layer, viz., the fully-connected layer was revised, since the original model is trained to predict over 1000 classes. Some of them were related to fruits, however, we need to scale it down to these particular classes to perform more reliably and reduce taxing on the computing process, or in the other word, fine-tuning.

Layer	DenseNet	Replaced by
Third from Last	FCL (1000) with pre-trained weights and biases towards a large scale of classes	DCL (15) with initial value randomly generated
Last	Classification Layer 1,000 classes of many many different kinds of objects	Classification Layer 15 classes of multiple kinds of fruits

Table 3: Revision of last 3 layers

Since the original model output was a very large number of 1000, which was hugely more than our study of 15 classes, we need to change this, and the classification layer to fit our own classification task. The revision is shown in Table 3. In this transfer learning model, the fully connected layer only has 15 neurons, a softmax layer, and a classification layer that only classify 15 classes.

Next, we need to set the training options. First, because this is transfer learning from a pre-trained model, we only used a small number of epochs because most of the model does not need much tweaking. In our experiments, we only use 5 epochs. Second, the learning rate will be a small value of 0.0001 in order to slow down learning as it was learned from a pre-trained one. Figure 12 illustrates how we will approach the transfer process. First couples of dense blocks wear frozen and un-retrainable as this is where the primitive features like curves or edges were extracted. Then we slowly implement the learning operation at the later layers where more specific features learning occurs. After these layers, we obtained the feature maps which are most suitable for our task. Finally, the classification block is redone completely for our task instead of the original 1000 classes of the pre-trained model.

Third, according to Figure 8, we learned that DenseNet-201 was the most effective model for our dataset. The model contains 121 learnable layers, in which $(6 + 12 + 48 + 32) * 2 = 196$ layers were in the Dense blocks, the last 5 layers were from the first convolution blocks, the last FLC, and 3 transition layers. Further details about DenseNet-121 are shown in Table 4, as well as other variations of the DenseNet model.

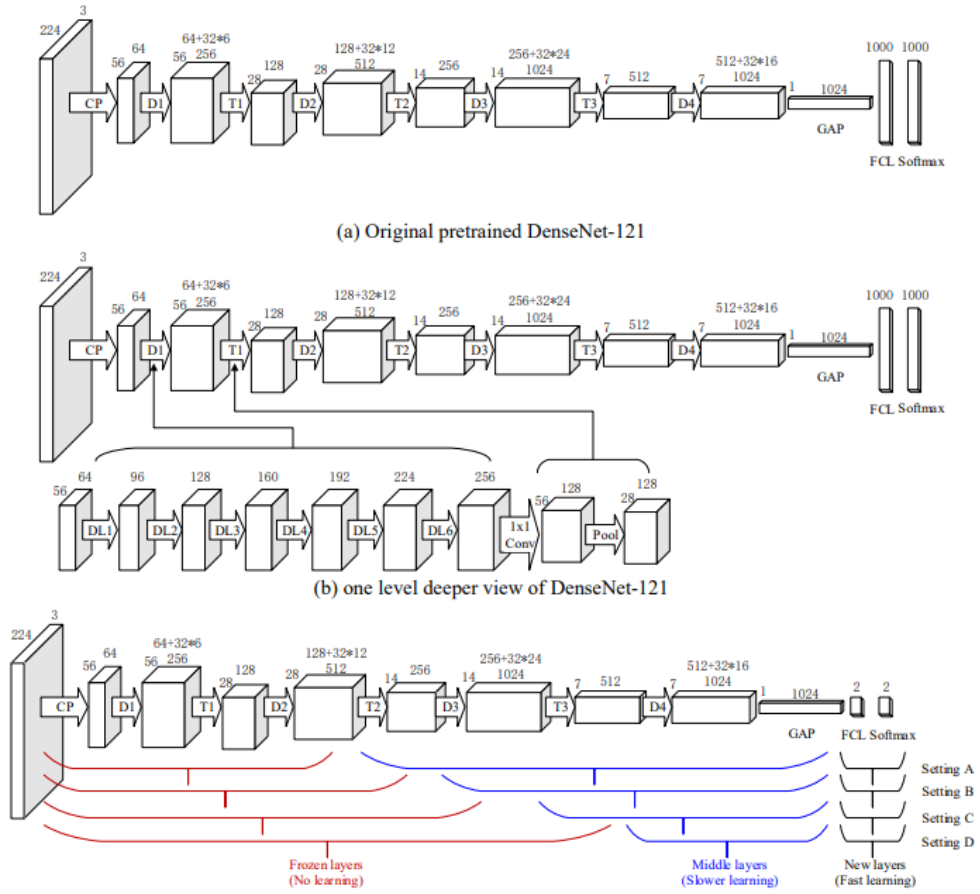


Figure 12: Implement transfer learning

Layers	Output size	DenseNet-121	DenseNet-169	DenseNet-201	DenseNet-264
Convolution	112 x 112	7×7 conv, stride 2			
polling	56 x 56	3×3 max pool, stride 2			
Dense Block (1)	56 x 56	$\begin{bmatrix} 1 \times 1 \ conv \\ 3 \times 3 \ conv \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1 \ conv \\ 3 \times 3 \ conv \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1 \ conv \\ 3 \times 3 \ conv \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1 \ conv \\ 3 \times 3 \ conv \end{bmatrix} \times 6$
Transition Layer (1)	65 x 56	1×1 conv			
	28 x 28	2×2 average pool, stride 2			
Dense Block (2)	28 x 28	$\begin{bmatrix} 1 \times 1 \ conv \\ 3 \times 3 \ conv \end{bmatrix} \times 12$	$\begin{bmatrix} 1 \times 1 \ conv \\ 3 \times 3 \ conv \end{bmatrix} \times 12$	$\begin{bmatrix} 1 \times 1 \ conv \\ 3 \times 3 \ conv \end{bmatrix} \times 12$	$\begin{bmatrix} 1 \times 1 \ conv \\ 3 \times 3 \ conv \end{bmatrix} \times 12$
Transition Layer (2)	14 x 14	1×1 conv			
	14 x 14	2×2 average pool, stride 2			
Dense Block (3)	28 x 28	$\begin{bmatrix} 1 \times 1 \ conv \\ 3 \times 3 \ conv \end{bmatrix} \times 24$	$\begin{bmatrix} 1 \times 1 \ conv \\ 3 \times 3 \ conv \end{bmatrix} \times 32$	$\begin{bmatrix} 1 \times 1 \ conv \\ 3 \times 3 \ conv \end{bmatrix} \times 48$	$\begin{bmatrix} 1 \times 1 \ conv \\ 3 \times 3 \ conv \end{bmatrix} \times 64$
Transition Layer (3)	14 x 14	1×1 conv			
	7 x 7	2×2 average pool, stride 2			
Dense Block (4)	7 x 7	$\begin{bmatrix} 1 \times 1 \ conv \\ 3 \times 3 \ conv \end{bmatrix} \times 16$	$\begin{bmatrix} 1 \times 1 \ conv \\ 3 \times 3 \ conv \end{bmatrix} \times 32$	$\begin{bmatrix} 1 \times 1 \ conv \\ 3 \times 3 \ conv \end{bmatrix} \times 32$	$\begin{bmatrix} 1 \times 1 \ conv \\ 3 \times 3 \ conv \end{bmatrix} \times 48$
Classification Layer	1 x 1	7×7 global average pool			
		1000D fully-connected, softmax			

Table 4: Structure of Various DenseNet models

Lastly, the fine-tuning process is as follows. At the start, we freeze a couple of first layers in the convolution or dense block layers to prevent them from re-learning. The reason is these layers are responsible for the training of low-level details like edges or curves, which is universal in any recognition task. The latter half of the convolution process is where the specific feature started to form. Thus, we slowly re-tuning them to fit out the dataset as well as the study's given problem. Finally, the classification of the FLC layers, which was at first in charge of classifying the 1000 classes of the original ImageNet database, now is completely rewritten to fit our task. The process now consists of 15 neurons matching the 15 kinds of fruit that need to be classified. That's all about the fine-tuning process of our study. The final model will be implemented through the kera-tensorflow library, just like the previous basic CNN model.

5.2 External devices

5.2.1 IoT weight sensor

This paper will use a load cell, an HX711 module, and an Arduino Uno microcontroller for designing a weight scale. The sensor requires the following components. Every component is described below:

S.N	Components	Quantity
1	Arduino Uno Board	1
2	Load Cell 10kg	1
3	HX711 Module	1
4	Scale base	1

- Load cell

  A bar-type load cell is basically a transducer device made up of a load and four strain gauges for measuring strain and then converting forces such as tension, pressure, and compression into electrical energy that could be measured to serve scientists and engineers as measurements.

  The load is a bendable metal bar, the strain gauges part contains 4 small resistors attached to the load that follows the Wheatstone bridge formation. When applied forces on one end of the load, the load cell is bent, and the strain gauges are either compressed or stretched. Consequently, return voltage irregularities use for weight reading.

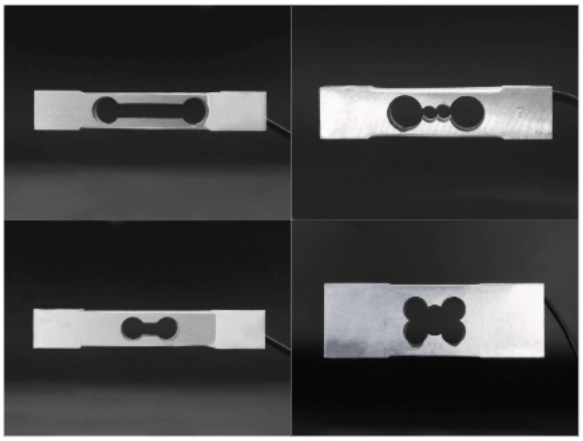


Figure 13: Bar-type load cell

- HX711 Load cell amplifier

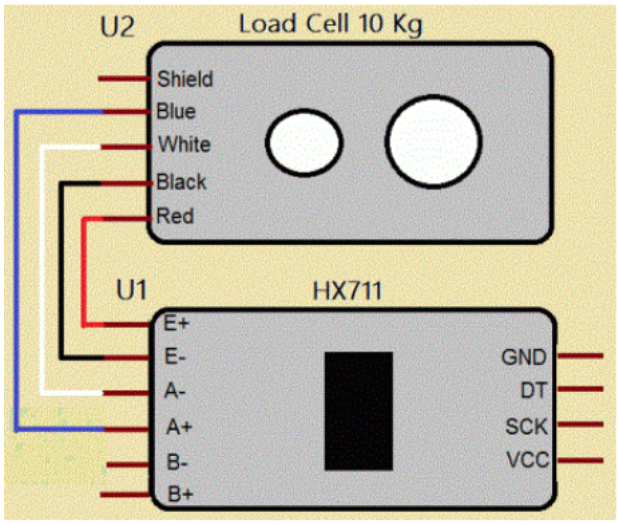
  The HX711 module allows users to easily measure weight by reading load cells. A microcontroller (Arduino Uno) connecting to the amplifier helps read changes in the resistance of the load cell. With some calibration, it is capable of getting accurate weight measurements.

  The HX711 communicates using a two-wire interface, namely Clock and Data. A variety of libraries have been written for reading data from the HX711.

The load cell is figured with four wires colored red, black, white, and green for connecting with the HX711 module. Each color corresponds to the color coding of load cells:

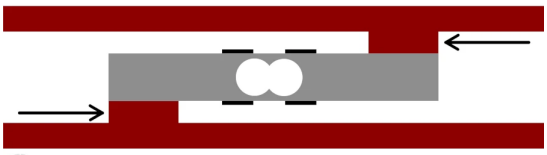
  - Red (E+) or VCC)
  - Black (E-) or GND)
  - White (A+)
  - Green or blue (A-)

- Design and consideration:
  - Load cell and HX711 amplifier module connection:

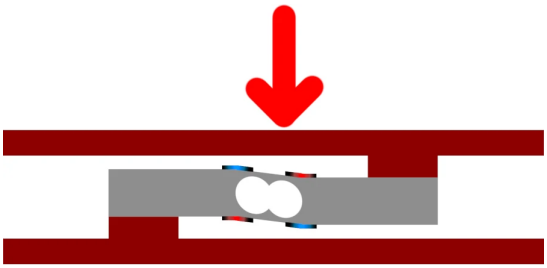


  - Load cell assembly on base:

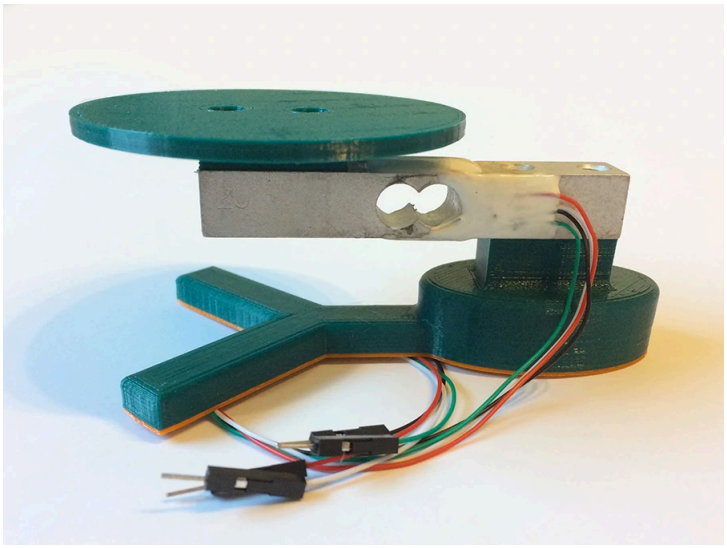
    Attaching the load cell between 2 boards is the most preferred way. The bottom board holds the load cell in place, and the weighing plate is the second board. To deform the load cell, make sure there are some spacers between the board and the load cell.



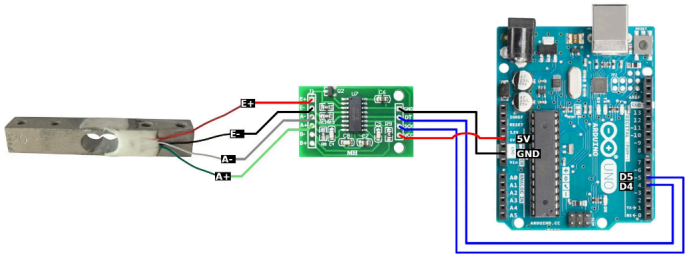
When applying force to the load cell, 2 of the 4 strain gauges will compress (marked red) while the rest will stretch (marked blue).



The weight scale base is 3D-printed, and the following design is capable of measuring objects equal to or below 10kg.



- Circuit diagram



The weight value is generated in a separate setting using the Arduino IDE and a third-party application called CoolTerm. Weight value transmitting from the Arduino Uno and the scale to the serial port captured by CoolTerm. The output is obtained in a .txt file for further processing.

5.2.2 Webcam

For the webcam, we use a HIKVISION webcam with a resolution of 1920 × 1080 and a USB plug-in cable. We use the opencv2 library to capture the frames from the webcam.

5.3 User interface

The UI of the program is done using Python and Tkinter. Tkinter is a Python framework that provides an interface to the Tk toolkit and works as a thin object-oriented layer on top of Tk. The Tk toolkit is a cross-platform collection of 'graphical control elements', for building application interfaces It supports many controls called Widgets such as buttons, labels, scroll bars, radio buttons, and text boxes used in a GUI. The user interface of this application is split into 2 sides: the right side shows the information on the transaction: a list of items that have been scanned with information like type, unit price, weight and price, and total price of items; on the left side show label camera sensor values, and user interaction: a button to capture the current frame and process to item, a button to delete the latest item scanned from the list, a button to process the transaction.

5.4 Putting it all together

Components including a trained model, a webcam, an Iot weight sensor, and a user interface are available. It is time to put all of them together.

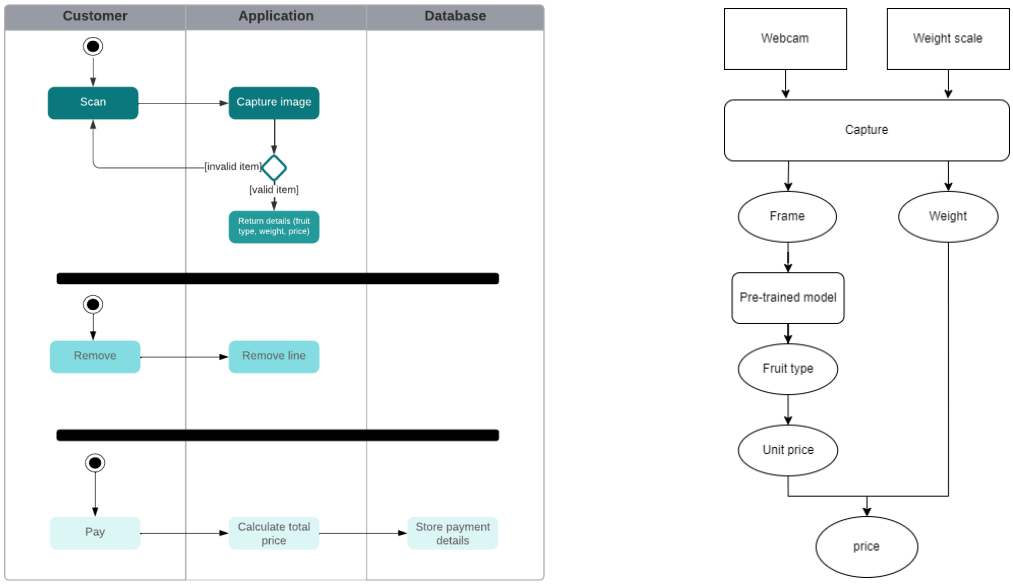


Figure 14: Application workflow and Price calculating progress

Figure 14 shows the method that is developed for recognizing fruits, getting weight, and calculating the price. First, the program will receive frames from the webcam and weight from the weight scale, the frames

are shown on the screen. The user will capture a frame from the webcam and at that time the program also gets the weight value from the scale. The program recognizes the type of fruit from the captured frame by the pre-trained model and uses that type to search for the unit price from the list. Then, the program will calculate the price of the fruit predicted by multiplying the unit price with the weight value got from the saved text file. Users then can decide to continue to add more value to the current bill or to print the bill by pressing the 'pay' button. After printing the bill, the information on the paid items will be stored in an SQLite database for future use and the program continues with the new transaction.

In this work, capturing images to recognize is chosen instead of real-time detection. The first reason is to give users the freedom to choose when to know the value of the fruit. The second reason is to make the program run faster and more consistently. This work gets both inputs from scale and webcam, real-time detection will make the program process more work by continuously processing images and the program will work slower. This will lead to the process of the webcam being unsynchronized with the scale and effect the result to be inaccurate.

6 Result and Discussion

6.1 Using the basic CNN

For the CNN model that we trained from scratch, the results were surprisingly good with a 97% accuracy score even with only 20% of the data. Figure 15 shows the accuracy vs validation score throughout the whole training process. It started at a fairly low, only 0.3 accuracy score but quickly increased within several next epochs. However, it seems that after about 15 epochs, the process slowed down a lot and did not improve much like before from there. And after about 30 epochs, it mostly remained the same. Moreover, there are some dips in performance, noticeably at some epochs after 10, and some at the 40 epochs mark. This is likely due to overfitting and it fixed itself later.

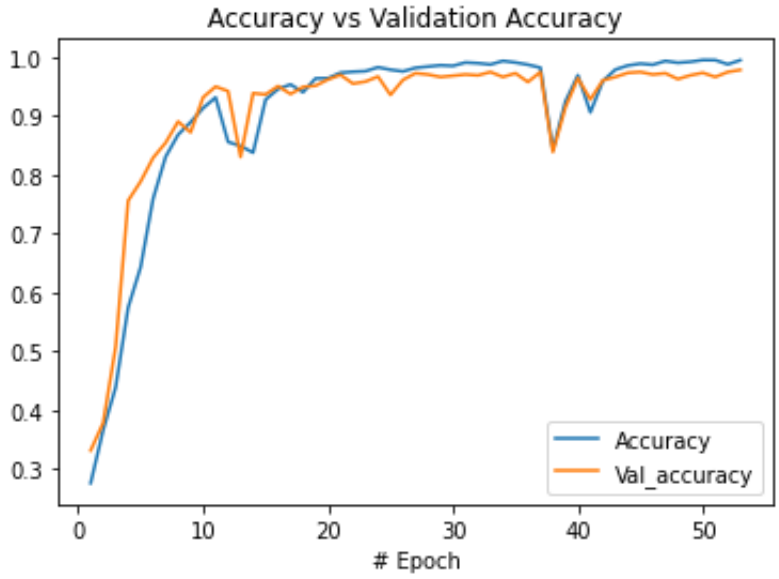


Figure 15: Accuracy vs Validation value chart

Next, let's take a look at some of the predictions and the confusion matrix as shown in Figure 16. The model has no problem when recognizing fruits with very distinct shapes and colors like bananas, mangoes, or pitayas. It has no trouble predicting these types of fruits. But, when it comes to more similar shapes and colors, it does show some struggles. Even though for the most part, it is still able to differentiate them, the false predictions occur noticeably more with apples, oranges, tomatoes, and persimmons. The model sometimes also falsely flags a muskmelon as an apple. This is because they have a similar shape and color to a green apple despite being a much bigger fruit.

This also shows that it is important to train the model with a full RGB image instead of gray-scaled like many other models. This is because there is so much more to fruit than its shape. Colors are such a crucial piece in this kind of task. Another thing to notice is that many false predictions of different kinds of fruit are more likely to be classified as "Apples". This is due to the large number of apple images during the training process, which can influence the model to predict more difficult and confusing images as apples.

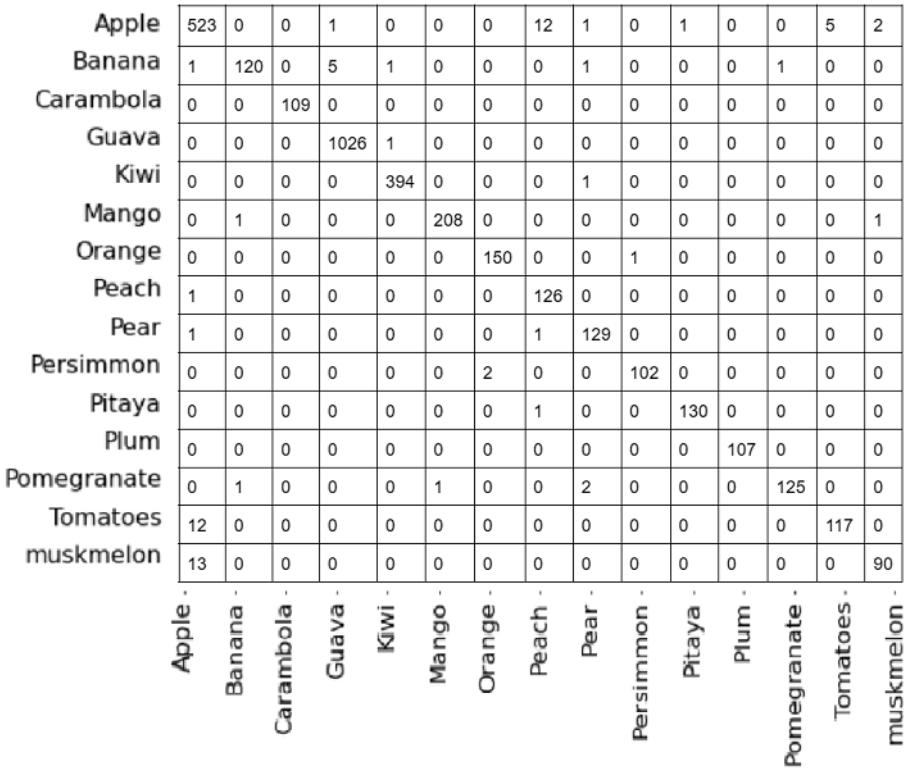


Figure 16: Confusion Matrix

Figure 17 presents some of the predictions using this newly trained model, proving that it can predict with higher accuracy compared to the basic model. It does not encounter many problems, even with highly confusing fruits like apples, peaches, persimmon, tomatoes, or oranges just as we mentioned above.

Figure 17: Some of the predictions using the basic CNN model

Even with a very basic model, CNN still performs relatively well with a respectable accuracy score. Still, there are some problems with overfitting and struggling to distinguish fruits with similar shapes and colors. These will be addressed and improved upon when we switch to a more advanced model below.

6.2 Using Pre-trained DenseNet-121 model

The model that was transferred and learned from DenseNet-121 has an impressive result of over 99% accuracy score. Take a look at Figure 18, even at the first epochs, the model already achieved a very high score of 0.98 accuracy score. After that, it reached the final score of over 0.99 only at epoch 2 with minimal loss. However, the validation accuracy does decrease after the training process. It's most likely due to a fitting operation but the loss increase is very little to worry about. Figure 19 presents some of the predictions using this newly trained model, proving that it can predict with higher accuracy compared to the basic model.

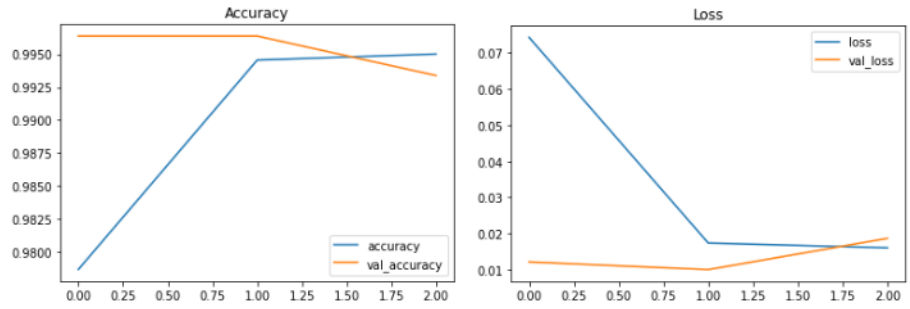


Figure 18: Accuracy and loss of the transfer learning model

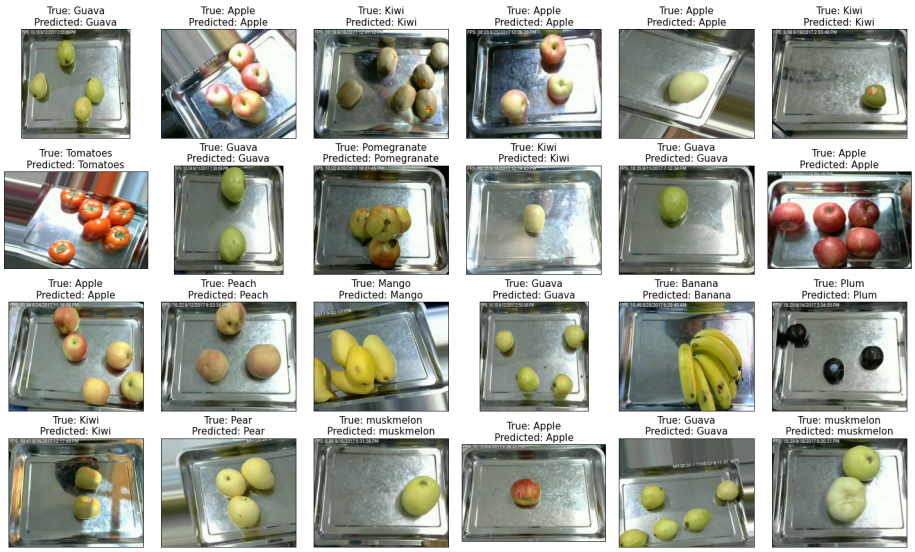


Figure 19: Some of the predictions using the transfer learning model

But after some further testing, we encountered some of the limitations of this model. First, the model did not seem to recognize an empty tray, as shown in Figure 20 below, it still predicted it as an apple image. This can be mitigated by turning off the predict function when the data from the weight is zero during the weighting process.

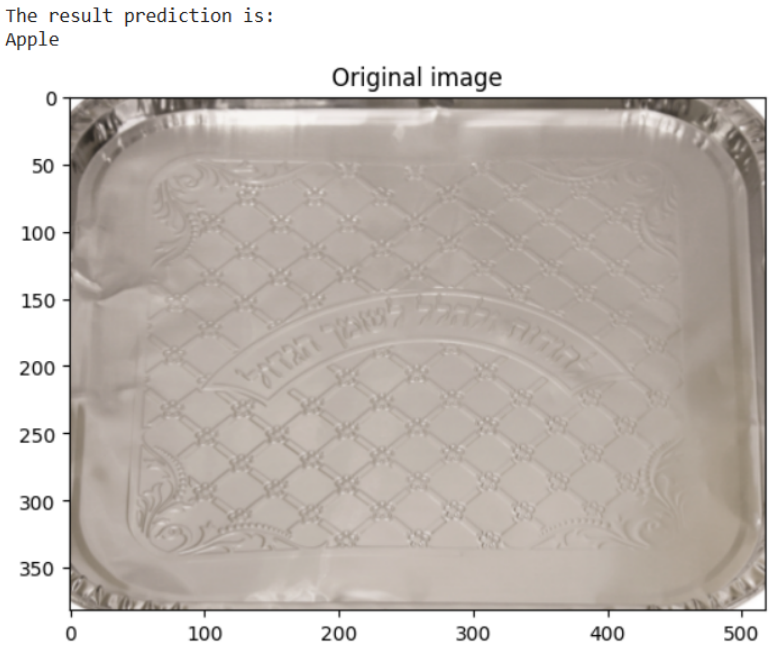


Figure 20: The model mistakenly predict an empty tray

Secondly, the model's accuracy goes down when predicting an image without a silver background like the training image. This is very expected because of bias in training. One method to bypass this limitation is to normalize the recognition environment.

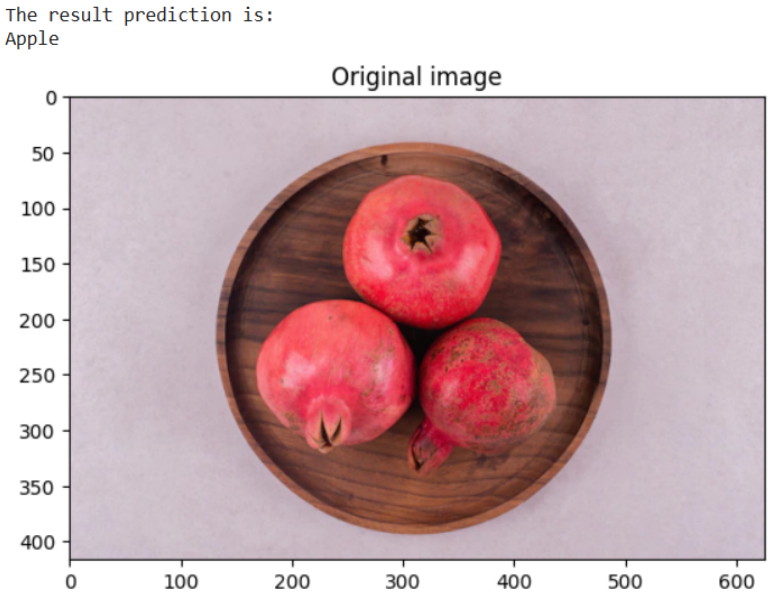


Figure 21: The model mistakenly predict pomegranate for apple

And thirdly, this is the biggest limitation of the model, it cannot recognize the image contained of multiple fruits. At the moment, we have not yet found any solutions to this problem. This is where the model can be improved and prove that we can not rely on DenseNet alone to cover all types of cases. A more difficult situation like this can be approached by implementing a segmentation section in addition to the classification model, hence the ability to detect different kinds of fruit presented on the camera at those times. Even though this is currently out of the scope of this study, given the chance and time, the project can go in the proposed way.

6.3 Application overview and performance

The weight sensor took roughly 4 seconds every time to calculate the weight of objects. The accuracy is stabilized with a low measurement error, about 0,05g.

This sketchy model processes values with relative stability, and also took up a large memory capacity because CoolTerm extracts weight values to a .txt file. However, this problem can be resolved with a modern IoT weight sensor that is built for this particular purpose, namely higher stability, and transmitting values directly to the application.

The application starts slowly because it has to load the pre-trained model and connect the external devices. The frame shown on the application got delayed for stable fps. The first process after starting always takes some time, around 4 seconds, for initializing memory allocators and the following recognition takes about 300 milliseconds(ms) to get the result, the reason for it is the application has to process many tasks at a moment. The speed of processing can be improved by using GPU and a better hard drive.

The model can mostly accurately recognize trained fruits in the tests, Sometimes, due to factors like light or fruit placement the model might recognize the wrong type of fruit.

6.4 Conclusion and future work

Technology advancements nowadays have revolutionized every aspect of our daily lives. The retail industry is no exception to this. One such technology is fruit and weight recognition, which has shown shopping experience is more efficient and convenient regarding the payment process.

Using cameras and sensors to identify fruits and vegetables, has essentially helped supermarkets to automate the check-out process. Customers could scan their products at the self-checkout, and the fruit recognition system will automatically recognize the item and its price. This process is imperative for eliminating the need for manual input, which is reported to be time-consuming and prone to human errors since a vast variety of fruit types to memorize. For items that do not have a barcode printed with weight and type details, such as fresh produce or bulk items, a weight recognition system is the key. This technology uses scales and sensors to weigh and calculate items for their weight and price. Weight recognition helps relieve supermarkets from the pre-packing process, and ensures that the purchased items are charged accurately. Therefore, this would increase buyer satisfaction and loyalty thanks to what the fruit and weight recognition system offered.

This technology also reduces the need for additional staff to manage the checkout process, which helps supermarkets to save on labor costs. Additionally, the number of errors during the payment process is reduced with the help of fruit and weight recognition, leading to increased accuracy in billing and customer satisfaction. Despite the benefits of these modern machinery, there are also challenges associated with their use. The cost of implementing the technology is foremost demanding, installing and maintaining the necessary equipment can be quite expensive, which may make it challenging for smaller supermarkets to adopt this piece of technology. Secondly, some customers might still prefer traditional ways to purchase items, or some might find dishonest ways to benefit themselves.

To conclude, fruit and weight recognition technology is beneficial and essential for payment purposes in the retail industry. While there are some challenges associated with its use, the benefits of this technology outweigh the drawbacks.

This paper's accomplishment shows the basic model of fruit and weight recognition systems:

- A CNN model capable of recognizing fruit types.
- Built up a sketch model of an IoT weight sensor and a webcam.
- Transmit data value from the weight via a third-party application, which helps to utilize memory capacity.
- Detects problems such as attempting to scan an empty tray.
- An application with a user-friendly interface, high stability, and utilized running time that connects all pieces and performs real-time recognition and price result.

However, to make it applicable in the retail industry, there are still loads of work that are required to improve such as reducing process time, working with the payment process, and security, and building a device that is a combination of camera and scale, or even a fully functional self-checkout machine with the management system.

Future work will concentrate on classifying different fruit varieties, implementing segmentation to detect mixed fruit situations, and widening the range of fruits available to the model. Additionally, for enhancing security and eliminate fraud, the model would detect obstacles or non-fruit anomalies on the scale upper platform, namely arms, hands, or non-fruit items.

For the hardware, further development would also focus on improving the weight sensor for higher accuracy and stability with more up-to-date and high-end equipment. To be specific, increasing the number of sensors (load cells), instead of one.

Computer power is also another improvement for higher accuracy, the reason is that generating more features and parameters means more computation, and the higher the computation the more powerful the machine is required. Furthermore, upgrading to a higher resolution camera would also be a betterment for the model's accuracy, resulting in more pixel information per inch (PPI) and creating a high-quality image.

Therefore, more details in each image and powerful computer resources are capable of differentiating fruits that are similar in color and shape. Last but not least, adding a QR code generator containing details, namely type, weight value, and total price to the application.

References

[1] https://mof.gov.vn/webcenter/portal/tttpen/pages_r/l/detail?dDocName=MOFUCM194832.

[2] https://www.helgilibrary.com/charts/which-country-eats-the-most-fruits/#:~:text=Total%20fruit%20consumption%20reached%20206%2C064,of%2041%2C577%20kt%20in%201962.

[3] https://www.columbiacountyny.com/weights--measures.html.

[4] Roland Geyer, Jenna R Jambeck, and Kara Lavender Law. Production, use, and fate of all plastics ever made. *Science advances*, 3(7):e1700782, 2017.

[5] Chris Gorgolewski. Fruit recognition. [Kaggle].

[6] Khurram Hameed, Douglas Chai, and Alexander Rassau. A sample weight and adaboost cnn-based coarse to fine classification of fruit and vegetables at a supermarket self-checkout. *Applied Sciences*, 10(23):8667, 2020.

[7] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[8] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017.

[9] Israr Hussain, Qianhua He, and Zhuliang Chen. Automatic fruit recognition based on dcnn for commercial source trace system. *Int. J. Comput. Sci. Appl*, 8(2/3):01–14, 2018.

[10] Hanwen Kang and Chao Chen. Fast implementation of real-time fruit detection in apple orchards using deep learning. *Computers and Electronics in Agriculture*, 168:105108, 2020.

[11] Rudnik Katarzyna and Michalski Paweł. A vision-based method utilizing deep convolutional neural networks for fruit variety classification in uncertainty conditions of retail sales. *Applied Sciences*, 9(19):3971, 2019.

[12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017.

[13] Md Robel Mia, Md Jueal Mia, Anup Majumder, Soummo Supriya, and Md Tarek Habib. Computer vision based local fruit recognition. *International Journal of Engineering and Advanced Technology*, 9(1):2810–2820, 2019.

[14] Horea Mureşan and Mihai Oltean. Fruit recognition from images using deep learning. *arXiv preprint arXiv:1712.00580*, 2017.

[15] José Naranjo-Torres, Marco Mora, Ruber Hernández-García, Ricardo J Barrientos, Claudio Fredes, and Andres Valenzuela. A review of convolutional neural network applied to fruit image processing. *Applied Sciences*, 10(10):3443, 2020.

[16] Mihai Oltean. Fruits 360. [Kaggle].

[17] K O'Farrell. 18 australian plastics recycling survey—national report. *Department of the Environment and Energy, Victoria State Government: Melbourne, VIC, Australia*, 2019.

[18] Alberto Patino-Saucedo, Horacio Rostro-Gonzalez, and Jorg Conradt. Tropical fruits classification using an alexnet-type convolutional neural network and image augmentation. In *Neural Information Processing: 25th International Conference, ICONIP 2018, Siem Reap, Cambodia, December 13-16, 2018, Proceedings, Part IV 25*, pages 371–379. Springer, 2018.

[19] Shadman Sakib, Zahidun Ashrafi, Md Siddique, and Abu Bakr. Implementation of fruits recognition classifier using convolutional neural network algorithm for observation of accuracies for various hidden layers. *arXiv preprint arXiv:1904.00783*, 2019.

[20] Woo Chaw Seng and Seyed Hadi Mirisaee. A new method for fruits recognition system. In *2009 International conference on electrical engineering and Informatics*, volume 1, pages 130–134. IEEE, 2009.

[21] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[22] Jan Steinbrener, Konstantin Posch, and Raimund Leitner. Hyperspectral fruit and vegetable classification using convolutional neural networks. *Computers and Electronics in Agriculture*, 162:364–372, 2019.

[23] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[24] Lisa Torrey and Jude Shavlik. Transfer learning. In *Handbook of research on machine learning applications and trends: algorithms, methods, and techniques*, pages 242–264. IGI global, 2010.

[25] Yu-Dong Zhang, Zhengchao Dong, Xianqing Chen, Wenjuan Jia, Sidan Du, Khan Muhammad, and Shui-Hua Wang. Image based fruit category classification by 13-layer deep convolutional neural network and data augmentation. *Multimedia Tools and Applications*, 78:3613–3632, 2019.

[26] Yudong Zhang, Shuihua Wang, Genlin Ji, and Preetha Phillips. Fruit classification using computer vision and feedforward neural network. *Journal of Food Engineering*, 143:167–177, 2014.

[27] Yudong Zhang and Lenan Wu. Classification of fruits using computer vision and a multiclass support vector machine. *sensors*, 12(9):12489–12505, 2012.