IET Journals
The Institution of Engineering and Technology

Fruit Classification Based on External Defects Using Deep Learning

ISSN 1751-8644
doi: 0000000000
www.ietdl.org

Henrique Tavares Aguiar[1] *Raimundo Cláudio da Silva Vasconcelos*[1]

[1] *Faculty of Computer Science, Federal Institute of Brasília, Brasília, Brazil*
** E-mail: henrique.aguiar@estudante.ifb.edu.br*

Abstract: The quality of fruit plays a fundamental role in their marketing and is mainly defined by its shape, color, and size. The classification process is traditionally done manually and takes time. The use of image processing techniques can help this task. Some methodologies for image classification are presented, using deep neural networks. A set of combinations between Convolution Neural Networks (CNN), deep neural networks (DNN) using Gabor filter, over RGB and grayscale images, extracting texture properties of a GLCM (Gray Level Co-occurrence Matrices) is used in this project. Background segmentation, contrast enhancement, and data augmentation are also used to improve generalization and minimize overfitting. Applying it to a set of tropical fruits resulted in an excellent set of results, above 95% on average.

1 Introduction

Fruit selection is a highly important economic activity. The traditional method of selection is made by human labor. During postharvest, some people are responsible for visually inspect fruits and classify them according to their color, weight, maturity, diseases, deformations, among other attributes.

An alternative to the traditional method is to automatize the process with the help of fruit selecting machines. They can have several conveyor belts available for carrying fruits towards their sensors. On them, data is obtained, and through software, the destination of each fruit is decided. Upon determining their destinations, the software acts on the conveyor belts to ensure that those fruits arrive correctly.

Automizing this process comes with three benefits:

- Error reduction. Given its repetitive nature, humans are more prone to errors when compared to machines;
- Cost-effectiveness. Classifying fruit fast and correctly demands a lot of training. Training people takes time, needs to be done for every new employee, and it is not certain that newly trained employees will perform the same way as more experienced employees. On the other hand, to replicate a selecting machine, building a new one and loading the same software is enough;
- Higher rate of selection. There is a maximum rate of how many fruits a human can classify per minute. A machine can easily go beyond that limit.

A more intuitive solution is to use traditional cameras to photograph fruits, while the software decides those fruits' destinations based on those images. Under this perspective, the challenges of building those fruit selecting machines involve computer vision.

Machine learning suggests the usage of deep neural networks to approximate functions of all types of complexities. The functions are built based on the union of simple computational units, and they are capable of recognizing relevant patterns to complete a task.

Therefore, the usage of machine learning is already established and offers promising results regarding the problem of detecting external defects on fruit. Pandey et al. [1] revise the literature on automatic fruit selection. The goal of this paper is to develop a computer vision algorithm as good as humans in identifying external defects on fruit with the help of deep learning.

2 Related Work

Image processing and computer vision techniques have been growing important for the fruit industry over the years, especially when applied to quality inspection tasks and defect classification.

It is presented in [2] an effort in recognizing defects on apples using a machine vision system based on three colors cameras. The apple image is segmented from its background by using a multiple threshold method, then, defect detection and counting are made upon the apple image.

Various techniques are analyzed in [3] for automatic fruit inspection with the help of computer vision. The analyzed fruits are apples and citric ones. The authors revised possible algorithms for defects detection on those fruits. The solutions are different and well suited for each type of fruit.

The proposed method in [4] uses pre-processing, segmentation, border detection, and features extraction to classify a fruit as either defective or fresh.

Fruit peel color is an important factor in identifying defects on fruit in [5]. The procedures adopted are background segmentation, identification of spots for a quality calculation, and a classification process.

A fruit defect detection in [6] is made by using some local features such as energy, homogeneity, contrast, and correlation, all from segmented and filtered images.

The purpose of the work made in [7] is to identify whether mangos are mature or not. The fruit images are obtained, then the fruits are harvested if mature. After the fruits' backgrounds are removed, features such as shape, color, and size are extracted. The result is a classification of whether the fruit is mature or not.

Another fruit external defects classification is made in [8]. The images are collected, segmented, and then a color histogram, global correlation, and local binary pattern are calculated in order to feed an SVM classifier.

The work in [9] consists of analyzing orange images and classifying them. Features such as color, texture, and shape are extracted, and then an RBPNN (Radial Basis Probabilistic Neural Network) is used to perform the classification.

Evaluated articles use some kind of pre-processing (e.g., background removal and segmentation). Direct features (color, texture, shape, size) or secondary features (color histogram, global correlation, energy, homogeneity, contrast, and correlation), or both are extracted. Aside from the pre-processing, they also use different classification techniques: algorithms based on thresholds, Support Vector Machines, or neural networks.

© The Institution of Engineering and Technology 2015

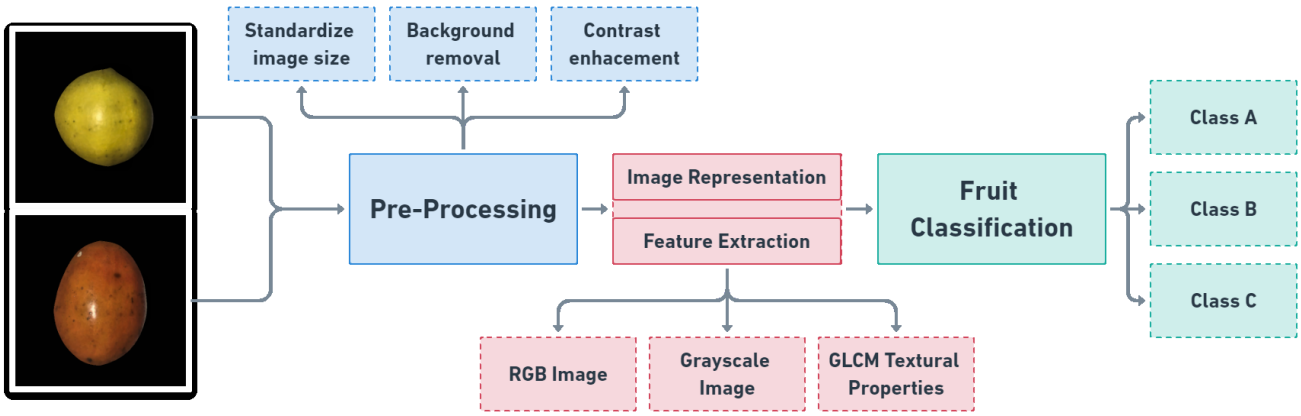


Fig. 1: Flowchart representing the whole process of fruit classification, divided into three parts: pre-processing, image representation, and the classification itself.

3 Fruits

We chose the following fruits for this paper: abiu (*pouteria caimito*); two variations of caju (*anacardium occidentale*): yellow-skinned and red-skinned; gabiroba (*campomanesia xanthocarpa*); pequi (*caryocar brasiliense*); and siriguela (*spondias purpurea*). All of them are tropical fruits usually found in Latin America. Some characteristics were taken into account, such as availability, size, shape, and colors.

4 Materials

4.1 Hardware

This work was made by using the Google Colab environment. Google Colab is a free cloud storage service in which it is possible to write and run Jupyter notebooks with support to GPU and TPU.

4.2 Software

As Google Colab uses Jupyter notebooks, the scripts were written in Python. As for the libraries, we used scikit-image that provides tools to perform image pre-processing like background segmentation and Tensorflow to build the neural networks.

5 Proposed Methodology

To enhance the quality and quantity of the agriculture product, there is a need to adopt the new technology. Image processing approach is a non-invasive technique, which provides consistent, reasonably accurate, less time consuming and cost effective solution.

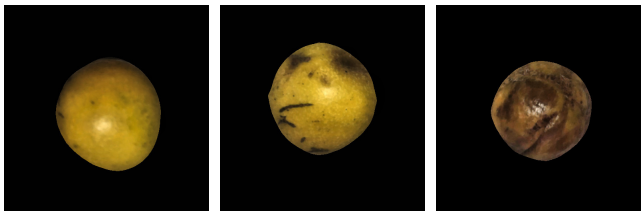
The process developed throughout this work can be summarized in four main steps:

- Manual imagery acquisition and initial classification
- Image pre-processing and segmentation
- Image representation and feature extraction
- Fruit classification

It can be better visualized in Figure 1.

6 Image Acquisition

We photographed all the fruits using the same camera, and then a group of twenty divided them into three classes (A, B, and C) accordingly to their quality. Class A represents the best fruits, almost stainless with minor defects. Class B represents mid-term fruits with a few significant defects limited only to the fruit peel, so the fruits remain edible and practically fresh inside. And Class C represents the ones with severe defects and are considered to be rotten. An example of each class for all the fruits can be seen in Figures 2 to 7.



(a) Class A (b) Class B (c) Class C

Fig. 2: Samples of *abius*

(a) Class A (b) Class B (c) Class C

Fig. 3: Samples of yellow-skinned *cajus*

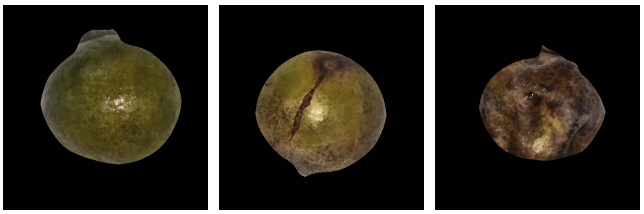
(a) Class A (b) Class B (c) Class C

Fig. 4: Samples of red-skinned *cajus*

7 Segmentation & Pre-processing

Figure 8 presents the first process, background segmentation. We chose this approach because the background contains irrelevant

© The Institution of Engineering and Technology 2015



(a) Class A (b) Class B (c) Class C

Fig. 5: Samples of *gabirobas*

(a) Class A (b) Class B (c) Class C

Fig. 6: Samples of *pequi*

(a) Class A (b) Class B (c) Class C

Fig. 7: Samples of *siriguelas*

information regarding the fruit. Thus, removing it will lead to better results.

Figure 8a shows images resized to 512x512 pixels. Figure 8b shows the phase in which images are being prepared for a contour detection algorithm. This phase was divided into two separate ways, the first one called plain method" smooths the image using a median filter, the other one called "erosion method" is more complex. It performs a morphological reconstruction by erosion on the image, in other words, low-intensity values replace high-intensity values and are limited to a minimum value. Alternatively, this reconstruction can be seen as a way to isolate connected regions of an image, in this case, the fruit and the background. Also, a median filter is applied.

In general, the erosion method is a faster but slightly less accurate method when compared to the plain method. When the background is too noisy, the erosion method most of the time is unable to correctly find the contours. However, when the shape of a fruit is too irregular, the contour found by using the plain method may not exactly fit the actual shape of that fruit. The erosion method is preferred over the plain method due to its time and computational efficiency.

The Active Contour Model [10] is used as shown in Figure 8c to find the contours of a fruit in an image. It is applied to the image obtained from the previous step. It works by minimizing the energy of an initial spline, seen as the red dotted line in Figure 8c, guided by both the image and shape of the spline, fitting onto nearby lines and edges. The result can be seen as the blue line in Figure 8c.

The image in Figure 8d is simply a mask image built from the contour calculated previously, filling its area. Alternatively, it is the shape of the fruit.

After that, in Figure 8e, a basic morphological erosion is applied on the mask image to reduce the shape's area, aiming to discard possible shadows and a portion of its area on the edges that may be misleading due to illumination differences.

Finally, in Figure 8f, the mask image is applied to the image from the first step. Also, a few improvements are made to the image: gamma adjustment, intensity rescaling and contrast enhancement.

8 Classification

8.1 Feature Extraction

There are several ways to represent an image and, for this work, the following ones were chosen: RGB image, grayscale image, and image representation through features.

The first method to represent the images is in the form of an RGB image. It can be considered the default method, since it does not require any additional computation after the pre-processing phase.

The second one transforms an image into a grayscale image, as the colors may not only be irrelevant but also misleading, and the difference in lighting between a healthy and a rotten fruit peel may be enough to classify the fruits correctly.

The third and last one represents an image through some textural properties from a Gray Level Co-occurrence Matrix (GLCM) [11]. A GLCM is a matrix defined over an image, and it represents the distribution of co-occurring gray values (intensity) at a given offset. In other words, it calculates the spatial distribution of intensity values in a neighborhood from a one-color channel image, like a grayscale image, which is why a GLCM is good to obtain textural information about an image.

For an image I of size $N \times N$, a co-occurrence matrix M of order $N_g \times N_g$ can be defined as:

$$M(i,j) = \sum_{x=1}^{N}\sum_{y=1}^{N} \begin{cases} 1 & \text{if } I(x,y) = i \text{ and} \\ & \quad I(x+\Delta_x, y+\Delta_y) = j, \\ 0 & \text{otherwise,} \end{cases} \tag{1}$$

In which i and j are the intensity values, x and y are the spatial coordinates. The offset (Δ_x, Δ_y) specifies the distance between the pixel-of-interest and its neighbors. This offset can also be parameterized in terms of a distance d and an angle θ. Note that when $d = 1$, it is called a normalized co-occurence matrix. Having that said, $m(i,j)$ will be referenced as the (i,j)-th entry in a normalized co-occurence matrix.

In order to calculate the textural properties yet to be mentioned, the following equations will be needed:

$$m_x(i) = \sum_{j=1}^{N_g} m(i,j), \quad m_y(j) = \sum_{i=1}^{N_g} m(i,j)$$

$$\mu_x = \sum_{i=1}^{N_g} i \cdot m_x(i), \quad \mu_y = \sum_{j=1}^{N_g} j \cdot m_y(j)$$

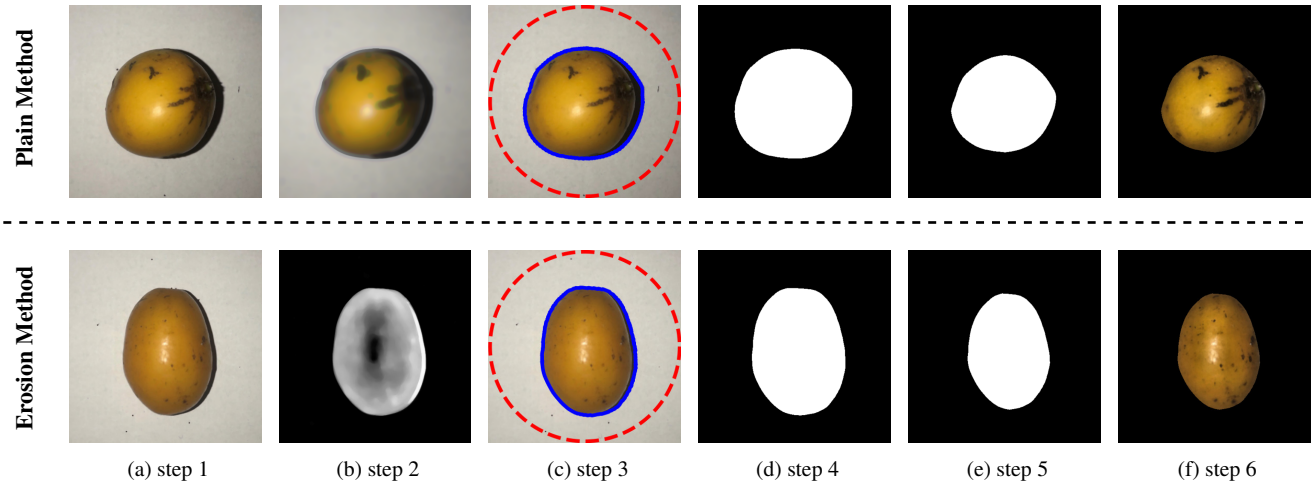
$$\sigma_x^2 = \sum_{i=1}^{N_g} (1-\mu_x)^2 \cdot m_x(i), \quad \sigma_y^2 = \sum_{j=1}^{N_g} (1-\mu_y)^2 \cdot m_y(j)$$

Concerning the GLCM method, to calculate the textural information necessary to classify the fruits, the following statistical descriptors will be used:

Contrast— measures the intensity contrast between a pixel and its neighbours over the whole image:

$$Contrast = \sum_{i=1}^{N_g}\sum_{j=1}^{N_g} m(i,j) \cdot |i-j|^2 \tag{2}$$

© The Institution of Engineering and Technology 2015



(a) step 1 (b) step 2 (c) step 3 (d) step 4 (e) step 5 (f) step 6

Fig. 8: Fruit images pre-processing, which includes contrast enhancement and background removal.
c Both lines were drawn on the image from the first step only for visualization purposes.

Dissimilarity— measures the distance between pairs of pixels in a region of interest:

$$Dissimilarity = \sum_{i=1}^{N_g}\sum_{j=1}^{N_g} m(i,j)\cdot|i-j| \tag{3}$$

Homogeneity— measures the closeness of the distribution of elements in the GLCM to the GLCM diagonal:

$$Homogeneity = \sum_{i=1}^{N_g}\sum_{j=1}^{N_g} \frac{m(i,j)}{1+|i-j|^2} \tag{4}$$

Angular Second Moment (ASM)— Provides the sum of the squared elements in the GLCM, it is used as a measure of orderliness:

$$ASM = \sum_{i=1}^{N_g}\sum_{j=1}^{N_g} m(i,j)^2 \tag{5}$$

Energy— Similar to the ASM, however it is considered a better texture measurement:

$$Energy = \sqrt{ASM} \tag{6}$$

Correlation— measures how correlated a pixel is to its neighbour over the over whole image:

$$Correlation = \sum_{i=1}^{N_g}\sum_{j=1}^{N_g} \frac{m(i,j)\,(1-\mu_x)\,(1-\mu_y)}{\sigma_x\cdot\sigma_y} \tag{7}$$

Considering that a GLCM uses gray levels values or intensity values, an RGB image cannot be used as an input, neither a grayscale image, as it loses relevant information about each color channel. So far, 3 GLCMs are needed, one for each channel in the RGB color scheme. In addition, as mentioned above, the distance used is one pixel, moreover, four angles were used: $0°$, $45°(\frac{\pi}{4})$, $90°(\frac{2\pi}{4})$, and $135°(\frac{3\pi}{4})$. A combination between an angle and a distance makes an offset. Now, 4 GLCMs are needed for each color channel. Therefore, in total, 12 GLCMs need to be built. Then, as 6 textural descriptors are calculated from one GLCM (2) to (7). To represent a single image, 72 textural descriptors are calculated.

Moreover, for every method discussed above, a Gabor filter is applied. If it is a grayscale image, the Gabor filter is applied after the image is converted into grayscale. If a GLCM is used, the filter is applied to the image before the GLCM is calculated.

A Gabor filter can be defined as a sinusoidal signal of a particular frequency and orientation, modulated by a Gaussian wave. In other words, it allows certain frequencies and rejects others. When a Gabor Filter is applied to an image, it emphasizes the edges and points where texture changes accordingly to the filter pattern, i.e., its parameters. Therefore, these Gabor Filters are well suited for texture analysis. Mathematically, a Gabor Filter has a real and imaginary component representing orthogonal directions. For this work, only the real component was used.

8.2 *Neural Network*

There were 120 samples of each of the six fruits, i.e., 720 images in total, and all the fruit images were split equally into those three classes. In other words, 40 images for each class (A, B, or C). 80% of fruit images were used to train the models and the remaining 20% for the validation step. Since there were defined very distinct ways to represent the images, it demands different neural networks architectures to fit better the data fed into them.

The architecture seen in Figure 9 represents the structure of a Convolution Neural Network (CNN). A CNN is mainly composed of four different layers: Convolution, Pooling, Flattening, and Classification.

Convolutional layers allow one to systematically create and apply a series of filters over an array input (e.g., an image). These filters act like feature detectors, and then the layer outputs a "feature map" that summarizes the features detected (e.g., lines and edges) by the filters. Although the Convolutional layers have their limitations, they are translation-invariant, because they are capable of detecting patterns locally rather than globally. It means that if a Convolution layer leaned a certain pattern, and if the same pattern occurs again, but in a different location, they would still be able to detect that same pattern.

A Pooling Layer is used to perform a downsample on feature maps, which reduces their dimensions, but they still carry the main elements. It works by aggregating a group of pixels and applying a pooling operation. The two most commons are Average Pooling, which calculates the average of each group, and Max Pooling, which calculates the maximum value in each group. The purpose of using a Pooling Layer is to decrease the computational power needed to process the data. And as a side-effect, it helps to extract dominant features, making the training process more efficient.

Before the Pooling layer, an activation function (e.g., ReLU (Rectified Linear Unit)) is used upon the features maps. This function usually breaks the linearity and adds complexity, much like it is done for a Dense Layer.

© The Institution of Engineering and Technology 2015

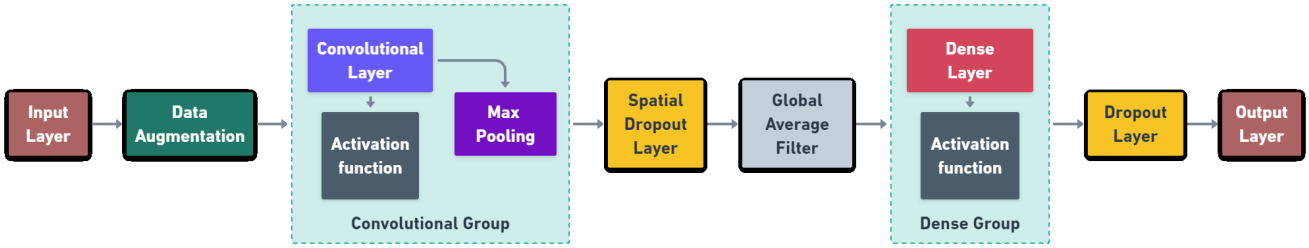


Fig. 9: Represents the structure of the overall structure of the convolutional neural network models used.

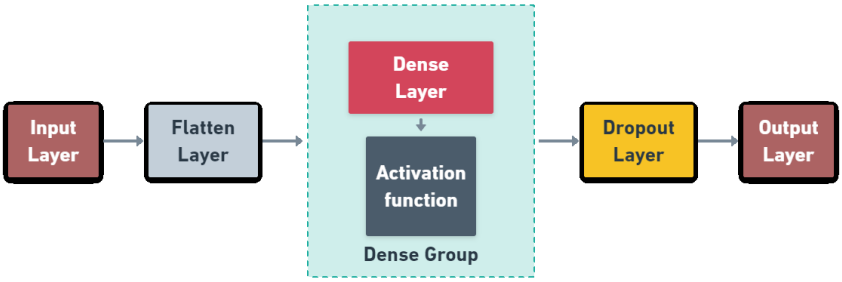


Fig. 10: Represents the overall structure of the deep neural network models used.

The combination between a Convolutional Layer, a Pooling Layer, and an activation function makes a Convolution Group, seen in Figure 9. When stacking these Convolutional Layers, instead of detecting simple features (e.g., lines and edges), deeper layers learn to detect more abstract features, like shapes or specific objects.

After that, a Flattening Layer is used because the output of the combination of multiple Convolutional Groups is multi-dimensional, composed of several features maps. So, it needs to be flattened.

Finally, the Classification part is composed of a sequence of Dense Groups. A Dense Group is made of Dense Layers, also known as Fully Connected Layers, followed by an activation function. Before the Output Layer, a Dropout Layer is used to help reducing overfitting while training (useful for a small dataset). It works by randomly deactivating a percentage of the neurons from the previous layer. Then for the Output Layer, which is also a Dense Layer, the activation function chosen is often different from the ones used in the hidden layers.

The other architecture, DNN, seen in Figure 10, is a simple deep neural network composed of a Flattening Layer at the beginning, a sequence of Dense Groups, and a Dropout Layer right before the Output Layer. This architecture may look very similar to the one used in the Flattening and Classification part inside the CNN. However, in reality, a CNN uses a combination of a convolutional technique to extract useful information from images and passes it as input to a fully connected deep neural network.

Data Augmentation technique was used at the beginning of the CNN seen in Figure 9. Since there was a total of 120 images for each fruit, and each model was trained for only one type of fruit, meaning the models were on a small dataset. It helps by reducing overfitting by generating new images from existing ones by doing simple modifications such as random horizontal and vertical flipping with a random rotation. For the same reason, a Spatial Dropout Layer was used right before the Flattening part. It works similarly to a regular Dropout Layer. However, it is applied to 2D feature maps instead of individual elements.

Regarding the Flattening Layer in the CNN, a Global Average Filter was used. It uses an average pooling to reduce the size of feature maps, and then it flattens and passes it as input to the next part. This Global Average Filter is important because its input came from either RGB or Grayscale image methods. Thus, if it were simply flattened, there would be too much information for the dense neural network, i.e., the Classification part, to process and classify efficiently. That was unnecessary for the DNN since its input came from the GLCM textural properties, and it consisted of only 72 elements, the GLCM descriptors when flattened.

Inside the Convolutional Groups, the activation functions used were the ReLU (8) or the swish (9) function, which behaves similarly to the ReLU. Concerning the Dense Groups, they use either the swish or the ELU (Exponential Linear Unit) (10) function, which is also similar to the ReLU.

The softmax (11) function is used at the Output Layer because it converts its input into a probability distribution. Useful for classification tasks when there are more than two classes.

$$ReLU(x) = max(0, x) \tag{8}$$

$$swish(x) = \frac{x}{1 + \mathrm{e}^{-x}} \tag{9}$$

$$ELU(x) = \begin{cases} x & \text{if } x > 0, \\ \mathrm{e}^{x} - 1 & \text{otherwise,} \end{cases} \tag{10}$$

$$softmax(x_i) = \frac{\mathrm{e}^{x_i}}{\sum_{j=1}^{K} \mathrm{e}^{x_j}} \tag{11}$$

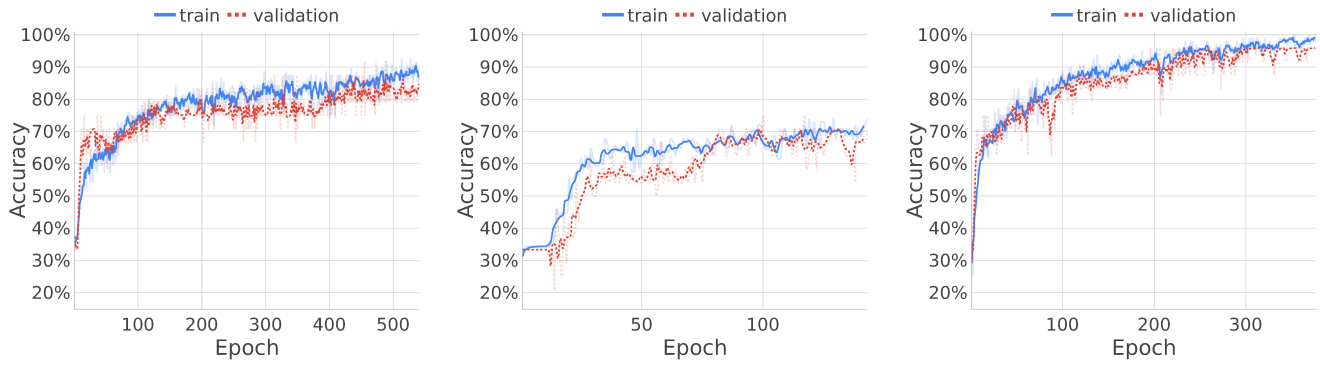
9 Results & Discussion

In this section, we will present the results of the combination between the various imaging representation methods, their appropriate neural network architectures along with a Gabor Filter. Experiments were made using all of the six fruits separately, they can be seen in Figures 11 to 16.

Across all the graphs, the y-axis represents the accuracy (training and validation) achieved by the model, while the x-axis represents the epochs from when the models were being trained. Those models were trained for different epochs, due to a technique called Early Stopping, which stops the training when there is no improvement in loss, which is a measure of how wrong a model is, for a number of consecutive epochs.

Analyzing the graphs, it is clear that the GLCM textural properties-based models outran the image-based ones. Between the image-based models, the ones that used RGB images managed to be slightly better in general. It is also visible a strange behavior on various graphs. It tends to occur more often in the GLCM based models. And this strange behavior is when a model achieved better results on the validation set. A possible explanation is when the validation set

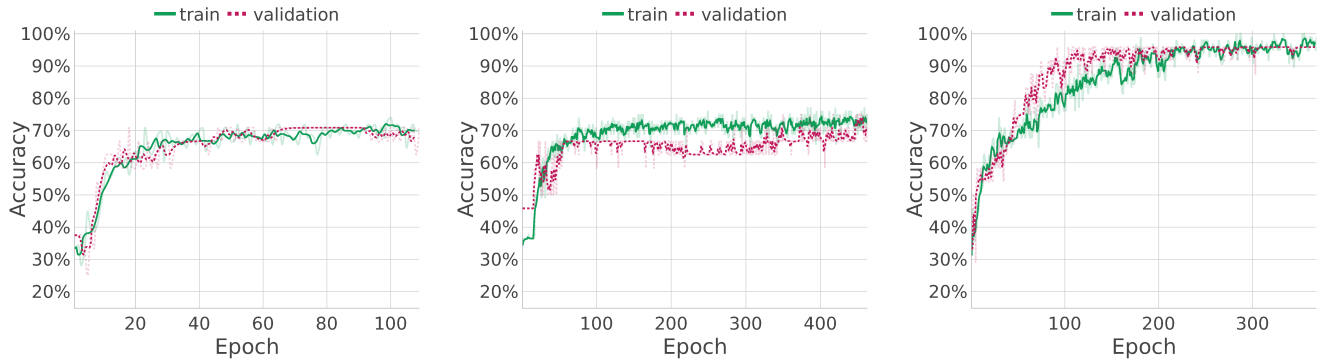
© The Institution of Engineering and Technology 2015



(a) RGB (b) Grayscale (c) GLCM

Fig. 11: Training curves for the models trained on *abius* images.

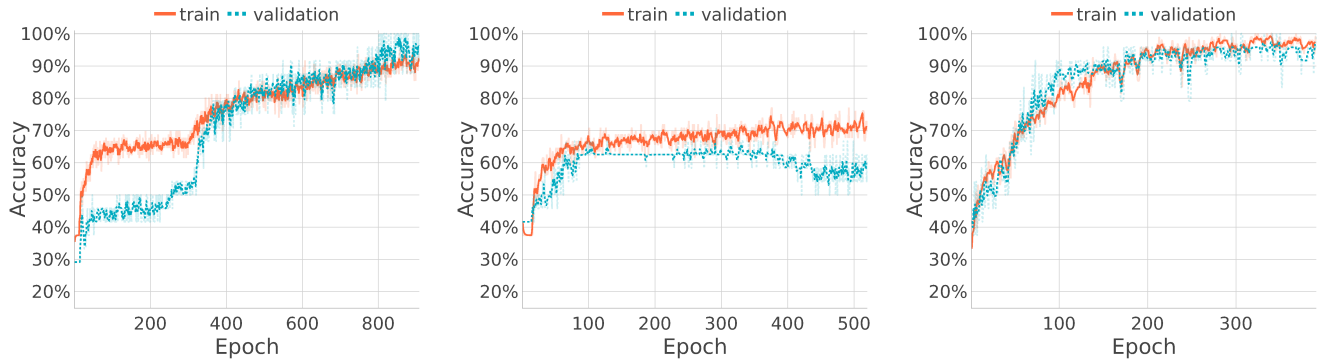
a It was used RGB images.
b It was used grayscale images.
c It was used some texture properties from a GLCM.



(a) RGB (b) Grayscale (c) GLCM

Fig. 12: Training curves for the models trained on *cajus amarelos* images.

a It was used RGB images.
b It was used grayscale images.
c It was used some texture properties from a GLCM.

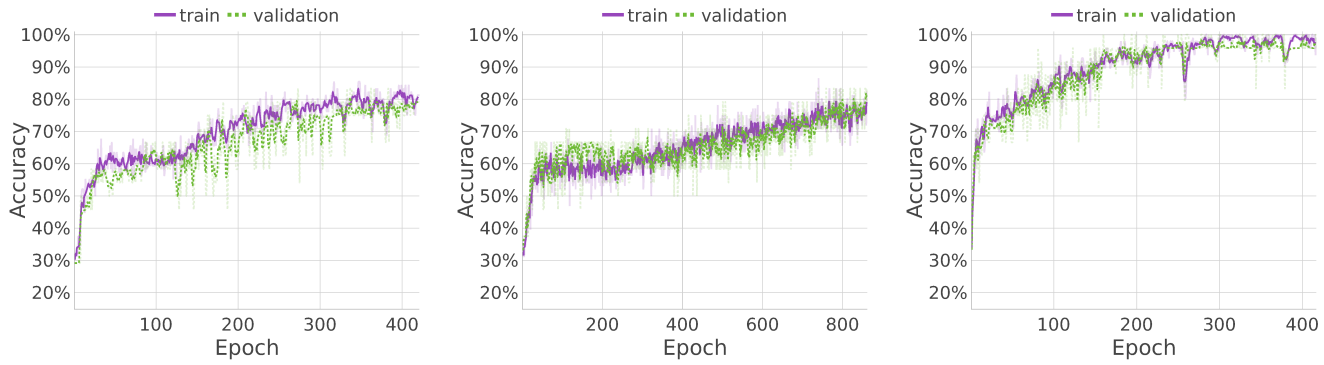


(a) RGB (b) Grayscale (c) GLCM

Fig. 13: Training curves for the models trained on *cajus vermelhos* images.

a It was used RGB images.
b It was used grayscale images.
c It was used some texture properties from a GLCM.

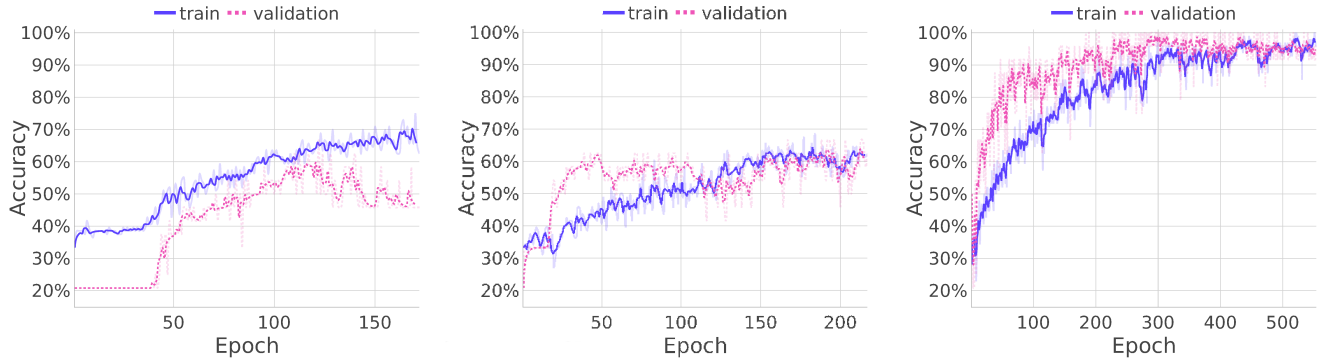
© The Institution of Engineering and Technology 2015



(a) RGB (b) Grayscale (c) GLCM

Fig. 14: Training curves for the models trained on *gabirobas* images.

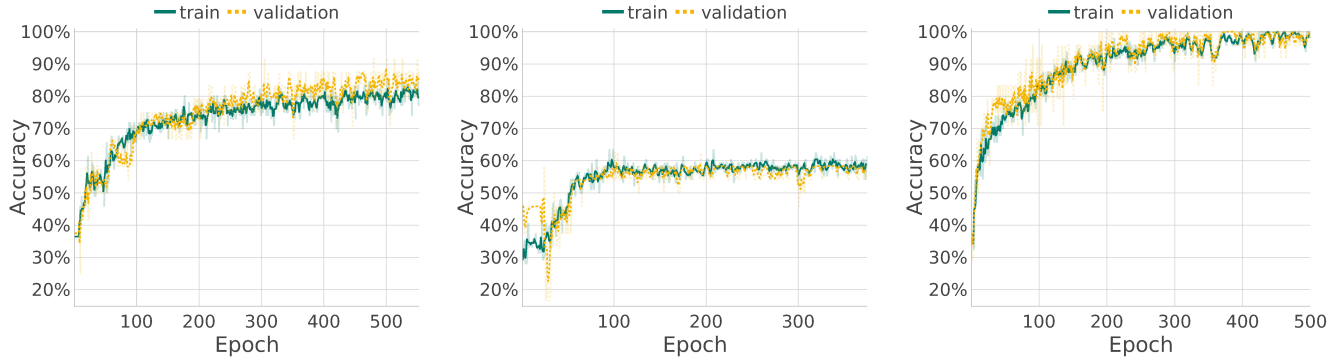
a It was used RGB images.
b It was used grayscale images.
c It was used some texture properties from a GLCM.



(a) RGB (b) Grayscale (c) GLCM

Fig. 15: Training curves for the models trained on *pequis* images.

a It was used RGB images.
b It was used grayscale images.
c It was used some texture properties from a GLCM.



(a) RGB (b) Grayscale (c) GLCM

Fig. 16: Training curves for the models trained on *siriguelas* images.

a It was used RGB images.
b It was used grayscale images.
c It was used some texture properties from a GLCM.

© The Institution of Engineering and Technology 2015

is too easy, which is completely plausible due to the small datasets used.

Table 1 Train and validation accuracy results for all fruits presented so far, restrited to only GLCM results. Values in bold show which fruits obtained the best results.

Fruit	GLCM - Accuracy	
	Train	Validation
Abiu	99.01%	95.83%
Yellow-skinned Caju	97.40%	95.83%
Red-skinned Caju	97.42%	96.75%
Gabiroba	97.67%	96.84%
Pequi	97.04%	94.77%
Siriguela	**99.42%**	**98.10%**

A closer look at the best results across all the fruits, i.e., GLCM textural properties with a DNN, can be seen in Table 1. First of all, it is safe to assume the results were overall great, since every result is above 95%, except for the *pequi* fruit. And the *siriguela* fruit was the one that achieved the best accuracies on both training and validation environments.

Table 2 Train and validation loss results for all fruits presented so far, restrited to only GLCM results. Values in bold show which fruits obtained the best results.

Fruit	GLCM - Loss	
	Train	Validation
Abiu	0.0534	0.1151
Yellow-skinned Caju	0.0955	0.1617
Red-skinned Caju	0.0734	0.0785
Gabiroba	0.0650	0.0929
Pequi	0.0779	0.0995
Siriguela	**0.0281**	**0.0418**

Now, analyzing this other measurement called loss in Table 2, which is basically a measurement of how wrong a model is. The *siriguela* once again achieved the best results with the lowest loss values in both train and validation environments. But the loss values for the other fruits are not bad, since they are somewhat proportional to the accuracies values, and the validation losses are close to the train losses. Which are good indicators that the models have not over-fitted, that is, memorized the training set.

Also, among all the graphics, those extracted from the *siriguelas* have more consistent and smoother lines.

10 Conclusions & Future Scope

In this work, we presented some solutions related to the problem of classifying fruit according to their quality. The solutions presented included: a CNN with an RGB or grayscale image; and a DNN with GLCM textural properties. A Gabor filter was applied to all of them.

The DNN with GLCM textural properties outperformed the other solutions for all the six fruits, with significantly better results in most cases. This means that texture is a key factor when analyzing fruits based on their appearance.

Table 3 A collection of similar works, fruit classification, displaying their results and accuracy.

Paper	Classification method	Accuracy
[2]	Counting of blobs found with a multiple threshold method	89%
[8]	Multi-class SVM	93%
[9]	RBPNN	88%

This solution proved to be effective, as even with a small dataset, i.e. 120 samples, still managed to achieve great accuracies as stated in the previous section, and also when comparing the results of previous works as seen in Table 3.

In the future, this fruit quality detection proposal should be extended to other tropical fruits with economic potential. Furthermore, this proposal should be compared with other automated techniques and some new parameters or features can be added. The proposed model is capable of detecting the quality of a fruit at a time and this can be expanded in to detect multiple fruits of different types at the same time. The efficiency in detect the precision level of fruit quality can be increased and time consuming can be reduced to a short period of time. The classifier can also be controlled via cell phone application.

11 References

1 Pandey, R., Naik, S., Marfatia, R.: 'Image processing and machine learning for automated fruit grading system: A technical review', *International Journal of Computer Applications*, 2013, **81**, pp. 29–39

2 Xul, Q., Zou, X., Zhao, J. 'On-line detection of defects on fruit by machinevision systems based on three-color-cameras systems'. In: Zhao, C., Li, D., editors. Computer and Computing Technologies in Agriculture II, Volume 3. (Boston, MA: Springer US, 2009. pp. 2231–2238

3 Devi, P., Vijayarekha, K.: 'Machine vision applications to locate fruits, detect defects and remove noise: A review', *Rasayan Journal of Chemistry*, 2014, **7**, pp. 104–113

4 Arunachalam, S., Kshatriya, H.H., Meena, M.: 'Identification of defects in fruits using digital image processing', *International Journal of Computer Sciences and Engineering*, 2018, **6**, pp. 637–640

5 Moradi, G., Shamsi, M., Sedaghi, M.H., Alsharif, M.R. 'Fruit defect detection from color images using acm and mfcm algorithms'. In: 2011 International Conference on Electronic Devices, Systems and Applications (ICEDSA). (, 2011. pp. 182–186

6 Yogesh, Dubey, A.K. 'Fruit defect detection based on speeded up robust feature technique'. In: 2016 5th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO). (, 2016. pp. 590–594

7 Sahu, D., Potdar, R.M.: 'Defect identification and maturity detection of mango fruits using image analysis', *American Journal of Artificial Intelligence*, 2017, **1**, pp. 5–14

8 Dubey, S.R., Jalal, A.S.: 'Adapted approach for fruit disease identification using images', *International Journal of Computer Vision and Image Processing (IJCVIP)*, 2012, **2**, pp. 44–58

9 Capizzi, G., Sciuto, G., Napoli, C., Tramontana, E., Woźniak, M.: 'A novel neural networks-based texture image processing algorithm for orange defects classification', *International Journal of Computer Science and Applications*, 2016, **13**, pp. 45–60

10 Kass, M., Witkin, A., Terzopoulos, D.: 'Snakes: Active contour models', *International Journal of Computer Vision*, 1988, **1**, (4), pp. 321–331

11 Haralick, R.M., Shanmugam, K., Dinstein, I.: 'Textural features for image classification', *IEEE Transactions on Systems, Man, and Cybernetics*, 1973, **SMC-3**, (6), pp. 610–621

© The Institution of Engineering and Technology 2015