Fruit Recognition

Joel Andersson
University of California San Diego
La Jolla, CA
janderss@ucsd.edu

Eskil Jarlskog
University of California San Diego
La Jolla, CA
ejarlsko@ucsd.edu

Richard Wang
University of California San Diego
La Jolla, CA
rsw014@ucsd.edu

***Abstract*—This report presents results from training two models on three dataset based on the same training data but where features are extracted differently. The aim with this project is to gain a better understanding of these methods and techniques and compare the results of the different model and feature combinations. The models used for classifying is Support Vector Machine (SVM) and Random Forest (RF). Feature extraction methods include Principal Component Analysis (PCA), custom features based on color and shape and lastly Discrete Fourier Transform (DFT) together with PCA. The final accuracies ranges from 80%-97%.**

***Index Terms*—fruit recognition, fruit detection, machine learning**

I. Introduction

During the last few years, image recognition and machine learning have become sophisticated and mature to the extent of being implemented in every day commercial devices such as smartphones. With applications and services aiming to recognize a wide array of different objects, there follows a need of training algorithms on wide varieties of different sets of objects.

A dataset of 28736 images was used to train established and well-understood algorithms with the aim of classifying each image to one of the 60 predetermined classes (fruits) and compare the accuracy of such predictions to results obtained with more sophisticated deep learning methods from previous work. RGB values in the images was used to extract three features sets. The first set was obtained from Principal Component Analysis (PCA) where 24 components were selected, the second set consisted of nine custom features derived from the two physical attributes color and shape, and the third set consisted of a two-dimensional Discrete Fourier Transform of the images followed by PCA. We then use Support Vector Machine (SVM) and Random Forest (RF) implemented in the Scikit library on each of the three sets to train in total six different models and obtain accuracies ranging from 80-97%.

II. Related Work

We draw inspiration from work presented in this paper [3] where a convolutional neural network with multiple layers including but not limited to convolutional, pooling and ReLU layers, was used to train a predictive algorithm on the same dataset as used in this paper. The final accuracy achieved by the model is 96.3%.

A purely statistical approach based on color and texture information has been explored to classify fruits, this paper [4] presents a method were 13 color and textured features are derived from the dataset and empirical statistical measures based on moments are derived. Minimum distance criterion are used to classify fruits and accuracies achieved on features solely based on either color or texture are concentrated to the ranges of 30-60% and 60-80% respectively. Combined together the algorithm achieves consistent results of around 90% on different classes.

A similar approach to ours has also been investigated and presented in this paper [6]. The pre-processing stage consists of simply resizing images to 90×90 pixels. Two sets of training sets based on two different feature extraction techniques were created. First set is extracted with color features based on statistical moments and shape features based on Centroid, Eccentricity and Euler Number. Second set uses The Scale Invariant Transfom (SIFT) to extract features. K-nearest neighborhood (KNN) and support vector machine (SVM) algorithms are run on the two datasets to achieve accuracies ranging from 85-100% depending on traning set and algorithm combinations.

Another report [5] investigating KNN algorithm performance based on color, shape and size features reported accuracy up to 90%

III. Dataset and features

The dataset consisted of 38409 images of fruits and was taken from Kaggle [3]. The training set consisted of 28736 images while the validation set consisted of 9673 images. All images were 100×100 pixels and has white background. The fruits were filmed while rotating around a fixed axis for either two or three different axises per fruit. Then the training set and validation set were obtained by randomly splitting all the images into a set with 3/4 of the images being in the training set and 1/4 being in the validation set.

The dataset consisted of fruits from 60 different classes, and the training set contained around 500 images from every class. Four fruits from different classes can be seen in figure 1 below, the classes are specified in the figure text.

Fig. 1. A visualization of images from the training set from four different classes (from left, 'Apple Red 3', 'Banana', 'Huckleberry' and 'Papaya').

For every class pictures were taken of the fruit from different angles. Four different images of the same class ('Apple red 3') is visualized in figure 2 below.

Fig. 2. A visualization of four images from the training set from the class 'Apple Red 3'. This figure shows four images taken from different angels of the same fruit.

The validation set consisted of images that were very similar to the ones in the training set. In figure 3 below, instances of the classes shown in figure 1 have been taken from the validation set to visualize the similarity between the training set and the validation set.

Fig. 3. A visualization of images from the validation set from four different classes (from left, 'Apple Red 3', 'Banana', 'Huckleberry' and 'Papaya'). Compare with figure 1.

IV. Method

Data Preprocessing

Because of the dataset coming from a relatively controlled environment, limited pre-processing was needed. Using the openCV library in Python, all images were rescaled to 45×45 pixels and the RGB pixel values in each image transformed into a flattened vector.

Feature extraction

Three different sets of features were created using Principal Component Analysis (PCA), custom feature extraction directly from the image and Fourier transform with PCA. The first technique involved projecting the data into a lower dimensional subspace using PCA, the second technique involved creating nine features based on color and shape and lastly the third technique was transforming each color channel into frequency space followed by performing PCA on the new image.

Methods used

Implementations of Support Vector Machine (SVM) and Random Forest (RF) in the SciKit library were used to train the fruit classifier. This section briefly explain the underlying theory of the methods used, including PCA and DFT.

A. Principal Component Analysis

PCA is often used to either increase the computability of a dataset or mitigate the so called "curse of dimensionality" when working with datasets in higher dimensions. The process involves finding principal components that are orthogonal to each other and accounts for as much variability as possible, i.e. explains as much of the variance in the dataset as possible. This is the same thing as finding orthogonal eigenvectors with the largest eigenvalues to the covariance matrix of the data. To gain even more intuition about the process, one can imagine the process of fitting an ellipsoid of the same dimensionality as the data where the axises represent the principal components. A longer axis implies a component that explains the variability more than a shorter axis. These axis are later normalized to unit vectors and forms a new lower dimension coordinate system where the data are projected to. Mathematically, the processes of finding principal components reduces to the optimization problem seen in equation 1 below [1].

$$\frac{1}{N}\sum_{n=1}^{N}(u_1^T x_n - u_1^T \bar{x})^2 = u_1^T S u_1 \tag{1}$$

Where $\bar{x}$ dentoes the mean of the data, S the covariance matrix and u_1 the variable to be maximized. It is a so called greedy algorithm where it seeks to maximize the next component in the iteration. That is, it does not necessarily maximize explained variability for a given number of components.

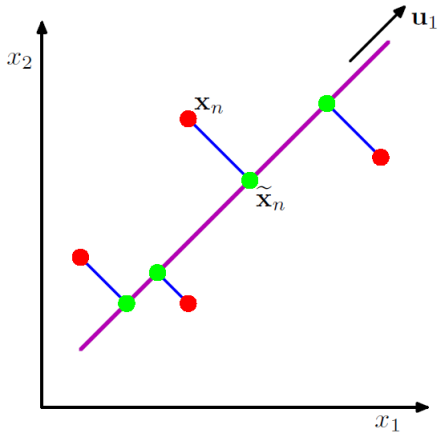

Fig. 4. Illustration of data points projected down on one component. Figure from p. 561, Bishop, M. (2006) *Pattern Recognition and Machine Learning*. Springer. [1]

B. 2D Discrete Fourier Transform

The Discrete Fourier Transform (DFT) is a transformation in which a complex sequence $X = \{x_0, x_1, ..., x_{N-1}\}$ is transformed into another complex sequence $X' = \{x'_1, x'_2, ..., x'_{N-1}\}$ according to:

$$x'_k = \sum_{n=0}^{N-1} x_n e^{-\frac{2\pi i}{N} kn} \tag{2}$$

In practical terms, the DFT decomposes a complex sequence into a new sequence of the same size where each element in the new sequence corresponds a frequency, with its magnitude indicates its contribution toward the original sequence.

The DFT can be naturally extended to 2D-dimensional data (e.g. images) where we use similar notation with the exception that elements of X and X' are now indexed by tuples and are assumed to be of size $M \times N$.

$$X'_{(k,l)} = \frac{1}{MN} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} X_{(m,n)} e^{-2\pi i (\frac{k}{M} m + \frac{l}{N} n)} \quad (3)$$

This form allows us to decompose any image into its frequency components, where each such component has a direction and frequency.

C. Custom features

The nine features that were extracted manually from the image were the mean and variance of selected RGB channels in the image (six features) as well as three features derived from the shape of the fruit.

The color features only took into consideration pixel values which were deemed not to be white, so as not to have the background impact the features. The idea behind extracting the mean RGB values is quite intuitive: a banana and a strawberry most certainly differ in color and generally color is a means of differentiating fruit. By also including the variance one can also take into consideration how uniform the color of the fruit is in each RGB component, which also should supply a means to differentiating fruit.

The shape features were extracted by first finding the biggest connecting component after having binarized the picture and then to calculate different measures for this component. Useful measures were properties of an ellipse fitted to the component and the component's area. The shape features that were ultimately extracted were:

1) the ratio of the minor and major axis of a fitted ellipse
2) the ratio of the area of the component and a fitted ellipse
3) the ratio of the area and the circumference of the connected component

D. Support Vector Machine

SVM is a non-probabilistic approach to classify data by maximizing the distance to the boundary from the nearest data point in each class. SVM is inherently a binary classifier but can be extended and generalized into a multi class classifier. We briefly discuss the theory for the binary case.

In order to partition the space where the data lies in, the data has to be linearly separable. Often, this is not the case, there are two common remedies to the problem. The first one involves projecting the data to a higher dimensional space where the data might be linearly separable. This is often done in practice with the so called "kernel trick" where the data are implicitly mapped to higher dimension space using a suitable kernel function.

Since data is rarely perfectly linear separable even in higher dimension, a soft margin is usually used in practice where a regularizing parameter is added to the objective function to allows some leeway for misclassification. The tuning of the regularizing parameter involves the usual bias-variance trade-off optimization. Letting $y_1 = sign(wx_i - b)$, w, b, denoting parameters for the separating hyperplane and λ denoting the regularizing parameter the error function can be written as[1]:

$$\frac{1}{N} \sum_{n=1}^{N} \max(0, 1 - y_i(wx_i - b)) + \lambda ||w||_2 \quad (4)$$

E. Random Forest

The Random Forest algorithm involves a procedure where decision or regression trees are fitted to a randomly chosen subset of the training set, a process that is often called tree bagging. The Random Forest algorithm adds a second stochastic element in the algorithm where not only a training subset is selected randomly but also a subset of the features are selected in each tree iteration.

The main reason that warrants this technique is the tendency to overfit the models to the data, this is especially true for deep trees. The two stochastic elements in the procedure alleviates the overfitting, i.e. reduces the variance to a relatively small bias increase.

The optimal number of trees can be found by e.g. cross validation and the final prediction of the model can be given by either taking the mean of the predictions made by the individual trees or by simple majority vote.

V. Results

A. Principal Component Analysis (PCA)

By iteratively testing it was found that the optimal number of components for the PCA was 24. This number was found by testing systematically, so a slight increase or decrease in the number of components would make the results worse.

In figure 5 below one can see how what proportion of the variance that each of the principal components explains.

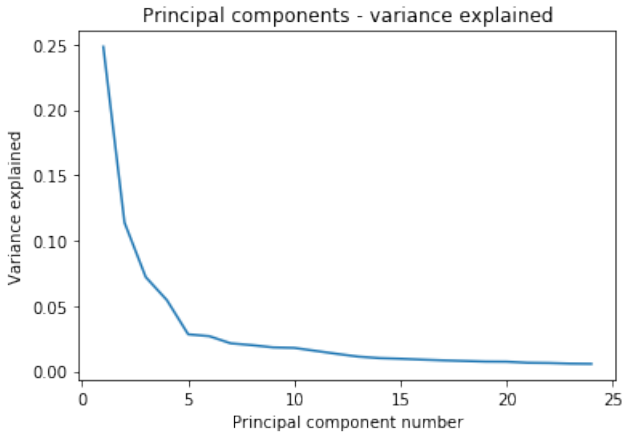

Fig. 5. A visualization of how much of the total variance each of the top 24 principal components explain. All together these principal components explained 74.4% of the total variance.

The top 4 principal components are visualized in figure 6 below and how much variance each of these explains can be seen in figure 5 above.

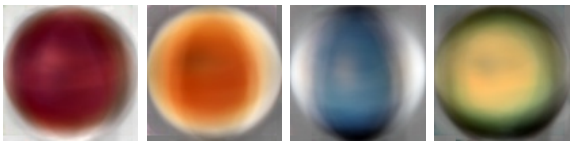

Fig. 6. A visualization of the top four (from left to right) principal components extracted from the training set.

Other, non-top, components are visualized in figure 7 below.

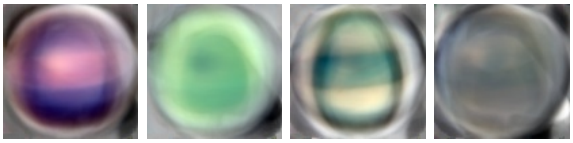

Fig. 7. A visualization of four principal components extracted from the training set. The components in this figure are the ones explaining (from left to right) the 9th, 10th, 14th and 20th most of the variance.

Even though not all the top 24 principal components are visualized in figures 6 and 7 it is worth to mention that all the principal components that were extracted from the training set are more or less symmetric. The components all have a round shape and the color of the top components differ while the last of the top 24 components look like the rightmost one in figure 7 above.

B. Custom features

The results of the classifiers when using the custom features can be seen in tables I for SVM and III for RF. The accuracy when testing on the validation data was 91.39% for the SVM classifier and 93.46% for the RF classifier.

C. 2D DFT

The results of the classifiers when using the the 2D DFT and PCA can be seen in tables I for SVM and III for RF. The accuracy when running on the validation data was 81.97% for the SVM classifier and 80.13% for the RF classifier.

D. Support Vector Machine (SVM)

The accuracies of the SVM classifier for the different preprocessing methods are stated in table I below.

TABLE I
A SUMMARY OF THE ACCURACIES OF THE SVM CLASSIFIER FOR THE DIFFERENT ALGORITHMS, CF STANDS FOR CUSTOM FEATURES.

	Training set	Validation set
PCA SVM accuracy (%)	100.0	96.73
CF SVM accuracy (%)	94.19	91.39
DFT+PCA SVM accuracy (%)	99.59	81.97

The SVM classifier gave an accuracy of 96.73% on the validation set when using PCA and a perfect accuracy on the training set. This accuracy was obtained with a linear kernel, since it was the kernel that gave the best results. The confusion matrix of this classifier can be seen in figure 8 below.

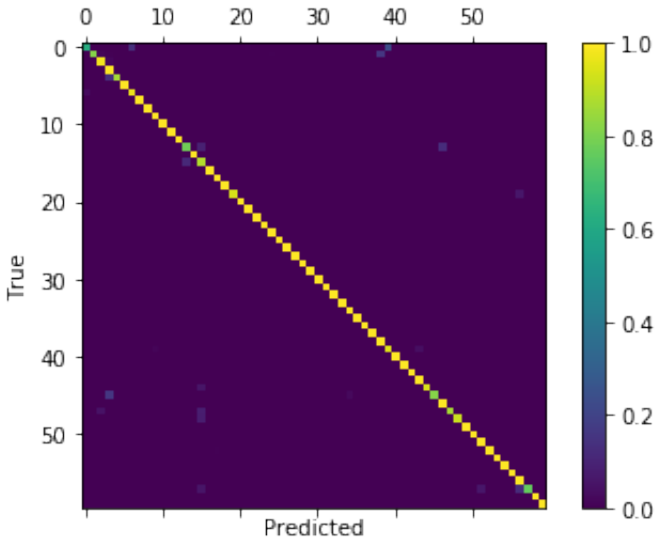

Fig. 8. The confusion matrix that is summarizing the results of the SVM classifier on the validation set.

The worst performing classes of the SVM classifier are summarized in table II below.

TABLE II
SUMMARY OF THE WORST PERFORMING CLASSES FOR THE SVM CLASSIFIER ON THE VALIDATION SET WITH THE DIFFERENT PREPROCESSING METHODS. THE ERROR FOR A CLASS IS HERE DEFINED AS THE PERCENTAGE OF WRONG CLASSIFIED INSTANCES OF THE PARTICULAR CLASS. ALL THE NUMBERS ARE SHOWN IN PERCENTAGES.

Class	PCA Error	CF Error	FFT Error	Avg Error
Apple Braeburn	37.8	34.8	28.7	33.7
Pomegranate	0.0	100.0	1.2	33.7
Peach Flat	8.5	66.5	23.8	32.9
Pear	18.3	29.9	41.5	29.9
Pear Monster	12.7	16.9	44.0	24.5
Pepino	0.0	28.9	42.8	23.9
Nectarine	6.7	19.5	40.9	22.4
Peach	0.0	47.0	12.2	19.7
Apple Red 2	3.0	32.9	22.6	19.5
Banana Red	0.0	15.7	39.8	18.5

The four instances of the classes in table II above are visualized in figure 9 below.

Fig. 9. A visualization of fruits corresponding to the eight worst performing classes of the SVM classifier with the PCA processing. From left: 'Apple Braeburn', 'Strawberry', 'Banana' and 'Apple Golden 1'.

E. Random Forest

The accuracies of the random forest classifier are stated in table III below for both the training set and the validation set.

TABLE III
A SUMMARY OF THE ACCURACIES OF THE RF CLASSIFIER FOR THE DIFFERENT ALGORITHMS, CF STANDS FOR CUSTOM FEATURES.

	Training set	Validation set
PCA RF accuracy (%)	100.0	91.73
CF RF accuracy (%)	100.0	93.46
FFT RF accuracy (%)	100.0	80.13

The RF classifier with PCA resulted in an 91.7% accuracy on the validation set, while this classifier with the custom features achieved an accuracy of 93.5%. The confusion matrix visualized in figure 10 below shows how the RF classifier (PCA) preformed on the validation set.

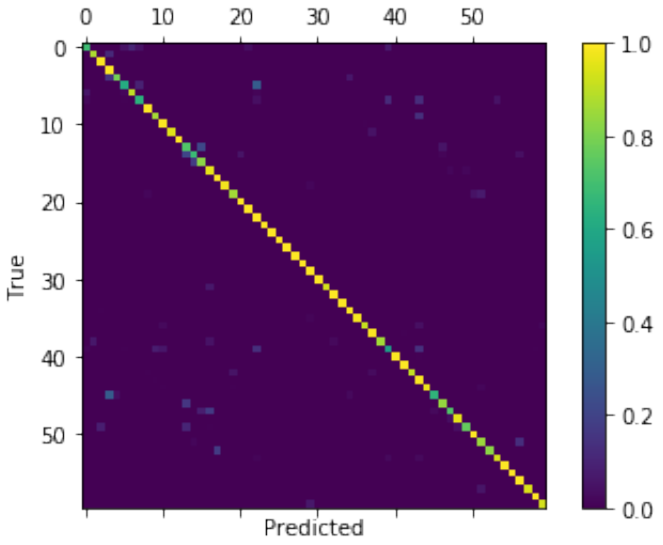

Fig. 10. The confusion matrix that is summarizing the results of the RF classifier (PCA) on the validation set.

Table IV below summarizes the worst classes of the Random Forest classifier.

TABLE IV
SUMMARY OF THE WORST PERFORMING CLASSES FOR THE RF CLASSIFIER ON THE VALIDATION SET. THE ERROR FOR A CLASS IS DEFINED AS THE PERCENTAGE OF WRONG CLASSIFIED INSTANCES OF THE PARTICULAR CLASS.

Class	PCA Error	CF Error	FFT Error	Avg Error
Apple Red 1	38.4	12.8	59.8	37.0
Nectarine	37.2	32.9	39.6	36.6
Pear	34.1	23.2	37.8	31.7
Pepino	27.7	27.1	36.7	30.5
Apple Braeburn	31.7	26.2	31.1	29.7
Pear Monster	23.5	21.7	33.7	26.3
Pear Abate	14.5	16.3	45.2	25.3
Banana Red	38.0	0.6	36.1	24.9
Apple Red 3	29.2	9.7	29.9	22.9
Pomegranate	7.3	32.9	25.6	22.0

Instances of the four classes for the random forest classifier with the PCA processing are visualized in figure 11 below.

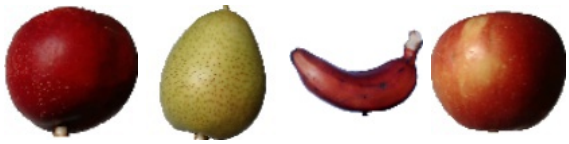

Fig. 11. A visualization of fruits corresponding to the four worst performing classes of the RF classifier with the PCA processing. From left: 'Nectarine', 'Pear', 'Banana Red' and 'Apple Braeburn'.

VI. DISCUSSION

A. Principal Component Analysis

By looking at figures 5 and 6 one can conclude that roughly a quarter of the variance is explained by a principal component that is red and round. This suggests a preponderance of these types of fruit in the dataset.

The accuracy of the support vector machine was 96.73% and better than expected. A linear kernel was used which means that the classes are close to being linearly separable. We had a big issue with the SVM classifier and throughout most of the project only a 45% accuracy was achieved. The reason for the poor accuracy was the kernel that was used. First we used the rbf kernel and changing this to a polynomial kernel increased the accuracy to over 90% immediately. By modifying the number of principal components we could finally achieve the final accuracy of 96.73% which was the best accuracy obtained by any classifier.

With the random forest classifier we could not obtain a better accuracy than 91.73% for validation set when using PCA.

B. Custom Features

The custom features appear to work quite well for both classifiers. Some intuition why this could be is that the first and second moment of the RGB channels along with a few key numbers on the shape of the fruit ought to be sufficient for distinguishing different fruits in all but the most pathological cases.

The SVM performs considerably worse in comparison with the SVM on just the PCA set of features as can be seen in table I. The reason is most likely that the custom features are not linearly separable, as the training set error implies. This is not unexpected as the custom features are fewer than the ones derived from PCA and very likely does not capture all distinguishing components from that set of features.

The Random Forest outperforms on this set in comparison to the others, as can be seen in table III. It is harder to explain why this ought to be the case but a possible reason could be that each of the individual components explain more variance (although they are correlated) and subsequently each split in the decision tree ought to have greater significance.

C. 2D DFT

This set of features performed the worst for both classifiers. The idea behind choosing it originally was that the 2D DFT preserves the original data (up to numerics) hence running it before running the PCA would yield another set of features. This set would possibly account for textures and patterns in the fruit as they exhibit periodicity, but the results are clear: it performed badly.

One of the reasons why this set performs the worst is that the background can be better dealt with by just using the PCA. When performing the FFT, the background gets added up between multiple frequencies (especially the lower ones) and these frequencies might encode e.g. mean colors, hence they still need to be used. However, when using just the PCA, the background is implicitly handled since the background is assigned to specific pixel values and thus remains in largely the same area over all pictures (for this specific dataset).

Another reason is that the 2D DFT is performed on a 45×45 pixel image, which might be too small to have the frequency components correspond to any real patterns.

Two ways of counteracting these negatives would be higher resolution images and a high-pass filter. Higher resolution images would naturally result in that patterns would be more easily distinguished and hence more prevalent in the frequency decomposition. The high-pass filter would mean that we ignore the lower frequencies corresponding to the background and the mean fruit color. The loss of the mean fruit color would however be compensated for given adequately high resolution such that texture is revealed, hence it would probably be a net-positive effect.

D. Worst Performing Classes

One can deduct from tables II and IV that the worst classified fruit are mostly red and round or green and oval. This is because a lot of classes have these properties, thus it is naturally hard to distinguish between these.

One can see in the figure 9 that bananas are poorly classified when using the SVM with PCA, but there is no other fruit with the same color and shape as the banana. The reason for the bad classifying is instead how the set of images were split into the training set and validation set. In one filming of the banana 327 images were extracted and 23 consecutive images of these happened to end up in the validation set. Events like these will cause bad classifying since there is no image in the training set looking very similar.

This event has been illustrated in figure 1 that is visualizing fruits from the training set and 3 that is visualizing fruits from the validation set. One can see that for the apple, huckleberry and papaya the images looks identical but for the banana it is a big difference.

This particular event for classes might cause them being poorly classified, and it shows a flaw in the dataset. Two consecutive pictures in one filming looks almost identical and for the most part there are images in the training set that looks identical to the ones in the dataset. This is most likely the reason why the simple classification methods that are used works so well. To mitigate this, one could augment the training set with flips in the particular case of such an asymmetric fruit as the banana to improve the probability of it being observed from all angles.

VII. Future Work

In this section it will be suggested how one could proceed if continuing working on the project for an extended period of time.

Linear Discriminant Analysis (LDA) is a method to extract features with the focus to model the difference between classes which is not modeled by the PCA [2]. To implement LDA would not be very time consuming and there is a possibility that better results can be achieved by this substitution.

To make this project more applicable to use, an interesting new path one can make to the project is to try to recognize fruit in real world pictures. That is in pictures which does not contain only a certain fruit and also not only a white background.

ACKNOWLEDGMENTS

We want to give an acknowledgment to Horea Muresan and Mihai Oltean for creating the dataset and for making it public.

We also want to acknowledge IEEE Computational Intelligence Society for providing the LaTeX- template that we used in this paper.

REFERENCES

[1] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Berlin, Heidelberg: Springer-Verlag, 2006. ISBN: 0387310738.

[2] A. M. Martinez and A. C. Kak. "PCA versus LDA". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 23.2 (Feb. 2001), pp. 228–233. ISSN: 0162-8828. DOI: 10.1109/34.908974.

[3] Horea Muresan and Mihai Oltean. "Fruit recognition from images using deep learning". In: *CoRR* abs/1712.00580 (2017). arXiv: 1712.00580. URL: http://arxiv.org/abs/1712.00580.

[4] Arivazhagan Selvaraj et al. "Fruit Recognition using Color and Texture Features". In: 1 (Oct. 2010), pp. 90–94.

[5] Woo Chaw Seng and S. H. Mirisaee. "A new method for fruits recognition system". In: *2009 International Conference on Electrical Engineering and Informatics*. Vol. 01. Aug. 2009, pp. 130–134. DOI: 10.1109/ICEEI.2009.5254804.

[6] Hossam M. Zawbaa et al. "Automatic Fruit Image Recognition System Based on Shape and Color Features". In: *Advanced Machine Learning Technologies and Applications*. Ed. by Aboul Ella Hassanien, Mohamed F. Tolba, and Ahmad Taher Azar. Cham: Springer International Publishing, 2014, pp. 278–290.