Fruits Image Classification

Davide Gianatti

6th February 2023

The aim of this report is to present a couple of different methods to classify fruits in order to automize some industrial procedure that exploit fruits classification.

1 Data set

Before trying to find suitable algorithms to deal with a problem, it is necessary to look carefully at the data set at our disposal. In particular the data set is composed by 6400 images of 16 different kinds of fruits, which they have been divided in a training, validation and test sets with the following proportions: 75/12.5/12.5 . The values of the images has been rescaled from $[0, 255]$ to $[0, 1]$.

1.1 Data augmentation

Considering that each fruit is represented by 300 samples, we can conclude that the data set is quite big but not huge, so an augmentation procedure for enlarging it could be implemented.
The filters tried are the following:
Noise: extracted from an uniform distribution $U(-0.1, 0.1)$; *Smoothing*: each pixels get its value from the mean of the first neighbours; *Cooling*: adding 0.1 to the blue component; *Warming*: adding 0.1 to the red component; *Mirroring*: reflection with respect to a diagonal of the image;
This way the training data set can be enlarged to include 28.800 images and hopefully our trained algorithms will be more robust. This data set will be used only in the very last part of the report.

1.2 Background

Images have been taken with a white background in order to be easier to analyze. It is clear that white constant pixels do not carry any information at all, but it seemed more natural to impose that the background should be black, such that the absence of information is carried by pixels with null values. Moreover this choice could be justified for a few reason; firstly, considering a CNN scheme, padding could be introduced (where we add black pixels on the edges of the images) and a sort of continuity is guaranteed for the first layer with this solution, and finally it is more practical to maintain a black conveyor belt of the same color despite a white one, just think of some tomatoes crushing on a stainless snow-white tape. The background was slightly noisy, so to blacken it a threshold was set (the mean of the red, green and blue pixels should have been greater than 0.95 in order to be set equal to 0).

2 Classification models

In this section the structure, the performances and the motivations behind the models implemented are presented.

2.1 First implementations

An algorithm based on transfer learning has been tried as the first attempt. The last layer of the already trained ”pvmlnet” was poped out, so that for each image it has been possible to extract 1024 features which were the activation functions of the new last neurons of output. A multinomial logistic regression with 16 outputs was performed and in few epochs the classifier reached 100% accuracy on the train, validation and finally the test data.
Then I tried to build a convolutional neural network from scratch using the library TensorFlow and the architecture considered was the following:
- Convolutional layer with a 3 × 32 multichannel filter (3,3) with (3,3) strides and ReLU;
- Max pooling (2,2);
- Convolutional layer with a 32 × 64 multichannel filter (3,3) with (2,2) strides and ReLU;
- Max pooling (2,2);
- Flatten to 1024 neurons;
- Dense layer with 16 neurons of output with a softmax;

Again in few epochs and with a not so shrewd choice of the hyperparameters 100% accuracy was obtained on all the sets. It couldn't be it!
With just two exploratory trials perfect performances were obtained, this is the reason why, from that point on, I tried to build algorithms as small (in terms of parameters used) and as fast as possible, in order to make them suitable for some industrial applications such as the spectacular one that can be seen through this YouTube link ”https://youtube.com/shorts/1ZiSrslVgRI?feature=share”.
For istance the extraction of the transfer learning features requires around 0.1 s, way too much in different applications.

2.2 Low-Level features

When thinking of models with fewer parameters, the first thing that came in to my mind was a class of algorithms based on low-level features chosen ad hoc by the implementer.
I tried to use the following features implemented during the laboratory activities:
- *Color histogram* with 64 bins;
- *Edge direction histogram* with 64 bins;
- *Grey cooccurrence matrix* with 8 bins;

Moreover I considered two additional histograms with 64 bins, one collecting the count of the *luminosity* values and the other one the *saturation*, where these two quantities are defined for a single pixel as:

$$\text{Luminosity} = \frac{\text{Max(RGB)} + \text{Min(RGB)}}{2} \qquad \text{Saturation} = \frac{\text{Max(RGB) - Min(RGB)}}{1 - |\text{Max(RGB)} + \text{Min(RGB)} - 1|}$$

In the following figure it is possible to see all the low-level feature of an apple and a banana concatenated in a single vector with dimensions equal to 448.

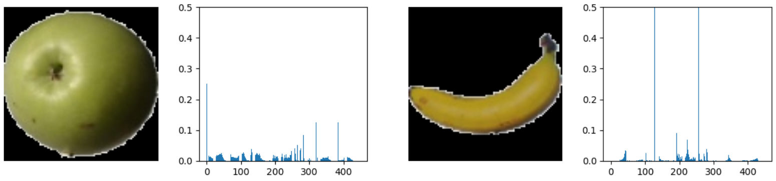

Figure 1: Low-level fatures of an apple and a banana.

In order to see if the classes could be approximately linearly separable I considered the first principal components of the features using the pvml library. When retaining just two of them we are describing 64%

of the actual variance, while with three components up to 74% . Four classes of the training set are plotted below using the feature extracted by the PCA.

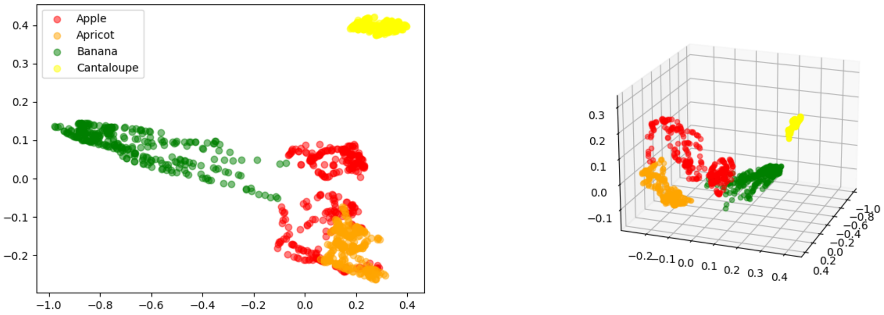

Figure 2: The first 2 and 3 principal components of some of the training samples.

The classes are grouped in small clouds that tend to become linearly separable as the number of PCs grows. Because of the observations made, I decided to use a linear SVM with soft margin for the classification problem, adopting the one-vs-all scheme since the classes were too many for the implementation of the one-vs-one strategy. The training was performed using the library sklearn with a built in stopping criteria. The accuracy on the validation set has been used as a measure of goodness for the coefficient of the L_2 regularisation term and the values tried were 0.03, 0.1, 0.3 and 1; the best value among these was $\lambda = 0.03$, with which the accuracies on both the validation and training set were equal to 100%.
The good performance of the model is confirmed even by the accuracy on the test set, again equal to 100%. However this method isn't actually that fast, the extraction of the features from an image requires approximately 0.1 s. The only poor advantage obtained over the transfer learning algorithm is that we have to store much less parameters $(448 \times 16 + 16 = 7.184)$.

2.3 Small CNN

Different architecture of CNN with a decreasing number of parameters were tried and in this section I will present the one that still achieved good performances while using very few parameters.

2.3.1 Architecture

The architecture of the network is the following:
- Convolutional layer with a 3×3 multichannel filter (3,3) with (5,5) strides and ReLU;
- Max pooling (2,2);
- Convolutional layer with a 3×6 multichannel filter (2,2) with (2,2) strides and ReLU;
- Average pooling (5,5);
- Flatten to 6 neurons;
- Dense layer with 16 neurons of output with a softmax;
The parameters that compose this network are 274, even fewer than the ones used in the low-level approach. The training of the CNN was performed exploiting the built in Adam optimization algorithm of the TensorFlow library.

2.3.2 Hyperparameters

Exploratory trials have been performed in order to decide the number of epochs and the size of the batches. The choice made for these hyperparameters was to go through all the data set 250 times considering batches of 3 samples; this way the training was quite fast while obtaining a good accuracy on the validation set. With the previous hyperparameters fixed is possibile to choose the optimal pair of the learning ratio η and the coefficient λ of the regularisation term L_2 through a search grid reporting the accuracy on the validation set. The best pairs seem to be $(\lambda, \eta) = (0, 10^{-2})$ and $(10^{-3}, 10^{-2})$, with both of which the accuracy achieved is equal to an impressive 0.998. Between the two I decided to choose the one with the regularisation coefficient different from 0 hoping that the classifier would be able to generalize better to new data. Finally the network kept was the one trained using 250 epochs, with batches of 3 samples, $\eta = 10^{-2}$ and $\lambda = 10^{-3}$. The accuracy obtained on the test set was 99.8%.

2.3.3 SVM classifier

Another classifier that could be implemented onto this network replace the last dense layer with a SVM, hoping that the CNN, in the last 6 neurons, learned to separate as much as possible the classes. We can try to see if this is actually the case by performing a PCA for the visualisation of the distribution of the fruits samples inside this new space of parameters. Considering respectively two and three principal component, 98% and 99% of the variance is described.

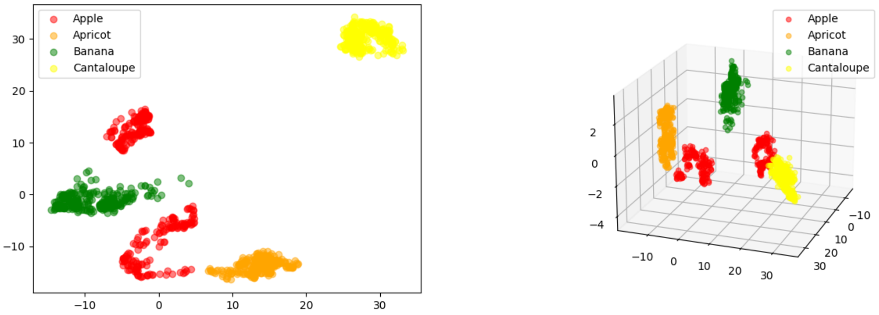

Figure 3: The first 2 and 3 principal components of some of the training samples.

The classes seem to be more compact than what we have seen as far as concerned the low-level features. Also it is interesting to notice that the apple classes is divided in two well separated clouds, probably this is due to the fact that it is possible to quite easily tell apart the red apples from the green ones.
Using a SVM with soft margin felt like a good choice. Therefore, considering the activation functions of the the last hidden layer as features, a linear support vector machine was trained with different values of the L_2 regularisation coefficient (the stopping criteria of the sklearn library was already built in and, since it was working well, I decided not to try different strategies). Between $\lambda = 0.01$, 0.03, 0.1, 0.3 and 1, the best performances on both the validation and the training set were obtained thanks to $\lambda = 1$.

2.3.4 SVM performances

With the choice of the hyperparameters seen above, the accuracy achieved on the test set was equal to 99.9%, greater than the one reached by the plain CNN. Just one single sample was missclassified and it was an apple mistaken for a peach (even in the validation set the errors came just from those two classes).
Another advantage of this classifier over the plain CNN is that we have to compute only linear functions and not the more computationally expensive softmax on the output layer.
The last model presented was selected as the best one among the small and fast algorithms that had been implemented. We shall see if it is fast enough to justify its usage over the more complex network described in (2.1) containing 35.792 parameters. The small classifier needs around 0.003 s to assign a class to a single sample while the bigger CNN requires 0.004 s. The gain in the time wasted is just one millisecond and this advantage is significant only if we need to classify more than 250 fruits per second with a single processor like the one of my computer.

2.3.5 Data augmentation

The SVM classifier is not infallible and in order to make it more solid I tried to extend the data set through an augmentation procedure. The filters applied were the ones described in (1.1) and the new training set included 28.800 images, unfortunately they were way too much for my domestic computer, so I decided to consider only the augmented data of the problematic apples and peaches. The search of the hypermeters (all fixed equal to the ones already obtained, except the weight decay coefficient of the CNN) has been conducted keeping an eye on the accuracy on the validation set. Overfitting occurred using very low value of the regularisation coefficient, as the algorithm tried to learn the "noise" introduced. The best choice of the regularisation coefficient was $\lambda = 0.003$, with which the accuracy on the validation set was 99% but with no apples mistaken for peaches. So we have worsened the accuracy but we have more sound predictions regarding apples and peaches. This is actually a promising result as leads to think that a perfect accuracy could be achieved if all the augmented data set is used.

2.3.6 Green and red apples

The best result for the SVM classifier on the non-augmented data had been achieved by recognizing that green and red apples have to be seen as two different classes (in order to convince yourself is enough to look at the images of the PCA). In fact considering 17 classes, through the splitting of the apple class, the accuracy obtained was 100% on the train, validation and test set.

2.3.7 Alternative models

Using the activations of the last 6 neurons as features is possible to implement many classifiers. In order to make the algorithm even faster I tried the minimum distance classifier that led to an accuracy on the test set of 85%. The last remarkable result was obtained considering a gaussian RBF kernel SVM with $\gamma = 1$, that was able to classify correctly all the samples in the test set. The only drawback is the computational burden of computing many gaussians that brought back the time wasted to be around 0.004 s.

3 Conclusions

Different strategies with different results were implemented, and the choice of which one to exploit depends on the practical application they are intended to work with. If a fast performance is required than the small CNN could be the best solution, while if the classification procedure is not requested to happen in tiny fractions of a second, a more solid and complex model should be preferred.