



POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Blob analysis for dimensional features and shape defect extraction in agri-food production

TESI DI LAUREA MAGISTRALE IN
FOOD ENGINEERING-INGEGNERIA ALIMENTARE

Author: **Davide Mondin**

Student ID: 964907

Advisor: Prof. Marco Tarabini

Co-advisor: Ing. Chiara Conese

Academic Year: 2022-2023

Abstract

This document deals with quality inspection of aubergine slices through the analysis of RGB images: localisation, classification, and features extraction. A deep learning neural network named “Yolov8” has been coupled with Radom Forest in order to localise/classify the targets and extracting the BLOBs; the results for localization are excellent while the poor classification performances confirm the difficulties experienced by image oriented neural networks when played to distinguish similar objects and feed with insufficient material. Feature analysis revealed high potential for classification. The proposed workflow is prone to be used in early stages of those processes that intend to implement an online non-contact sorting technique based on deep learning imaging.

Key-words: Object detection, Machine learning, Food sorting, Deep learning, BLOB analysis.

Abstract in italiano

Questo documento tratta l'ispezione qualitativa delle fette di melanzana attraverso l'analisi di immagini RGB: localizzazione, classificazione ed estrazione di caratteristiche. Una rete neurale conosciuta come "Yolov8" è stata utilizzata insieme a "Random Forest" con l'obiettivo di localizzare/classificare i bersagli ed estrarre i BLOBs, i risultati dell'attività di localizzazione sono eccellenti, tuttavia, le scarse prestazioni in fase di classificazione confermano la difficoltà affrontate da queste reti neurali quando chiamate a distinguere classi simili, lo stesso succede quando il materiale a loro disposizione è insufficientemente elevato. L'analisi delle caratteristiche ha dimostrato un buon potenziale per l'attività di classificazione. Il workflow proposto può essere utilizzato nelle fasi iniziali di quei processi che intendono implementare lo smistamento di oggetti basato sull'apprendimento profondo di immagini.

Parole chiave: Rilevamento di oggetti, Apprendimento automatico, Smistamento di alimenti, Apprendimento profondo, analisi di BLOBs.

Contents

Abstract	i
Abstract in italiano	iii
Contents	v
1 Introduction	1
1.1. Scheme of the thesis	5
1.2. State of the art	5
1.2.1. YOLO.....	5
1.2.2. SGD.....	10
1.2.3. Decision Tree	11
1.2.4. Random Forest.....	13
1.2.5. Softmax Regression Classifier.....	13
1.2.6. K-Means	14
1.2.7. DBscan.....	15
1.2.8. Object Detection.....	16
1.2.9. BLOB analysis	20
2 Methods	23
2.1. Workflow.....	23
2.2. Working dataset	24
2.3. Yolov8 dataset preparation.....	26
2.4. Yolov8x	29
2.5. Pixel sorting.....	33
2.5.1. SDG application.....	35
2.5.2. Softmax Regression Classifier application.....	37
2.5.3. Decision Tree application.....	37
2.5.4. Random Forest application	38
2.5.5. DBscan application.....	39
2.5.6. K-Means application	40
2.6. Features extraction	40
2.7. Feature description & Feature comparison	45
3 Results	47

3.1.	Yolov8x results.....	47
3.2.	Machine learning results	48
3.2.1.	Supervised ML algorithms results	48
3.2.2.	Unsupervised/Semisupervised ML algorithms results.....	52
3.3.	Features extraction results	53
4	Discussion and Conclusion.....	59
	Bibliography	63
	List of Figures	67
	List of Tables	69
	Acknowledgments.....	73

1 Introduction

Food sorting plays a crucial role in food manufacturing facilities in order to filter out alien objects or/and products characterized by poor quality (unsellable). In this context, machine vision technologies have revealed high potential in recent years, Jun-Hu Cheng managed to predict chemical composition of fish muscle with Hyperspectral imaging [1], a paper from Daniel Caballero demonstrates the possibility to sort ham pieces on their tasting properties in a non-destructive process based on Magnetic resonance imaging [2], in 2015, Weishan Zhang trained a deep learning network on RGB images depicting various fruits with the purpose to locate and classify the objects [3]. Indeed, the typical food sorting procedure takes place along a conveyor belt, with a few human operators that are in charge of selecting and removing non-compliant items following instructions provided by someone else. This human activity inherently has effect on the production, mainly due to the following: human beings are unable to maintain the same standards over a long period of time, sorting operators are expensive, the task is alienating, food products selection criteria are different among the operator staff and sometimes conveyor belts' speed is too high to be followed by human hands.

This Master thesis deal with BLOB analysis, a technique used to classify extracted objects leveraging on their properties (like colour, shape and size). The thesis' aim is to locate aubergine slices from RGB images obtained from a conveyor belt going up to 3 m/s, distinguish between slices that match different predefined classes and describe them through BLOB analysis; detection was performed in various ways, including deep learning. Deep learning technology relies on neural networks (so called because it tries to simulate neural connections in natural brains), a model made by a huge

number of parameters, it allows to achieve difficult tasks and sometimes can “explain” how solve problems in those fields in which humans perform good but are unable to provide the process that leads to the solution/s [4]. The main challenges encountered throughout the whole process were:

- Overlapped objects isolation.
- Distinguish pixels belonging to aubergine skin when compared to background due to the artificial light reflects.
- Dataset's underrepresentation of certain classes.

This Master thesis is part of a project named TESORO (Studio e sviluppo di TECnologie avanzate per il SORTing automaticO nei processi di produzione industriali), “Politecnico di Milano” university is partner of such project as well as startups and big companies operating in automation, machine vision and food manufacturing. The TESORO project is carried out in a series of sequential steps, such steps are summarized as follow:

- Selecting the most effective wavelength bands able to sort the products.
- Realize a solution to detect the objects to be removed.
- Realize a solution able to remove items from a high-speed conveyor belt.
- Apply a prototype to an operative food manufacturer facility.

TESORO's novelty involves the usage of a hyperspectral camera, able to acquire hundreds of wavelengths in parallel for selecting the most effective. Although several wavelengths are acquired, only few channels can be processed on a real time application due to the high computation load to be processed. Data coming from HSI (Hyperspectral imaging) undergo to data reduction that select few channels, the filtered channels are used to localise and classify the products moving along the conveyor belt; this thesis investigates an algorithm that may be used in this phase. The algorithm involves a deep leaning application named Yolov8 and machine learning

techniques; the working database is made of RGB images of aubergine slices, such images were acquired under two different conditions, one of which portrays highly overlapped aubergine slices, a condition that makes harder to separate the product instances; this dataset is affected by a further couple of issues that are common to be found in real application cases: the presence of disturbing reflects and the colour similarities between the objects and the background.

1.1. Scheme of the thesis

Three main chapters in addition to an introduction compose this thesis. A simple review of the most recent technologies employed in this thesis is provided in the Introduction chapter, along with methods used to overcome challenges similar to those encountered during its development. Yolo's versions are described in this chapter, with a spotlight on the first and last because Yolo's original kicks off the successful idea behind this network while Yolov8 is the one used in this thesis; an overview on the most common techniques applied in object detection field is provided too. Methods chapter shows the workflow in detail as well as the tools and procedures adopted to address the thesis, the target of the work, the type of data that has been collected and used in this thesis work are explained in this part. The second main chapter is "Results", it presents the outcome of those activities described by "Methods" chapter, visual results as well as metrics and performances are provided. The last part is the one dedicated to "Discussion and Conclusion", here the results are summarized and analysed, in particular, the repeatability of the presented workflow to real case scenarios, possible limitations arising when trying to apply the same process, whether or not the metrics are satisfying, possible improvement that may be done to achieve better results and what are possible future researches suggested by this thesis 'outcomes.

1.2. State of the art

1.2.1. YOLO

Yolo (you only look once) is a deep learning technology used in the field of machine vision for object recognition and classification. Since its first appearance [5], Yolo and followings have been state-of-the art online applications due to their rapid inference times and good precision.

Yolo relies on an end-to-end “neural network” responsible for both localise and classify one or more targets, the great intuition behind Yolo is the usage of a unified “neural network” to predict class and bounding box coordinates whereas state-of-art solutions (before 2015) were based on split networks, like R-CNN [6], Yolo is a simplified solution that decreases inference speed, a highly critical performance for online applications.

Yolo received multiple updates from its release, constituted by minor advancements, Yolo's performance has substantially improved as a result of those changes. Yolo's development was carried out by a research group led by Joseph Redmon until the arrival of Yolov3 (Yolo version 3), when he gave up due to ethical concerns surrounding Yolo applications [7]. The desire to improve Yolo's performance and adapt it to real case applications drove the continued development of the software and its interface, today's versions are Yolov7 and Yolov8, they take few minutes from the official repository to the train setup and even less than ten lines of code from the user side.

The original Yolo works on a simple network composed almost exclusively of sequence of convolutional layers, activation function, pooling and a flat layer like shown in Figure 1.1.

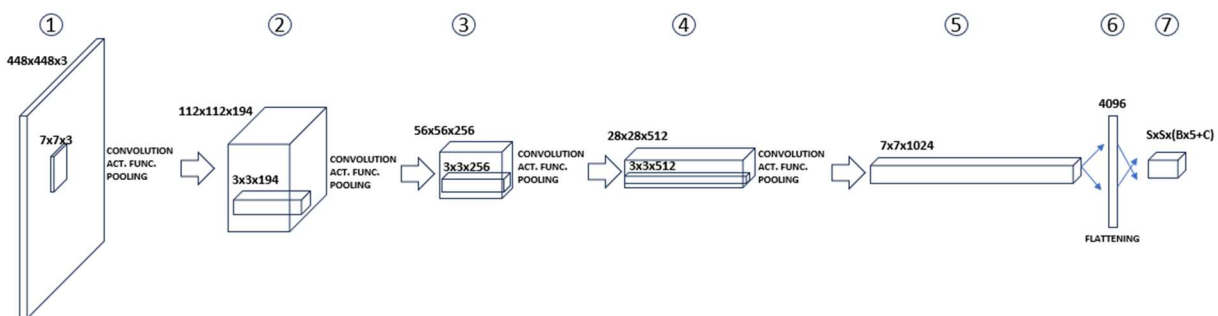


Figure 1.1:Original Yolo network.

The aim of Yolo's training is to optimise hundreds of thousands of parameters across the network, all of them play a role in “convolution” layers.

The first layer is represented by a 3D information matrix, two of these dimensions are the image's height and width (or their proxies in case a resize is necessary), and a third dimension is the number of channels, which is 3 when dealing with RGB pictures, the values range from 0 to 255 (only integer values) representing the grade of green, blue and red colours for each pixel.

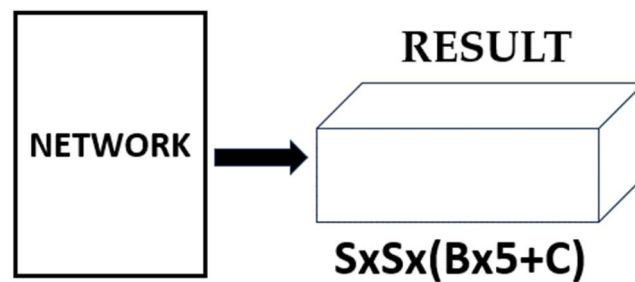


Figure 1.2: Yolo's output structure.

The final data object is named "Result", it is represented as a $S \times S \times (B \times 5 + C)$ parallelepiped, it is shown in Figure 1.2. The picture is divided into a $S \times S$ grid, each cell is responsible for predicting the bounding boxes of objects whose centres fall inside that cell; each cell predicts B bounding boxes too (decided by the user). A box has 5 data: x , y , w , h and *confidence score*, x and y locate box's centre inside the cell, w and h are the relative width and height of the box with respect to the whole image size, the *confidence score* is defined by Jospheh Redmon as follow: "These confidence scores reflect how confident the model is that the box contains an object and also how accurate it thinks the box is that it predicts. Formally we define confidence as $\text{Pr}(\text{Object}) * \text{IOU}_{\text{truthpredict}}$." [5], IoU means "intersection over union", it is the ratio between the intersection of two rectangles and their union, larger the value larger the similarity of those boxes, a graphical explanation is shown in Figure 1.3.

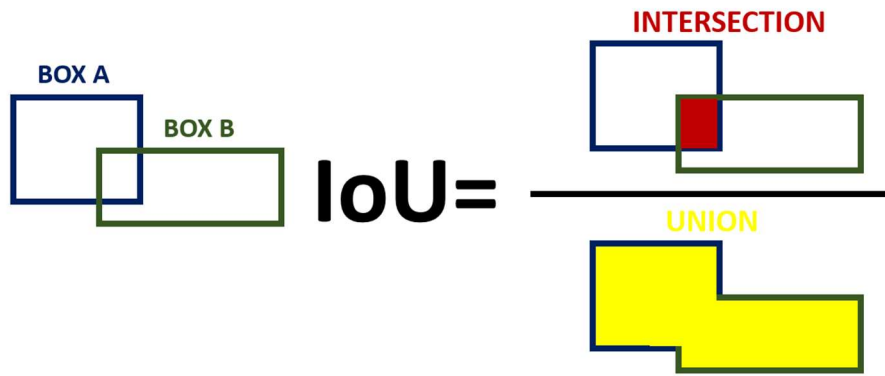


Figure 1.3: Intersection over union (IoU).

Each cell has multiple conditional probability values (C), one for each class in the model ($\text{Pr}(\text{CLASS}_i | \text{Object})$). The *confidence scores* are multiplied by C to measure both classification and localisation during the inference, a cutoff is used to filter out those boxes having an extremely low score, and a non-max suppression is applied to eliminate duplicate boxes assigned to the same object [5]. Figure 1.4 summarizes the inference process assuming B equal to 2, S equal to 5 and two classes.

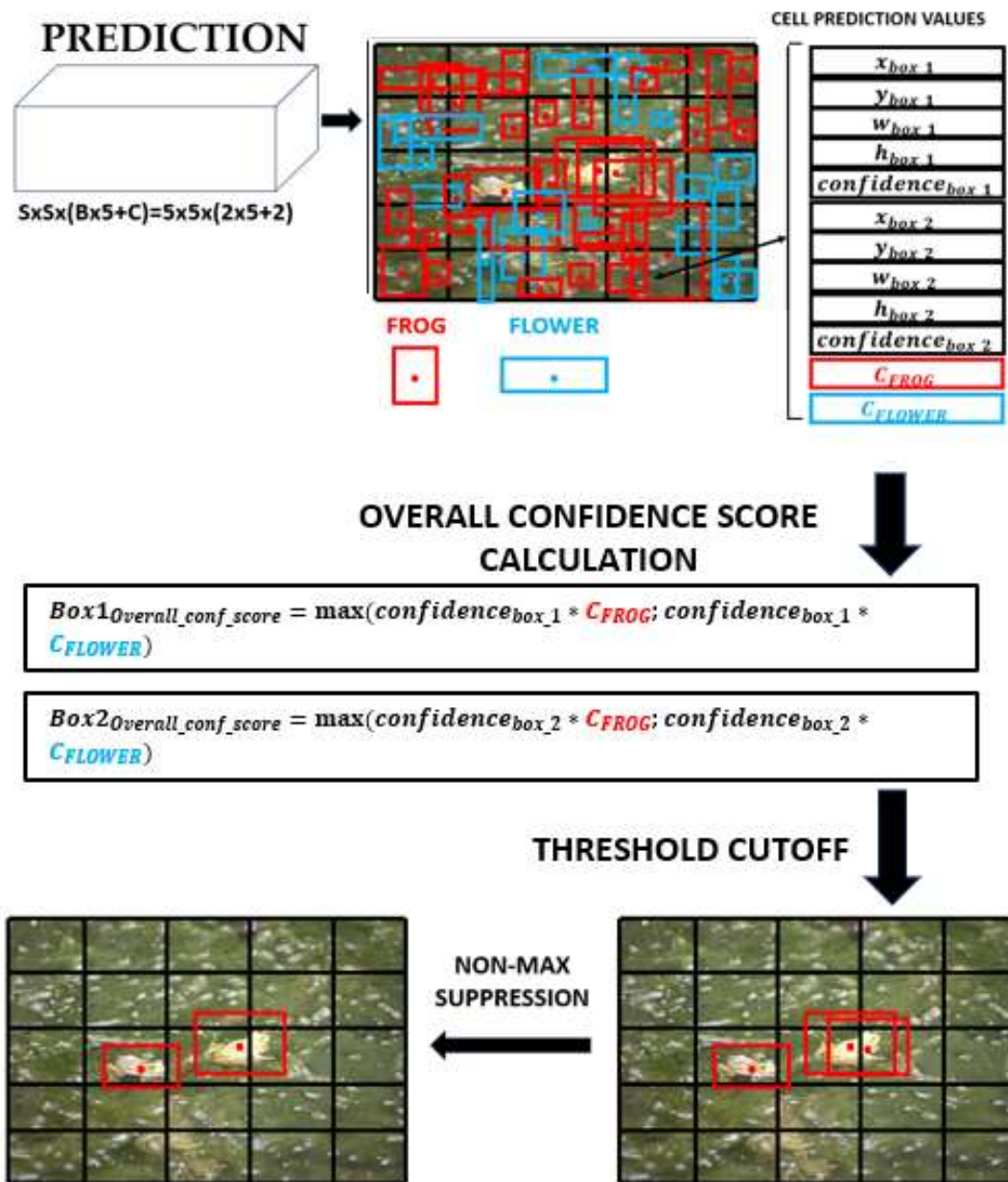


Figure 1.4: Yolo result postprocessing and output representation.

When trained, YOLO relies on backward propagation in order to adjust the weights (and biases). Once an image used in training has reached the network’s end, is compared to the ground truth, a loss function is calculated, and the parameters are adjusted. The learning speed it governed by an hyperparameter called “learning rate”, it should be controlled since a small value may cause huge training time while large values could lead to overshooting or divergency.

1.2.1.1. Yolov8

Yolov8 is one of the most recent versions, it has been released by *Ultralytics* in latter half of 2022.

The network's structure and its performances depend on the subversion (Yolov8x/l/m/s/n) (Figure 1.5), they realize a trade-off between precision and inference speed, Yolov8x is the most precise while Yolov8n is the fastest. The precision is obtained fixing an IoU threshold that defines whether or not the ground truth object has been correctly identified, the usual procedure imposes to measure the average precision (across the classes) value sliding the IoU threshold between 0.5 and 0.95, then the mean of those values defines the mAP (mean average precision), this value is widely used in literature to compare object detection techniques.

Yolov8x has been used during the development of this thesis, even though Yolov8's technical paper doesn't exist (as for many others Yolo releases), *Ultralytics* provides a detailed official documentation and the official repository, that are far enough to satisfy a wide range of needs in the machine vision field.

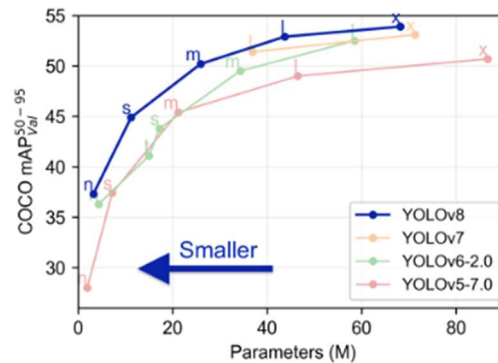


Figure 1.5: Yolo releases comparison, average precision vs number of parameters(<https://github.com/ultralytics/ultralytics>).

1.2.2. SGD

SGD (Stochastic Gradient Descent) is a machine learning technique applied to optimize an error function; it relies on sensing the function's gradient from time to time in order to modify the set of parameters associated to a model, like Equation (1.1) describes.

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta \begin{bmatrix} \frac{\partial \text{Error function}(\mathbf{X}_t, \boldsymbol{\theta}_t)}{\partial \theta_1} \\ \frac{\partial \text{Error function}(\mathbf{X}_t, \boldsymbol{\theta}_t)}{\partial \theta_2} \\ \dots \end{bmatrix} \quad (1.1)$$

$\boldsymbol{\theta}_t$ is a vector that represents the model's parameters at iteration t of the training, η is the learning rate, a value that imposes the pace of gradient descending, \mathbf{X}_t vector includes model's features at iteration t , the function's gradient is subtracted to the parameters vector since the aim is to minimize the error function.

The learning rate plays an important role: a large value might lead to divergency (overshooting) while a small value consumes more time and possibly prevent training algorithm to find the global minimum. A possible solution to figure out this issue is to change learning rate while training the model.

SDG is more efficient when compared to BGD (Batch Gradient Descent) since it doesn't need to iterate through the entire training set but only one instance at time is processed. However, the gradient is less "stable" in contrast to BGD.

When SGD is applied to linear regression, MSE (mean squared error) is chosen as error function. MSE hasn't local minimum but only a global minimum, even though BGD or Equation (1.2) are more precise, SGD is favoured in those cases where hundreds of thousands of parameters are present since it's faster [8].

$$\hat{\boldsymbol{\theta}} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y} \quad (1.2)$$

\mathbf{y} vector retains the ground truth values in the training set.

1.2.3. Decision Tree

Decision Tree is a machine learning application developed for classification tasks, Decision Tree is a graph made of three types of elements: "root", "flower" and "leaf". The "root" is the graph's starting node, it and the "flowers" are nodes that represent threshold choices based on features, these choices split the dataset in two parts with

the precise aim to build up instance groups being pure in term of class. During prediction, the Decision Tree is climbed up to a termination node named “leaf” that tells which class is to be chosen. A Decision Tree’s structure looks like the one in Figure 1.6.

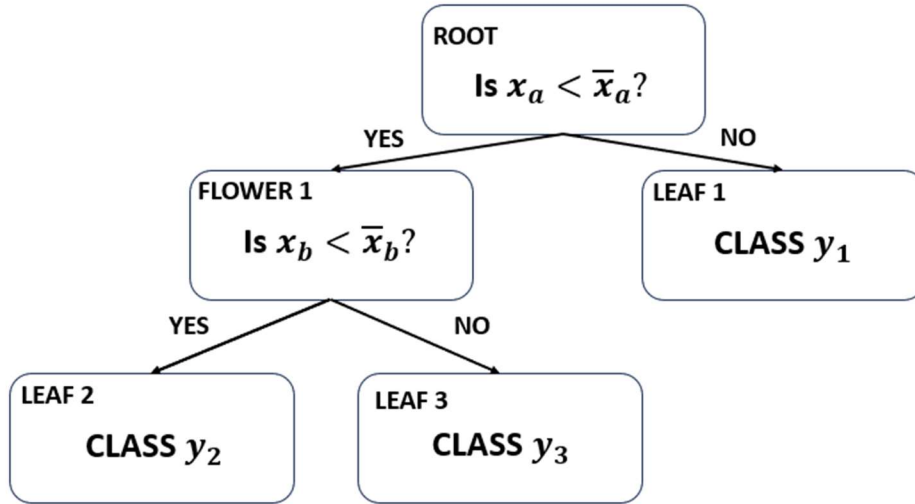


Figure 1.6: Decision Tree structure.

The main training algorithm is CART (Classification And Regression Tree), it works in a simple way: starting from the root, the algorithm iterates through all tuples of classes and values (i, ϵ) that minimize Equation (1.3)-Equation (1.4)-Equation (1.5).

$$\text{Cost funcion} = \frac{n_{YES}}{n} G_{YES} + \frac{n_{NO}}{n} G_{NO} \quad (1.3)$$

$$\begin{cases} y \in YES & \text{if } x_i < \epsilon \\ y \in NO & \text{else} \end{cases} \quad (1.4)$$

$$G_k = 1 - \sum_{j=1}^J p_{k,j}^2 \quad (1.5)$$

n is the number of training instances undergoing to division on the node, n_{YES} and n_{NO} are the number of instances on the “yes” branch and “no” branch respectively if that specific (i, ϵ) tuple is chosen, G is the Gini impurity index as described by Equation (1.5), J is the number of possible classes, k identifies a node, $p_{k,j}$ defines the ratio

between the instances belonging to j^{th} class in the k^{th} node and all the instance in the k^{th} node.

The procedure continues until one or more of the following stop criteria are satisfied: the node's Gini index reaches zero, the tree reaches its deepest point, or the node has fewer residual instances than a threshold. The Entropy index (Equation (1.6)) could be used instead of Gini impurity index (Equation (1.5)).

$$H_k = - \sum_{j=1}^J p_{k,j} \log_2(p_{k,j}) \quad p_{k,j} \neq 0 \quad (1.6)$$

1.2.4. Random Forest

Random Forest is an ensemble of Decision Trees, trees' variance is pushed forcing CART to ignore a subgroup of features then trying to split the dataset; prediction is made passing an instance to the "forest" and selecting the class having the higher relative frequency among Decision Trees' outcomes. Random forest's variance could be pushed forward letting CART evaluate only one threshold for each feature/node couple.

Random forest includes the possibility to rank the features by their "importance", such "importance" values could help to remove useless features, it is particularly useful then dealing with a high number of features [8].

1.2.5. Softmax Regression Classifier

Softmax Regression Classifier (SRC) is a machine learning technique widely applied for classification tasks. SRC's prediction takes an instance and computes many prediction scores as there are classes, each one is a linear combination of the instance's features (without bias). Prediction scores are calculated as imposed by Equation (1.7) [8].

$$score(x) = x^T \theta \quad (1.7)$$

θ matrix represents the model's parameters, so the matrix's size is equal to the number of features by the number of classes.

The score vector ($score(\mathbf{x})$) feed the Softmax function, doing so, each class has a probability value associated to the instance (Equation (1.8)).

$$p(\mathbf{x}, k) = \frac{e^{score(\mathbf{x})_k}}{\sum_{j=0}^J e^{score(\mathbf{x})_j}} \quad (1.8)$$

$p(\mathbf{x}, k)$ is the probability that instance \mathbf{x} belongs to class k (estimated), J is the number of classes in the model.

The higher score (or probability) defines the chosen class for that instance.

Training relies on tools like SGD, in that case, the loss function to be optimized is the Cross entropy cost function (Equation (1.9)).

$$H = -\frac{1}{m} \sum_{i=1}^m \sum_{j=1}^J y_j(i) \log(p(\mathbf{x}(i), j)) \quad (1.9)$$

$y_j(i)$ is the ground truth value, it is equal to 1 if the i^{th} instance belongs to the j^{th} class (otherwise is zero), m is the number of instances processed during the training.

1.2.6. K-Means

K-Means is an unsupervised machine learning algorithm; this unsupervised tool doesn't depend on a ground truth set like happens in supervised classification tasks (Softmax, Decision Tree...). K-Means is in charge to partitioning the training instances leveraging on their known features, each "group" is also named "cluster". The prediction is made by comparing the new instance's features to the training once and selecting the cluster sharing more similarities with it, the new instance is then assigned to that cluster [8].

The training's output is a set of clusters' centroids, such centroids are used to define clusters' boundaries (hard clustering) or to calculate the Euclidean distance between the new instance and the centroids in the features space (soft clustering).

The original training algorithm works in few simple steps:

- Select K (provided by the user) starting centroids.
- Label the training instances with respect to the distance between them and the centroids.
- Update centroids' locations based on the labels, if the centroids moved after this, run the previous step.

It is possible for proof that the algorithm reaches an end point after a finite number of steps. The training is usually performed on minibatches in order to speed up the process, the cost to play for that is a lower training inertia.

Selecting the number of clusters is a key activity when dealing with clustering; a possible way to address the problem points toward the "silhouette score", in practice, the mean silhouette score is calculated once the training is over (Equation (1.10)).

$$\text{silhouette score}(i, K) = \frac{b - a}{\max(a, b)} \quad (1.10)$$

a is the mean distance of the i^{th} instance with the other instances in the same cluster, b is the distance to the closest centroid (excluding the one to which the instance belongs), K is the number of clusters. The K to be chosen is the one having the higher mean "silhouette score".

1.2.7. DBscan

DBscan defines cluster regions, it works under the assumption that close instances (in the features space) belongs to the same group. DBscan hasn't its own prediction routine but it is used as an input by KNN (K-nearest neighbours) and other tools [8].

The training process is summarized in three steps:

- The instances that have at least n_{min} instances that are less than ε (distance in the features space) away from the instance itself are the core instances (n_{min} and ε are passed parameters).
- All instances being closer than ε from a core instance belongs to the same cluster (it applies to the core instances too).
- The residuals are assumed to be anomalies.

Figure 1.7 shows a possible successful case, where DBscan is applied because the training dataset obeys to DBscan's main assumption (dense and continuous regions).

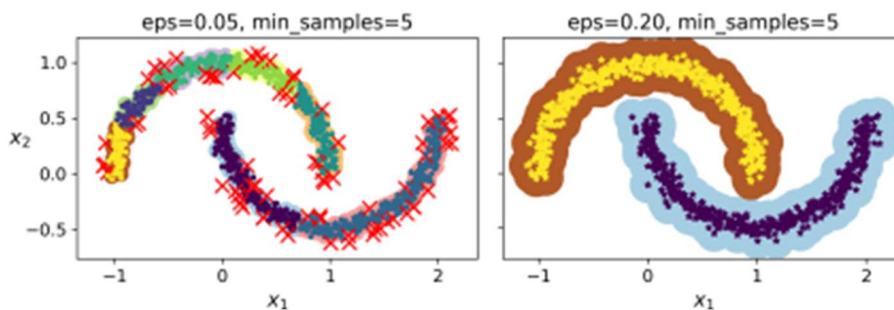


Figure 1.7: DBscan working principle [8].

A new instance is labelled by KNN predictor based on DBscan in accordance with the higher relative frequency in the K (user-selected parameter) closest instances (labelled using DBscan criterion).

1.2.8. Object Detection

Object detection is part of machine vision techniques, its goal is to locate something inside a picture or a video, it is strictly related to classification since object detection should be able to separate image's area it is interested into with respect to others. A detection has various output types depending on the used techniques, a common detection output uses bounding boxes to tells where the object/s is/are located, Yolo is one of them; someone may be interested in encircling the interest object or locating it through a single point. Representing object's place using a single point is wide used in those applications where the user is interested in moving the target. [9] [10] [11].

Deep learning technologies are becoming widely applied for object detection tasks, an increasing interest from researchers for this immature technology makes deep learning for machine vision a highly dynamic topic where the State-of-the-Art changes every few months (Figure 1.8); Yolov8 is only the last of several tools developed in the attempt to push networks' performances.

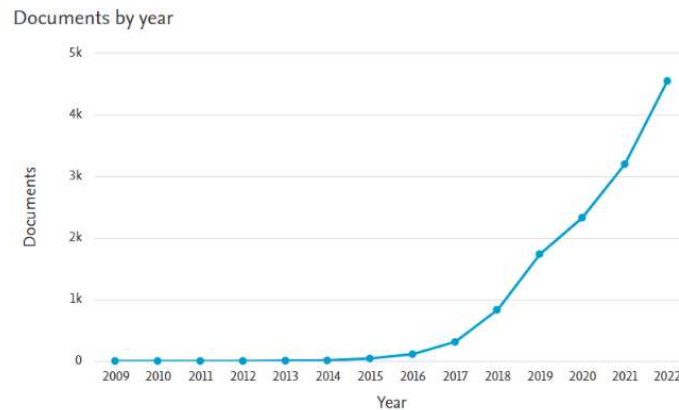


Figure 1.8: Documents indexed by Scopus on "deep learning" AND "object detection" entries over time.

Detection was possible even without deep learning, techniques based on human knowledge or "semi supervised" machine learning techniques, such techniques leverage on known objects' properties to isolate those pixels belonging to potential interesting pixels, if the conditions allow it, the same may be done leveraging on background's properties. The already mentioned techs are shared with instance segmentation/ semantic segmentation.

The colour is a possible sorting criterion when trying to filter pixels, it has been exploited by Dewi [9] to sort grape pixels and tomato pixels; it works when the background's intensity values are bounded in a certain range, colour usage is particularly weak in variable light conditions due to uncontrolled background and reflects, some authors, like Yulua Ekawaty managed to correct bad lightening condition applying "gamma correction", a preprocessing method used to fix non-uniform illumination [12].

Hailing Zhou managed to modelling ocean with GMM (Gaussian Mixture Models) and acquiring all pixels that were different from it like foreground, this is a successful case where known background's properties are exploited to sort it out and detecting objects of interest [13], a visual result is show in Figure 1.9.

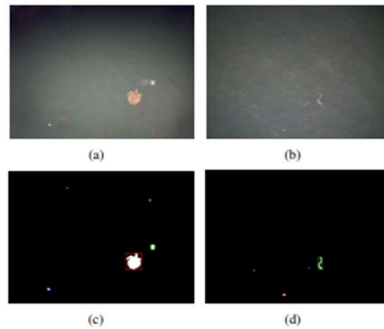


Figure 1.9: Object detection based on background filtering [13].

An object detection application deals with several issues depending on the specific task, variable and unpredictable light conditions are major obstacles when the objects to be detected are located in natural environments or in saturated light conditions, the presence of partially occluded objects are a relevant barrier when the target is to locate targets randomly distributed in front of the camera.

1.2.8.1. Overlapped objects.

Overlapped instances don't affect the performances when dealing with semantic segmentation but still a common problem to solve in those cases where the separation of the instances makes the difference. Common techniques like "erosion" are enough to face slightly overlapped items. "Erosion" modifies a binary mask zeroing those pixels having at least a zeroed pixel the $n \times m$ size kernel centred in the pixel itself, "Erosion" applied to a picture portraying rice grains is shown in Figure 1.10.

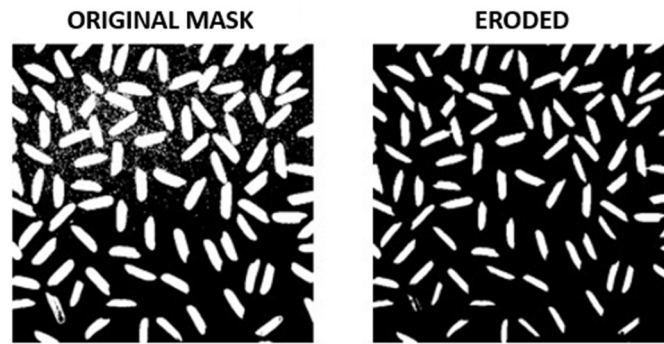


Figure 1.10: Erosion on rice grains image.

There are few sophisticated tools that are used to separate overlapped instances, like filling the foreground with known shapes [14], K-Means and watershed transformation; K-Means is always an option when dealing with objects that are different among them but are internally uniform, in those cases where similar but overlapped instances have been detected, watershed transformation could be applied.

Watershed transformation is a technique invented to separate blood cells exploiting their well-known circle-like shape and their almost constant size, even though gives its best when the input has the already listed properties, this thesis will show that is possible to obtain decent results providing an “unperfect” input only making few changes to the original watershed transformation [15].

Watershed transformation is fed with a binary mask, the algorithm is summarized in three steps:

- A distance map is created: each pixel's value in the input map is turned into the minimum distance between it and the closest background pixel.
- A threshold is applied to the distance map to get nuclei, connected regions we are sure they belong to the objects if the assumptions on input structure are met.
- The nuclei are labelled and expanded to fill the input mask, they expand until they find the background bound or another expanding nucleus is met, each nucleus is assumed to be a different object.

1.2.9. BLOB analysis

BLOBs (Binary Large Objects) are connected components on a binary mask, most of the times, they come from semantic segmentation tools. The binary map tells which pixels are of interest, to distinguish each component one from each other an algorithm named “connected-component labelling” is run. Once the association between objects and pixel is known (BLOB extraction), is possible to calculate a wide range of features on an object level depending on the number of channels of the original image. Figure 1.11 shows an example of BLOB extraction applied to an RGB picture.

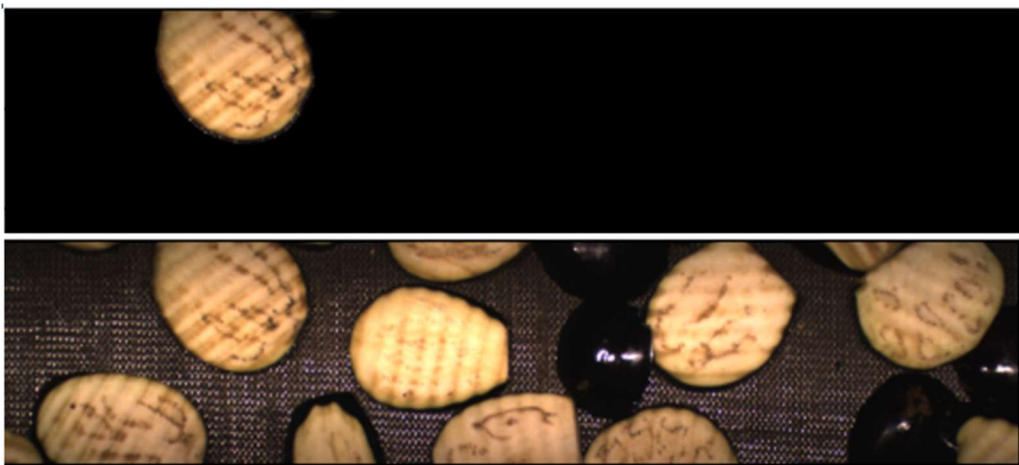


Figure 1.11: Isolated BLOB/ aubergine slice.

The sorting, classification and comparison of those feature is called BLOB analysis.

Here are listed possible features sorted by number of channels (grey scale image, RGB and multispectral) [16] [17] [18] [19]:

- Gray scale features: perimeter, diameter, minor axis length, major axis length, area, circularity, eccentricity, convexity, shape factor, compactness, mean intensity, mass centre and solidity.
- RGB features (they can be calculated for other colourspace too): mean red, mean green, mean blue, std. dev red, std. dev green and std. dev blue.
- Multispectral features: mean in X-Y nm range and spectral variance.

The classic approach to figure out BLOB analysis is different for each specific application, an example applied to Cocoa pods is provided in Figure 1.12 [12].

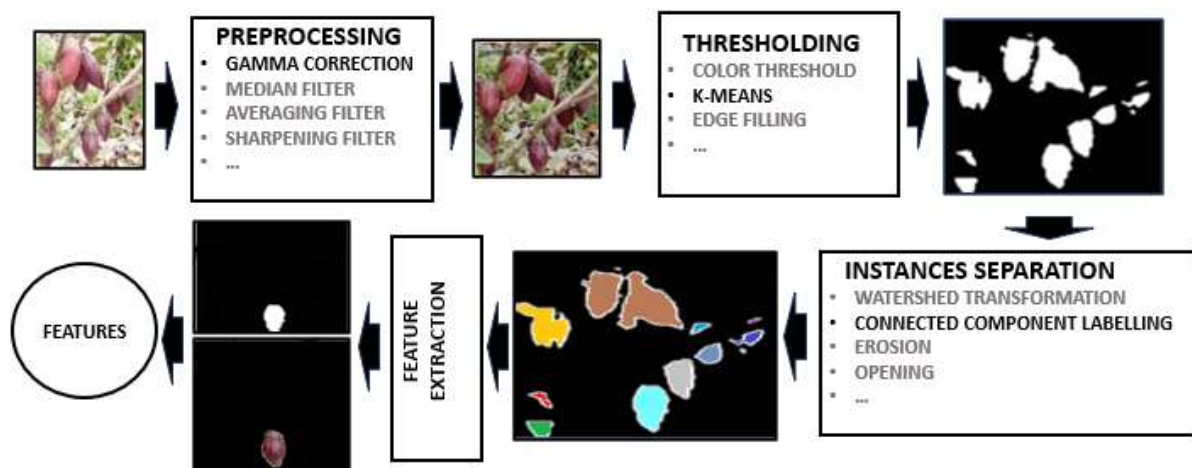


Figure 1.12: Classic BLOB analysis pathway.

The raw image is pre-processed to fix some problems, such tools are usually performed in difficult lightning conditions, then a wide range of technologies is applied with the purpose of sorting out the pixels of interest; in this stage, there are several techniques used to refine the result of the previous step, these procedures help to distinguish the instances and clean the image from noise. The instances are isolated through the “connected component labelling” algorithm which allows to calculate the features for each BLOB. The obtained features are used in a wide range of applications based on classification, like quality inspection [20], quality grading [21] [22] or fruit inspection [23] [24] [25] .

2 Methods

2.1. Workflow

This thesis' workflow is split in three parts, the first involves Yolov8 as well as the necessary procedures to setup the dataset: establishing the classes, dataset labelling, deciding the images mix used to train Yolov8 and hyperparameters setup. The second part is dedicated to set up and run machine learning (ML) algorithms with the purpose of sorting pixels in aubergines classes or background. The output of the ML algorithm reveals better performances is combined with the results of Yolov8, in order to filter out background pixels. The third step is focused on features extraction applied to the BLOBs obtained in the previous phase as well as the description and comparison of such features (Figure 2.1).

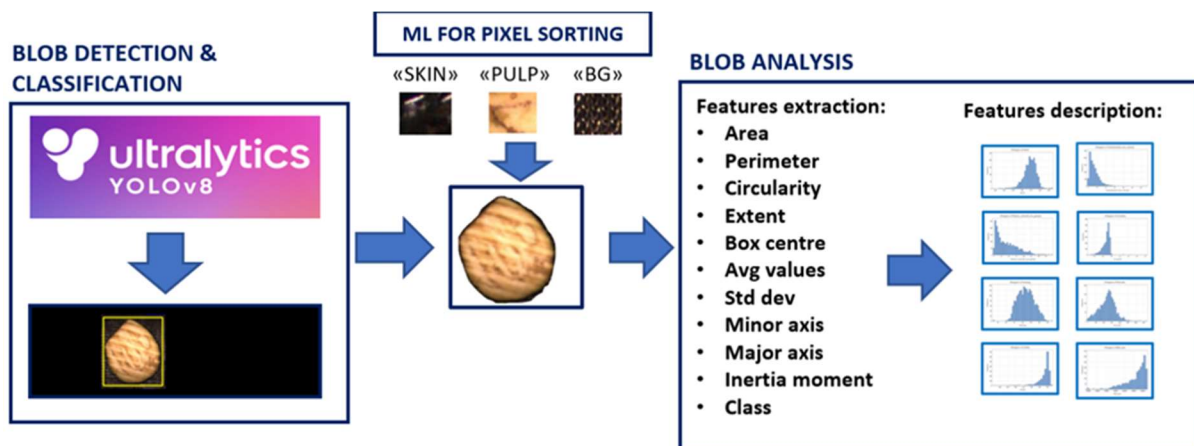


Figure 2.1: Thesis workflow.

2.2. Working dataset

The images used in this thesis has been acquired in a real food production plant called *O.P. COTRAPA 2000*, an Italian food manufacturer and partner of the TESORO project collaborating with *Politecnico di Milano*.

The components used for RGB image acquisition are the following and are shown in Figure 2.2:

- Genie Nano C2420 camera.
- LM8JC optics, with focal distance of 8 mm.
- MidOpt BP550-2 filter, centered in the visible light spectrum.
- SmartVision Lights LW300-WHI-L light emitter.



Figure 2.2: Camera and lightning system setup.

The aforementioned system is applied to an ongoing process into a manufacturing plant owned by *O.P. COTRAPA 2000*, more precisely, the location is between the product selection and the oven inlet, such place is shown in Figure 2.3.



Figure 2.3: Camera system location in COTRAPA.

The pictures' size is 2048x448 pixels and it doesn't include the entire conveyor belt width.

The working material used in this work has been originated from an ongoing sorting process carried out by multiple human operators, the sorting criteria applied are aligned with the company's ones which deviates to the ones presented in this thesis because the sorted product was intended for sale and consumption, as a result, the processed image material was biased and inconsistent, caused by the company's need and the number of people in charge of the sorting respectively.

Aubergine slices were acquired under two conditions, the first condition ("compliant" from now) refers to slices acquired downstream of the sorting process carried out by human operators, here the product is almost completely compliant and properly distributed on the conveyor belt in term of quantity and without significantly overlapped slices, 824 images of this type are available, one of them is show in Figure 2.4. The second part of the dataset was acquired under a different condition (named "discard" from now), it is made of 256 pictures obtained arresting the sorting process and loading an high quantity of product, such product was classified as non-compliant by the operators working on that specific line, as a consequence, the majority of the product doesn't satisfy the quality requirements chosen by the company, the shoots got under this condition doesn't looks like a typical industrial application case since

the slices are highly overlapped and the conveyor belt is overloaded, an example is shown in Figure 2.5. Both “compliant” and “discard” have been acquired on an online working process, therefore the image acquisition process needed to affect the ongoing production process as little as possible, “compliant” pictures portraits the “as is” working conditions while “discard” acquisition caused a production times loss due to the setup procedure needed to reload the product discarded during a previous run; none of the images used in this work represents the expected “to be” working condition, which is similar to “compliant” dataset in term of product arrangement but the relative frequency of the product to be discarded is higher.



Figure 2.4: "Compliant" image sample.



Figure 2.5: "Discard" image sample.

2.3. Yolov8 dataset preparation

Yolov8, like others supervised classification tasks needs to establish the working classes before the technique is applied. The chosen classes are listed:

1. Back/Top: aubergine slices exposing the skin to the camera.
2. Speckled: those products having one or more significant black spot/s.
3. Broken/Holes: this class includes slices that are unintentionally broken or presenting other kind of discontinuities, like holes.

4. Other/Compliant: this class is dedicated to all the other objects released on the conveyor belt that can't be sorted into the previous three classes. Most of the material in this class is expected to be compliant slices.

Few samples for the already described classes are shown in Figure 2.6.

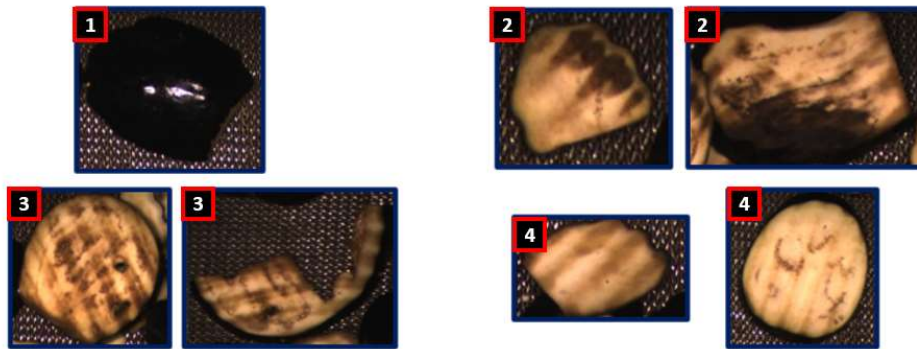


Figure 2.6: Class samples: "1" is "Back/Top", "2" is "Speckled", "3" is "Broken/Holes", "4" is "Other/Compliant".

Even though the acquisition was made under two different conditions with the purpose of acquiring more non-compliant products, not all the “discards” objects belong to class “Back/Top”, “Speckled” and “Broken/Holes”, and the same applies to the product under “compliant” condition, they don't belong entirely to “Other/Compliant” class. There are some reasons behind that: the sorting process is affected by human errors, the slices may hide the deciding defect on the side which is occluded to the camera and there are defects that aren't covered by “Back/Top”, “Speckled” and “Broken/Holes” classes but still relevant for the food manufactured when the images were acquired.

The images were labelled using *LabelImg*, an open-source software written in python that allows to label in the most common formats through a user-friendly interface (Figure 2.7).

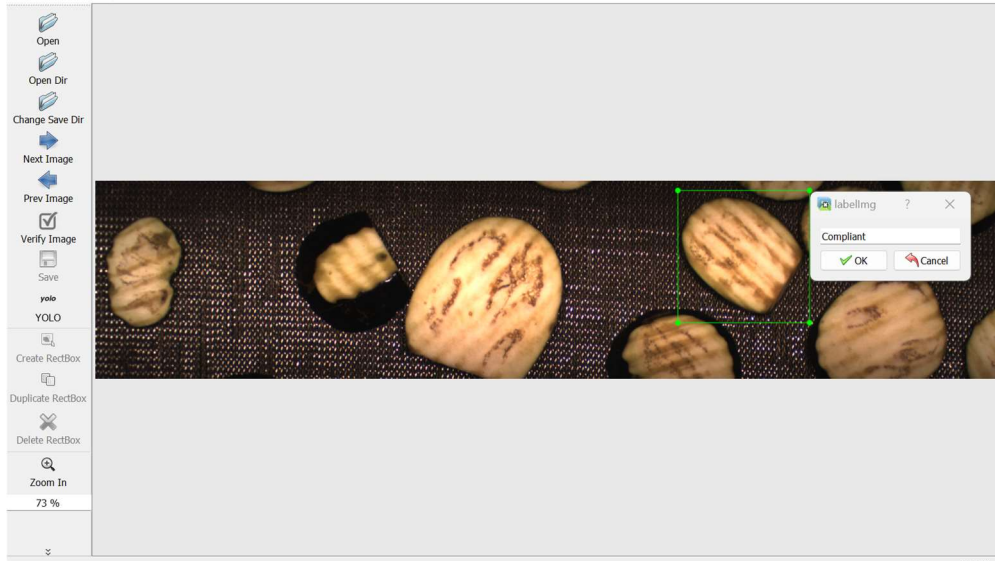
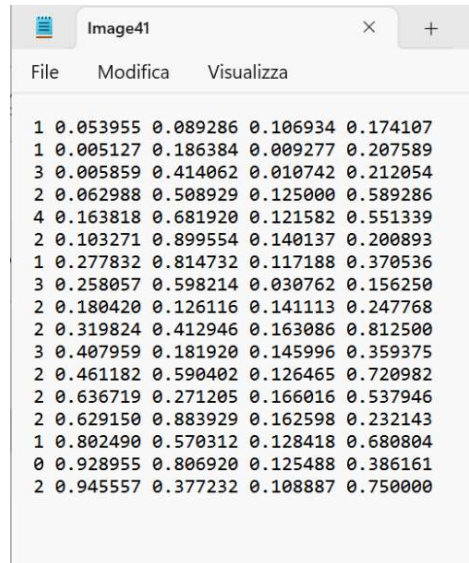


Figure 2.7: *LabelImg* interface.

LabelImg gives the possibility to save the annotation files in “YOLO” format, it is the standard used by Yolo’s family, the standard requires one annotation file in “.txt” format for each annotated image and, in addition, a further “.txt” file that codifies the classes. Each row identifies a bounding box which is made of five information pieces separated by “spacebar” key; the first data is an integer number that codifies for a specific class, the remaining data are float numbers that represents the box’s centre coordinates as well as its width and height, normalized with respected to the whole image size, an example is given in Figure 2.8. The annotation file that codifies the classes’ names is a simple list of the true class names separated by “enter” key, the sequential position of the name defines the integer number used as its proxy in the other annotations files, *LabelImg* produces this file too and automatically names the annotation file as the image it came from. The class defining file is shown in Figure 2.9.

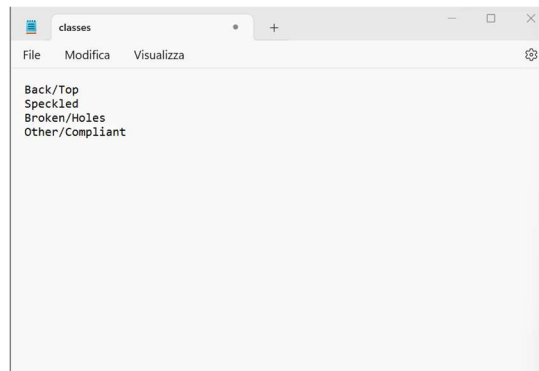


```

1 0.053955 0.089286 0.106934 0.174107
1 0.005127 0.186384 0.009277 0.207589
3 0.005859 0.414062 0.010742 0.212054
2 0.062988 0.508929 0.125000 0.589286
4 0.163818 0.681920 0.121582 0.551339
2 0.103271 0.899554 0.140137 0.200893
1 0.277832 0.814732 0.117188 0.370536
3 0.258057 0.598214 0.030762 0.156250
2 0.180420 0.126116 0.141113 0.247768
2 0.319824 0.412946 0.163086 0.812500
3 0.407959 0.181920 0.145996 0.359375
2 0.461182 0.590402 0.126465 0.720982
2 0.636719 0.271205 0.166016 0.537946
2 0.629150 0.883929 0.162598 0.232143
1 0.802490 0.570312 0.128418 0.680804
0 0.928955 0.806920 0.125488 0.386161
2 0.945557 0.377232 0.108887 0.750000

```

Figure 2.8: Annotation file in "Yolo" standard.



```

Back/Top
Speckled
Broken/Holes
Other/Compliant

```

Figure 2.9: Class defining file in "Yolo" standard.

2.4. Yolov8x

Yolov8x is the heaviest Yolov8's network, it has been chosen due to the importance of the precision when compared to the inference speed, in fact, the last depends on the subversion. However, the most impacting leverage on this performance is proved to be the hardware and the software used for running the inference process which have great potential.

The model set up was carried out with the convenient interface supplied by *Ultralytics*, it is inserted in a package named "Ultralytics", it is written in "Python". "Ultralytics" isn't the only "Python" package required to run Yolov8x; the developers released "requirements.txt" on Yolov8 official repository, running this file with "pip3", all the required packages are installed all in one go, "pip3" is a package manager tool

dedicated to “Python”. The packages to be installed are numerous, they include general purpose packages like “Matplotlib”, “NumPy” and “pandas” but also machine learning oriented tools like “nvidia-pyindex”, “TensorFlow” and “PyTorch”.

To run the training is necessary to specify which subvariant is desired, where the algorithm is supposed to find the training set and the hyperparameters. In most of the cases, the new model is initialized with “preloads”, “preloads” is nothing but an alias of the model’s parameters in case they aren’t initialized randomly. The preloads are useful even if they are specialized on detecting objects different from the ones the user is interest in, in fact, initializing the model with preloads coming from a completely different training session is better than initialize the parameters with random values because it saves training time, for this reason, the training performed in this thesis is “preloaded” with the parameters from Yolov8x trained on COCO (Common Objects in Context) dataset(<https://cocodataset.org/#overview>).

The hyperparameters set used in this case are summarized in Table 2.1.

Table 2.1: Yolov8x training hyperparameters.

Hyperparameter	Value	Default	Description
Epochs	400	100	Maximum number of training cycles.
Patience	50	50	Epochs to wait without significant improvements.
Batch	2	16	Number of images for each iteration.
Size	640	640	Size of input image is Size x Size .
Mosaic	True	True	Include mosaic augmentation.
Optimizer	auto	auto	Type of optimizer, "auto" means that "AdamW" is used up to 10,000 iterations, then "SDG" for the surplus.
Seed	0	0	Random seed value for reproducibility.
AMP	True	True	Usage of Automatic Mixed Precision training, a speed up tool for deep learning applications.
Fraction	1.0	1.0	Fraction of the designated training set to use in the training phase.
LR0	0.01	0.01	Starting learning rate.
LRF	0.01LR0	0.01LR0	Final learning rate.
OP1	0.937	0.937	This value is the momentum when using "SDG" and beta1 if "Adam" optimizers family is in play.
Decay	0.0005	0.0005	The optimizer decay value penalizes larger weights in the attempt to avoid overfitting.
WU epochs	3.0	3.0	Warmup epochs.
WU momentum	0.8	0.8	Momentum value during the warmup.
WU LR bias	0.1	0.1	It is the starting learning rate in the warmup, starting from this value, LR approaches LR0 linearly.
BOX loss	7.5	7.5	Box loss gain.
CLS loss	0.5	0.5	Classification loss gain.
DFL loss	1.5	1.5	Dual Focal loss gain.

The maximum number of epochs has been increased to avoid early stoppage during the training, the batch size was modified too, higher batch sizes require higher

memory consumption, however, the GPU used to run the training is “ASUS GeForce GTX 1050Ti”, which is unable to work under the default batch size condition (16 images for batch) due to its limited resources.

The dataset has been split following the usual proportion:

- 70% for “training set”.
- 20% for “validation set”.
- 10% for “test set”.

The “training set” and “validation set” are made by 50/50 mix of “discard” and “compliant” images while “test set” is made solely by “complaint” images. Although the inference on “discard” images lead to poor results due to the high density of overlapped products, they have been included in the “training set” and “validation set” because of their preciousness in shortage conditions, in fact, this technology works properly when trained with several images. The instances were mixed before the sorting procedure.

The material used to feed Yolov8x is summarized in Table 2.2.

Table 2.2: Yolov8x dataset composition.

Set split	Compliant	Discard	Total
Train	179	179	358
Validation	51	51	102
Test	51	0	51

Since the entire “compliant” dataset undergoes to the workflow, a method to apply the Yolov8x training’s output is necessary, “Ultralytics” package offers a convenient toolbox for the inference phase too. The “prediction” needs the following inputs: images, the network model and hyperparameters; the output is nothing but a list of Yolo annotation files, one for each processed image.

The hyperparameters of interest that could be modified in this phase are show in Table 2.3.

Table 2.3: Yolov8x prediction hyperparameters.

Hyperparameter	Value	Default	Description
IOU NMS	0.7	0.7	IoU value threshold used in non-max suppression.
Confidence	0.25	0.25	Confidence score value threshold.
Size	640	640	Size of input image is Size x Size .

2.5. Pixel sorting

Some machine learning algorithms were trained to predict whether a pixel belongs to the product or not, in order to facilitate the candidate models and understand the results, two classes have been assigned to the foreground: aubergine pixels are sorted in “Pulp” and “Skin” classes, they represent the internal part of the product and the vegetable skin respectively.

Most of the proposed machine learning techniques are supervised, therefore, they require a ground truth to be trained. 50 pictures from “compliant” dataset were selected and two masks for each of them were drawn, these masks cover “Pulp” and “Skin” pixels. Masks creation was carried out by hand through *Photoshop.exe*; the masks look like the ones shown in Figure 2.10. The proposed algorithms were applied through “Scikit-learn”, a python library package specialized on machine learning applications while the data management relies on “Pandas” library.

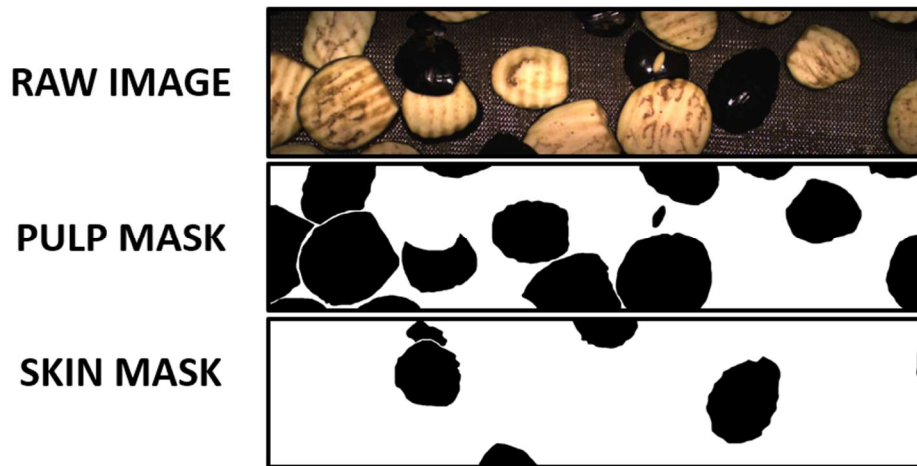


Figure 2.10: Pixel level mask drawn through *Photoshop.exe*.

Those pixels that have been classified as foreground in both the masks have been excluded by the process, their frequency account for 0.2% of the total. The pixel values are shuffled and split in “train set” and “test set” following a 90/10 proportion, the machine learning tools are fed with the pixel values, the most promising technique was selected after the performance comparison, F score has been used as deciding performance for the supervised models; the whole process is show in Figure 2.11.

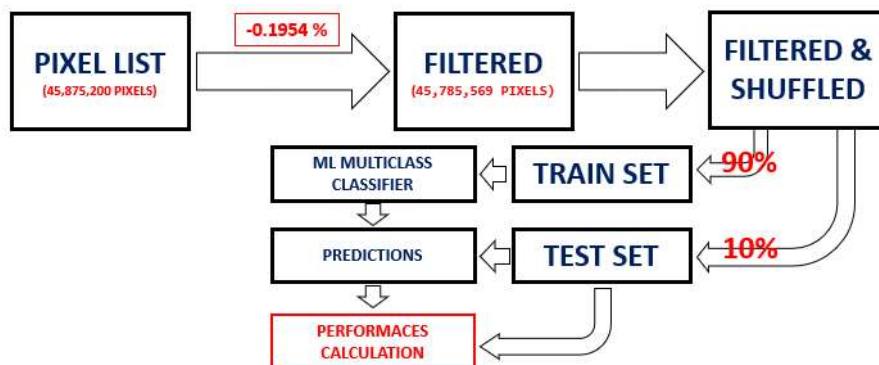


Figure 2.11: ML for pixel sorting workflow.

Even though the RGB colourspace is the most common representation in imaging and machine vision field, others colour representations have high potential, depending on the application. In this case, 4 colourspaces were used to feed machine learning applications, they are “RGB”, “HSV”, “Luv” and “YCbCr”; all the already listed colourspaces are made of three values and can be calculated from “RGB”, which is the

standard for the raw images used in this thesis. The interpretation of the channels values is provided in Table 2.4.

Table 2.4: Colour channels description.

Colourspace	Channel	Description
RGB	R	Red component.
RGB	G	Green component.
RGB	B	Blue component.
HSV	H	Hue component.
HSV	S	Saturation component.
HSV	V	Value: defines the black quantity.
Luv	L	Intensity component.
Luv	u	First chromaticity coordinate.
Luv	v	Second chromaticity coordinate.
YCbCr	Y	Luminance component.
YCbCr	Cb	Blue scale component.
YCbCr	Cr	Red scale component.

2.5.1. SDG application

“Scikit-learn” package’s SDG multiclass classifier proposes SVM (Support Vector Machine) as support algorithm, the approach is called “One-vs-All”, it means that multiple SVM are trained, as many as the classes. When coming to the prediction task, the trained models predict a score independently, the scores represent how “far” is the proposed instance in the positive region (positive score) or in the negative region (negative score), the higher score defines which class the new instance belongs to. A graphical explanation of SVM algorithm in One-vs-All approach is provided in Figure 2.12.

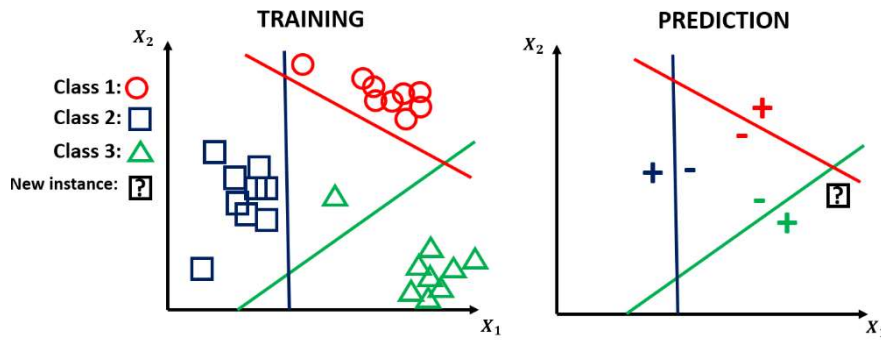


Figure 2.12: One-vs-All Support Vector Machine (SVM).

SDG classifier has few available parameters, the once used in this thesis are listed in Table 2.5.

Table 2.5: SDG parameters.

Parameter	Value	Description
Loss Function	hinge	Linear SVM loss function.
Penalty	L2	What kind of penalty function should be applied, L2 is the standard regularized penalty.
Alpha	0.0001	Regularization term gain.
Iter_Max	10000	Maximum number of iterations.
E_Stop	None	Early stop criterion.
LR	optimal	Learning rate function, "optimal" is a function that decrease through each iteration, it depends on Alpha parameter.
Validation	0.1	Validation set as percentage of training set.
Stop	0.001	If the calculated loss is larger than the best experience loss score minus Stop for NumIter in a row, then the training is stopped.
NumIter	5	Number of iterations in a row that cause training stoppage if the designated criterion is met.

2.5.2. Softmax Regression Classifier application

Softmax is nothing but the Logistic classification applied to multiple classes, this algorithm tries to guess the probability that a new instance belongs to a given class, the higher probability drives the decision. The parameters listed in Table 2.6 were used in this thesis to run Softmax.

Table 2.6: Softmax parameters.

Parameter	Value	Description
Solver	lbfgs	Is the solver algorithm used by scikit-learn; "lbfgs" approximates Broyden-Fletcher-Goldfarb-Shanno algorithm.
Penalty	L2	What kind of penalty function should be applied, L2 is the standard regularized penalty.
Alpha	1	Regularization term gain.
Iter_Max	10000	Maximum number of iterations.
Tolerance	0.001	Tolerance for the stop criterion.

2.5.3. Decision Tree application

"Scikit-learn" offers a package to train "Decision Tree" on custom datasets, it is a supervised machine learning algorithm. The parameters used to run the algorithm are listed below, in Table 2.7.

Table 2.7: Decision Tree parameters.

Parameter	Value	Description
Criterion	Gini	The function to be used as criterion to decide the split.
MAX_depth	6	Maximum Tree's depth.
MIN_node_split	2	Minimum number of samples to split a "leaf".
MIN_leaf_split	1	Minimum number of samples to make a "leaf".
MIN_imp_decr	0.0	During the "Tree growth", if a split decreases the impurity less than MIN_imp_decr , such split doesn't happen.

2.5.4. Random Forest application

Although Random Forest is one of the most promising supervised machine learning algorithms, it requires a lot of time to be trained depending on the "forest" size, the same is true for the prediction stage. Random Forest offers few parameters to rank features based on their importance, therefore Random Forest has been used to exclude the less relevant features (image channels). In this thesis, when Random Forest has been applied, the parameters shown in Table 2.8 have been used.

Table 2.8: Random Forest parameters.

Parameter	Value	Description
Trees	50	Number of "Trees" in the "forest".
Criterion	Gini	The function to be used as criterion to decide the split.
MAX_depth	7	Maximum Tree's depth.
MIN_node_split	2	Minimum number of samples to split a "leaf".
MIN_leaf_split	1	Minimum number of samples to make a "leaf".
Feat_split	sqrt	The number of features to consider when split a node, "sqrt" is the squared root of the number of features. The candidate features are chosen randomly.
MIN_imp_decr	0.0	During the "Tree growth", if a split decreases the impurity less than MIN_imp_decr , such split doesn't happen.

2.5.5. DBscan application

For comparison, few unsupervised algorithms were trained, DBscan is one of them, it was built and used to feed a K-nearest neighbours (KNN) model. The result would be successful if DBscan is able to build up groups that are pure in term of ground truth classes. DBscan was trained with the parameters listed in Table 2.9.

Table 2.9: DBscan parameters.

Parameter	Value	Description
eps	1.0	Distance parameter, named "epsilon".
Core_thresh	800	Minimum number that defines a training instance as core point.
Algorithm	Ball tree	The algorithm to be used to find the nearest neighbours.

When coming to KNN, the following parameters (Table 2.10) are applied.

Table 2.10: KNN parameters.

Parameter	Value	Description
N_neighbours	30	Number of closest neighbours to be found.
Algorithm	Ball tree	The algorithm to be used to find the nearest neighbours.

2.5.6. K-Means application

K-Means is an unsupervised algorithm, the user defines the number of clusters. In this case, K-Means has been run changing the number of clusters, they range from 3 up to 13. The best number of groups has been decided looking at the higher “silhouette score”. Minibatching was required due to the high number of samples to be processed. The same parameters were used to run K-Means, only “K” changed, such parameters are shown in Table 2.11.

Table 2.11: K-Means parameters.

Parameter	Value	Description
K	3-13	Number of clusters.
Iter_Max	100	Maximum number of iterations.
Batch	1024	Batch size.
NumIter	10	Number of iterations in a row that cause training stoppage if the designated criterion is met.

2.6. Features extraction

Refinement tools, Yolov8x, and pixel sorting are all part of the feature extraction process. The refining tools are a collection of techniques used to address substantial issues put on by machine learning's poor performance. The whole process aims to get features for each BLOB.

The process starts when Yolov8x locates the aubergine slice and its class, for the way Yolov8x works, it identifies the object through a rectangle. From this rectangle, the pixels are filtered according with the predicted class. The products belonging to

“Back/Top” are made almost exclusively by “Skin” pixels whereas the other classes are associated to “Pulp” pixels, hence only one type of pixel is extracted from the drawn rectangle. This procedure is accomplished for every Yolov8x’s output line unless the box’s centre fall within 250 pixels from the image’s vertical border or 75 pixels from the horizontal border, those products are excluded because is challenging to reconstruct the true shape for those products that aren’t completely included in the picture. The intermediate result and the first part of the process looks like the ones shown in Figure 2.13: Random Forest applied to Yolov8x. The BLOB isn’t extracted immediately since few problems still in place, in fact, Yolov8x’ rectangle may include more than one complete slice or exclude parts of the object, moreover, “Back/Top” products are affected by a relevant lack of pixels due to presence of reflects.

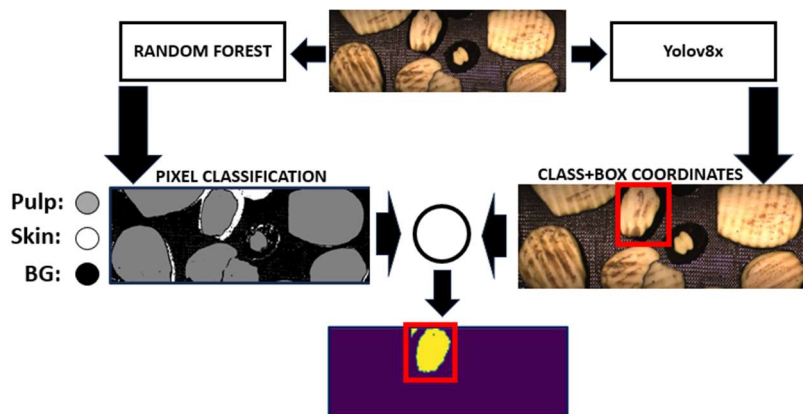


Figure 2.13: Random Forest applied to Yolov8x.

Only the largest BLOB inside the rectangle is chosen and “expanded”, this means that the pixels contiguous to the BLOB and belonging to the same pixel class are included in the BLOB itself, such procedure tackle those cases where the aubergine slice is partially located outside the rectangle; this solution introduces an extra problem if the target is overlapped (or extremely close) with one or more slices of the same pixel class, along with the target, the neighbouring slice/slices may be fully enclosed in the BLOB. The previously obtained BLOB undergoes to “closing”. “Closing” is a morphological transformation which applies to single channel masks, it is a “dilation” followed by an “erosion”, always in this order, “dilation” is the opposite of “erosion”, it turns positive

the pixels close to the once that are already positive; as its name suggests, “closing” close small and large gaps inside the slices, a critical problem that affects “Back/Top” more than the remaining classes. In this thesis, the kernel is a circle 30 pixels in diameter.

Watershed transformation is applied in order to separate the target from the neighbouring products. Since aubergine slices don't meet the ideal conditions to apply watershed transformation, the tool has been properly modified. Watershed transformation relies on a threshold cut to extract the nuclei from the distance map, to do so, is necessary to define a threshold value which is inflexible to the object size and causes larger nuclei to overcome the smaller generating a highly distorted border. Distance map associates to each foreground pixels a value equal to the minimum distance between it and the background, an example of distance map applied to all foreground elements can be visualized in Figure 2.14.

Watershed transformation was modified in the nuclei extraction method, the distance map is processed by means of a median filter and the local maxima of such map are recorded.

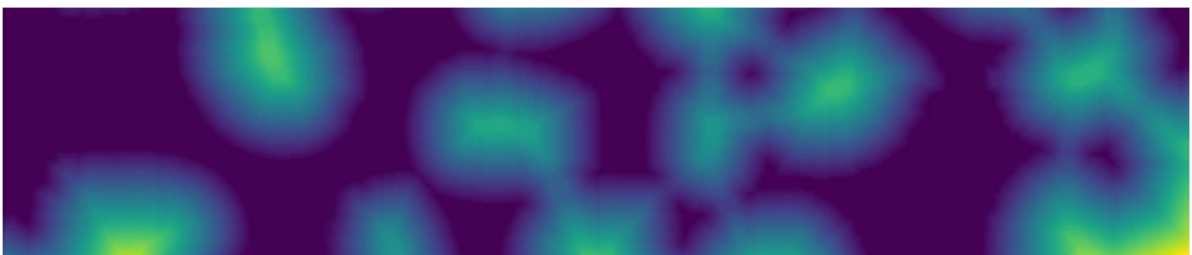


Figure 2.14: Distance map on aubergines slice.

The nuclei are circles located in the local maxima having a diameter equal to a half of the pick value. The core growth and borders definition are the same used in the original watershed transformation [15](Yi and Zhou, 2000) (1.2.8.1).

Watershed transformation output is a group of BLOBs that are close each other but no more contiguous due to the borders introduced by watershed itself; only one of such BLOBs is the target, it is selected in three steps:

- Candidates BLOBs having an area equal to 5 pixels or less are deleted.
- For each residual BLOB, calculate the distance between the centre of the rectangle circumscribed to the BLOB and the centre of the Yolov8x output rectangle.
- Chose the closer BLOB.

Once target BLOB is located, dimensional and shape-based features can be measured, all the residual features are extracted superimposing the BLOB to the original RGB picture.

A list of the feature taken into consideration while writing this thesis are displayed in Table 2.1 as well as their description.

Table 2.12: Features list.

Feature	Description
Box distance	Is the centre-centre distance between the rectangle circumscribed to the BLOB and the rectangle coming from Yolov8x; the unit of measure is “pixels”.
Area	Number of pixels that makes the BLOB.
Perimeter	BLOB perimeter length in pixels.
Circularity	It measures how the BLOB is similar to a circle, it is calculated by Equation (2.1).
Solidity	It measures the BLOB convexity. Is the ratio between Area and Hull area.
Extent	Is the ratio between Area and the area of the rectangle circumscribed to the BLOB.
G_avg	Is the average value in the green channel.
G_stddev	Is the sample standard deviation in the green channel.
B_avg	Is the average value in the blue channel.
B_stddev	Is the sample standard deviation in the blue channel.
R_avg	Is the average value in the red channel.
R_stddev	Is the sample standard deviation in the red channel.
H_avg	Is the average value in the hue channel.
H_stddev	Is the sample standard deviation in the hue channel.
S_avg	Is the average value in the saturation channel.
S_stddev	Is the sample standard deviation in the saturation channel.
V_avg	Is the average value in the V channel.
V_stddev	Is the sample standard deviation in the V channel.
Major_axis	Is the major axis length of the ellipse that approximates the BLOB, measured in pixels. The elliptical approximation is carried out by <i>opencv.fitEllipse</i> function.
Minor_axis	Is the minor axis length of the ellipse that approximates the BLOB, measured in pixels. The elliptical approximation is carried out by <i>opencv.fitEllipse</i> function.
Inertia_X	Is the second momentum of the BLOB, calculated in BLOB’s centre with respect to the X axis passing through the centroid.
Inertia_Y	Is the second momentum of the BLOB, calculated in BLOB’s centre with respect to the X axis passing through the centroid.

$$Circularity = \frac{4\pi Area}{Perimeter^2} \tag{2.1}$$

2.7. Feature description & Feature comparison

The features are collected and sorted by class. For each feature-class combination the sample mean and the sample standard deviation are calculated, such data are useful to describe the products, moreover the features are potentially able to distinguish between the classes. In this thesis, the t-test is applied to each class couple in order to test whether or not their mean is the same, and this is done for each feature. In case the hypothesis that the mean difference isn't equal to zero, the feature may be used to distinguish between two classes, all the statistical tests are performed with alpha equal to 5%. Figure 2.15 summarizes t-test characteristics. The calculations and the statistical tests are performed by means of *Minitab.exe*.

Two sample t-test for feature comparison

<i>Null hypothesis</i> $H_0: \mu_{class_X1}^Y - \mu_{class_X2}^Y = 0$	$\forall Y$ feature
<i>Alternative hypothesis</i> $H_1: \mu_{class_X1}^Y - \mu_{class_X2}^Y \neq 0$	$\forall (X1, X2)$ class combination
	$\alpha = 5\%$ (type I error probability)
	S : sample standard deviation over the class
	\bar{X} : class average
	N : number of instances for that class

$$Statistic\ test\ T_0: \frac{\bar{X1} - \bar{X2}}{\sqrt{\frac{S_{X1}^2}{N_{X1}} + \frac{S_{X2}^2}{N_{X2}}}}(Y)$$

Figure 2.15: T-test for feature comparison.

3 Results

3.1. Yolov8x results

Yolov8x performances can be represented in several ways, confusion matrix is one of them, “Ultralytics” package offers a function dedicated to model’s performances evaluation, it provides a confusion matrix and the trade-off profile related to precision, recall and F score. The training has been stopped after 81 training epochs, the resulting cross-class confusion matrix is shown in Figure 3.1.

Confusion matrix		True				
		Back/Top	Speckled	Holes/Broken	Other/Compliant	Background
Predicted	Back/Top	0.9	0.00	0.00	0.01	0.30
	Speckled	0.01	0.49	0.24	0.02	0.16
	Holes/Broken	0.01	0.01	0.21	0.01	0.07
	Other/Compliant	0.03	0.47	0.56	0.92	0.48
	Background	0.06	0.03	0.00	0.03	-

Figure 3.1: Yolov8x confusion matrix, values as percentage of the true instances for that class. This matrix was calculated with a fixed IoU threshold value, equal to 0.45, when comparing prediction and ground truth, only those boxes having IoU larger than 0.45 with a ground truth of the same class are accounted as true positives.

Yolov8x is able to detect “Back/Top” instances, it is expected since this class is well represented in the dataset and is fairly different with respect to the other classes, 10 % of the instances belonging to “Back/Top” are misclassified and, in those cases, half of the times are classified as “Background”. “Speckled” and “Holes/Broken” classes are characterized by poor performances, “Speckled” instances are misclassified as “Other/Compliant” half of the times while “Holes/Broken” are predicted as

“Other/Compliant” in 56% of the cases, the remaining are almost evenly distributed between the true positive (“Holes/Broken”) and “Speckled” class; “Other/Compliant” products are retrieved 92% of the times. In case the IoU value between the prediction and ground truth boxes is low, the prediction is recorded in the last column, this happened 48% of the times when trying to predict the location of a “Other/Compliant” object.

The localization performance is one of the best achievements in this case, the “box loss”, a performance measuring the deviation between the predicted box and the ground truth, is equal to 0.55.

Yolov8x applied to the whole dataset led to the detection of 8,835 objects, distributed in the four classes, Table 3.1 shows the relative frequency for the ground truth instances too.

Table 3.1: Yolov8x predictions on "compliant" dataset vs ground truth.

Class	#Prediction instances	Relative frequency on predictions	Relative frequency on ground truth
Back/Top	1220	13.8%	11.6%
Speckled	168	1.9%	7.9%
Holes/Broken	80	0.6%	1.3%
Other/Compliant	7397	83.7%	79.2%

3.2. Machine learning results

3.2.1. Supervised ML algorithms results

Supervised machine learning algorithms were compared based on confusion matrix performances.

SGD for pixel sorting has been applied and the results are shown in Figure 3.2, the deciding performance is the weighted F score, the second most relevant performance

is the recall on “Skin” class that looks the weaker performance caused by the reflects on slices skin.

STOCHASTIC GRADIENT DESCENT

Confusion matrix		True		
		BG	Pulp	Skin
Predicted	BG	0.93	0.04	0.07
	Pulp	0.03	0.96	0.00
	Skin	0.04	0.00	0.93

	Precision	Recall	F-score
BG	0.93	0.95	0.94
Pulp	0.96	0.97	0.97
Skin	0.93	0.74	0.82
Weighted avg	0.95	0.95	0.95

Figure 3.2: SGD confusion matrix and performances.

While the F score is equal to 0.95, the recall on the “Skin” class is equal to 0.74, such performance is significantly lower than the same for the other two classes and it impacts negatively on the F score associated to the “Skin” class; the same applies to most of the machine learning algorithms used in this thesis.

Softmax Regression classifier results are show in Figure 3.3.

SOFTMAX REGRESSION

Confusion matrix		True		
		BG	Pulp	Skin
Predicted	BG	0.93	0.02	0.08
	Pulp	0.03	0.98	0.00
	Skin	0.04	0.00	0.92

	Precision	Recall	F-score
BG	0.93	0.96	0.95
Pulp	0.98	0.96	0.97
Skin	0.92	0.78	0.85
Weighted avg	0.95	0.95	0.95

Figure 3.3: Softmax Regression confusion matrix and performances.

Softmax Regression Classifier performances are better when compared with SGD, even though the overall weighted F score is the same, Softmax Regression performs better on predicting “Skin” pixels, its recall on this class is 0.04 higher.

Decision Tree Classifier results are show in Figure 3.4.

DECISION TREE

Confusion matrix		True		
		BG	Pulp	Skin
Predicted	BG	0.94	0.02	0.09
	Pulp	0.03	0.98	0.00
	Skin	0.03	0.00	0.91

	Precision	Recall	F-score
BG	0.94	0.96	0.95
Pulp	0.98	0.97	0.97
Skin	0.91	0.81	0.86
Weighted avg	0.95	0.95	0.95

Figure 3.4: Decision Tree confusion matrix and performances.

Decision Tree multiclass classifier performed better than SGD and Softmax classifier, although weighted F score still equal to 0.95, the recall in “Skin” gain 0.03 over Softmax classifier.

Random Forest classifier was applied to all channels listed in Table 2.4 and the results are show in Figure 3.5.

RANDOM FOREST

Confusion matrix		True		
		BG	Pulp	Skin
Predicted	BG	0.94	0.02	0.09
	Pulp	0.03	0.98	0.00
	Skin	0.03	0.00	0.91

	Precision	Recall	F-score
BG	0.94	0.96	0.95
Pulp	0.98	0.97	0.97
Skin	0.91	0.81	0.86
Weighted avg	0.95	0.95	0.95

Figure 3.5: Random Forest confusion matrix and performances, 12 features run.

Random Forest classifier is slightly better than Decision Tree, it offers a feature ranking too. Random Forest were run deleting the features having an “importance score” lower than 0.1. The importance scores and the feature used in the second run are listed in Table 3.2.

Table 3.2: Features ranked by Random Forest classifier.

Channel	Importance score	Retained
R	0.1039	Yes
G	0.0367	No
B	0.0099	No
H	0.0050	No
S	0.0205	No
V	0.0873	No
L	0.0439	No
u	0.1050	Yes
v	0.1773	Yes
Y	0.0463	No
Cb	0.2304	Yes
Cr	0.1333	Yes

Random Forest was rebuilt feeding the model with R, u, Cb and Cr features in the attempt to reduce the inference time; the same happened for the maximum “Tree’s depth” and the maximum number of “trees” in the “forest”, they were reduced to 4 and 50 respectively. Figure 3.6 shows the new performances.

RANDOM FOREST (5 FEATURES)

Confusion matrix		True		
		BG	Pulp	Skin
Predicted	BG	0.92	0.02	0.08
	Pulp	0.05	0.98	0.00
	Skin	0.03	0.00	0.92

	Precision	Recall	F-score
BG	0.92	0.97	0.94
Pulp	0.98	0.95	0.97
Skin	0.92	0.79	0.85
Weighted avg	0.95	0.95	0.95

Figure 3.6: Random Forest confusion matrix and performances, 5 features run.

The performance loss caused by the model’s dimensional reduction still limited if compared with the dimensional reduction degree, in fact, F score still equal to 0.95 and

the recall is close the full dimension model's one. The last is the chosen one to be inserted in the thesis' workflow as show in Figure 2.1.

3.2.2. Unsupervised/Semisupervised ML algorithms results

DBscan+KNN has been evaluated as an alternative to supervised algorithms, a good result is achieved when the clusters are pure in terms of class frequency. DBscan identified 8 clusters and, the test instances were classified in these groups. The result is show in Figure 3.7.

DBSCAN+KNN

CLUSTER	1	2	3	4	5	6	7	8
CLUSTER WEIGHT	0%	1%	1%	1%	7%	81%	9%	0%
BG	2%	1%	1%	30%	45%	29%	45%	2%
Pulp	0%	0%	0%	0%	0%	40%	0%	0%
Skin	97%	99%	99%	70%	55%	31%	54%	98%

Figure 3.7: DBscan+KNN cluster purity.

DBscan+KNN detected 4 highly "pure" clusters, they are: cluster 1, 2, 3 and 8, instances belonging to those groups are classified as "Skin"; cluster 6 is the only one retaining "Pulp" pixels but its purity degree is only 40%; the remaining clusters are distributed between "Background" and "Skin" pixels with a proportion that is close to 50%, this result is aligned to the lack in recall experienced by the supervised algorithms due to the reflects on product's skin.

When dealing with K-Means, the user is in charge of selecting the number of clusters used to run the algorithm, for this reason, a "silhouette score" has been computed, it is plotted in Figure 3.8.

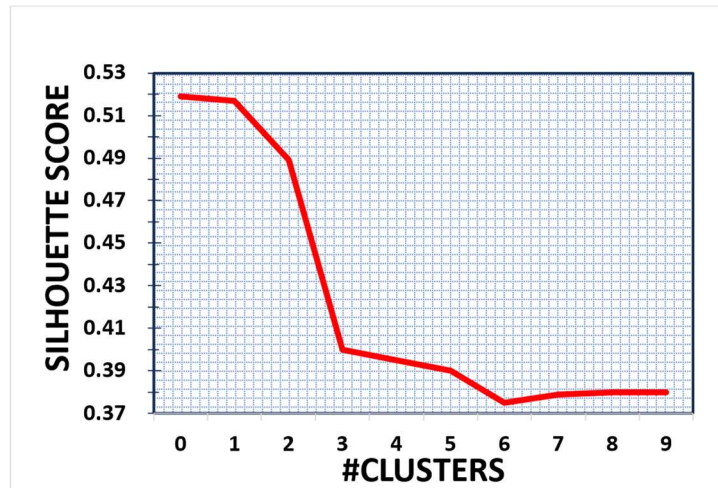


Figure 3.8: Silhouette score vs #clusters.

The silhouette score doesn't show its maximum at a meaningful number of clusters; therefore, a large "K" was selected, equal to 9. K-Means was tested, and class frequency for each group was measured, the result is show in Figure 3.9.

K-MEANS

CLUSTER	1	2	3	4	5	6	7	8	9
CLUSTER WEIGHT	10%	10%	8%	13%	10%	8%	22%	4%	16%
BG	44%	7%	5%	47%	48%	3%	48%	15%	6%
Pulp	9%	85%	90%	0%	4%	94%	1%	0%	34%
Skin	47%	7%	5%	53%	48%	3%	51%	85%	60%

Figure 3.9: K-Means cluster purity.

K-Means can identify "Pulp" pixels, they are enclosed in the second, third and sixth cluster, the residuals are similar to the clusters find by DBscan+KNN, those clusters struggle to separate "Skin" and "Background" pixels.

3.3. Features extraction results

The feature extraction process has been applied as described in 2.6, the resulting BLOBs are less than the ones listed in Table 3.1 since those predictions placed close to the border have been excluded from the process; the residual BLOBs are summarized in Table 3.3.

Table 3.3: Yolov8x predictions excluding elements close to the border.

Class	#Instances	Relative frequency
Back/Top	731	12.2%
Speckled	148	2.5%
Holes/Broken	59	1.0%
Other/Compliant	5047	84.3%

Mean and sample standard deviation have been calculated for each class-feature, the output values are listed in Table 3.4 and Table 3.5.

Table 3.4: Features mean by class.

Feature	Back/Top	Speckled	Holes/Broken	Other/Compliant
Box distance	38.51	25.12	15.67	13.16
Area	18,340	24,895	38,218	39,302
Perimeter	570.7	673.3	830.4	802.2
Circularity	0.56	0.63	0.67	0.74
Solidity	0.81	0.88	0.92	0.95
Extent	0.56	0.64	0.69	0.72
G_avg	14.8	140.5	144.0	156.1
G_stddev	6.6	28.4	29.1	28.4
B_avg	16.7	188.0	195.4	206.7
B_stddev	8.5	32.6	32.9	29.5
R_avg	13.6	85.6	88.8	97.7
R_stddev	6.1	21.9	22.1	22.3
H_avg	30.6	15.7	15.0	15.6
H_stddev	31.5	1.4	1.5	1.8
S_avg	37.4	140.4	140.4	135.9
S_stddev	22.9	13.3	13.8	14.3
V_avg	16.8	188.0	195.4	206.7
V_stddev	8.7	32.6	32.9	29.5
Major_axis	192	208	257	262
Minor_axis	127	149	195	193
Inertia_X	16.2e+9	18.6e+9	43.9e+9	33.0e+9
Inertia_Y	16.2e+9	18.6e+9	43.9e+9	33.0e+9

Table 3.5: Features standard deviation by class.

Feature	Back/Top	Speckled	Holes/Broken	Other/Compliant
Box distance	29.67	32.51	22.08	14.15
Area	15,399	16,633	14,936	15,541
Perimeter	261.8	242.5	177.3	167.4
Circularity	0.20	0.15	0.12	0.10
Solidity	0.15	0.08	0.07	0.04
Extent	0.16	0.09	0.09	0.07
G_avg	5.3	23.3	19.3	22.5
G_stddev	7.8	7.3	5.7	5.6
B_avg	6.9	28.7	22.1	23.1
B_stddev	10.0	8.4	4.6	6.4
R_avg	4.2	16.9	13.2	16.3
R_stddev	5.8	5.6	4.5	4.4
H_avg	17.8	1.0	0.9	1.4
H_stddev	17.8	0.6	0.6	0.8
S_avg	18.5	8.5	6.6	9.1
S_stddev	9.6	3.2	3.0	3.8
V_avg	7.0	28.7	22.1	23.1
V_stddev	10.1	8.4	4.6	6.4
Major_axis	94	75	52	56
Minor_axis	73	54	46	45
Inertia_X	24.2e+9	22.8e+9	33.4e+9	33.7e+9
Inertia_Y	24.2e+9	22.8e+9	33.4e+9	33.7e+9

The t-tests on feature means have been performed for each class couple, and the results of such tests are shown in Figure 3.10, 22 features are tested and the null hypothesis has been accepted 21 times over the 132 tests.

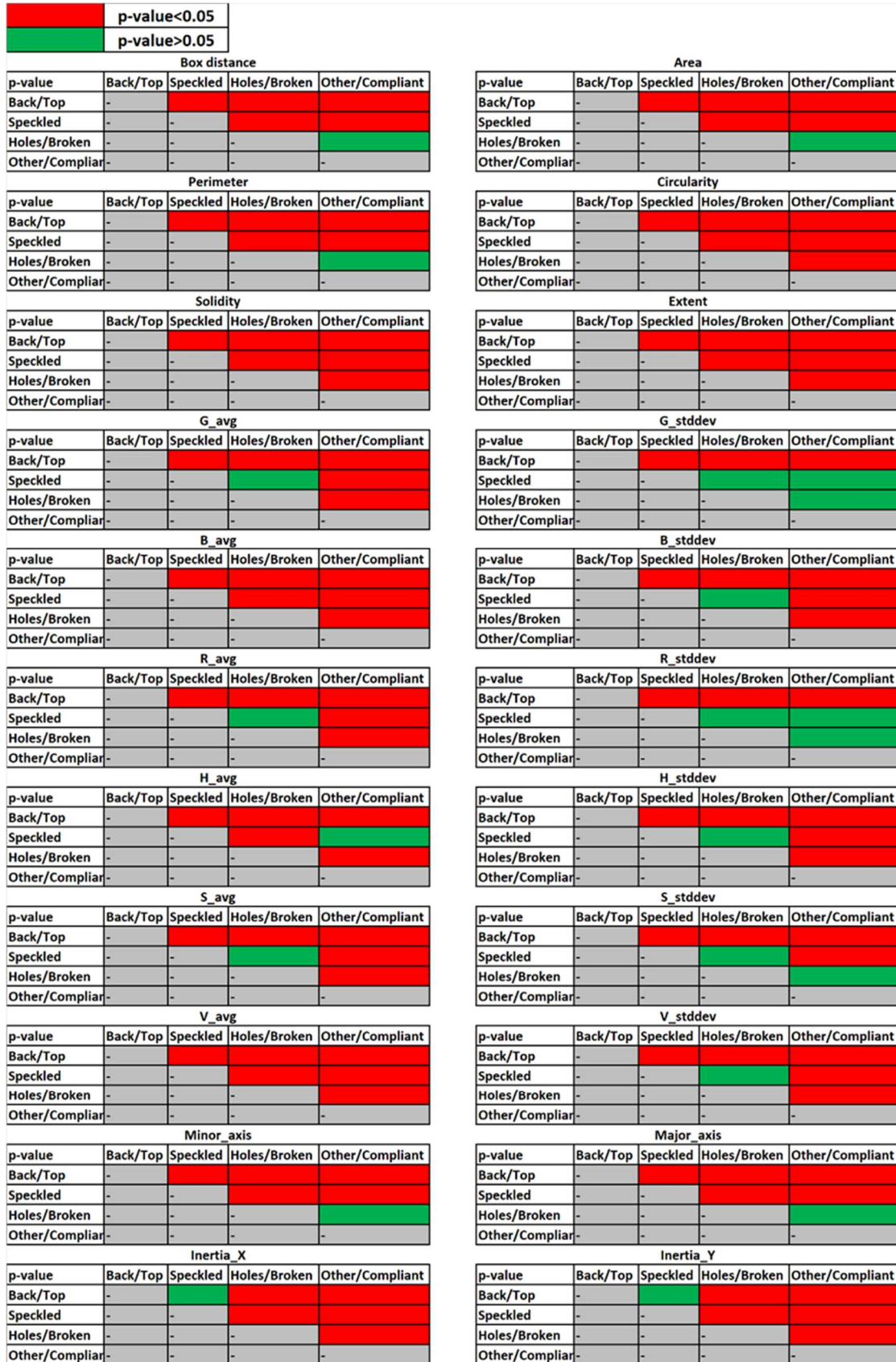


Figure 3.10: T-tests on features mean, alpha equal to 0.05.

Most of the features means are statistically different when the pairwise comparison is performed but only the differentiation is even higher whether the null hypothesis is rejected in t-test against all the classes; it happened for the classes for the following features: *Circularity*, *Solidity*, *Extent*, *B_avg* and *V_avg*; this result suggests that the features measuring the object compactness are able to distinguish the target class. All but two of the t-tests involving “Back/Top” class have been rejected, it suggests this class as the most different with respect to the others and almost all the features may be used to characterize the class, the RGB colour averages still the most characterizing ones due to their significant difference with respect to the other classes and their understandable meaning. The less valuable features are the ones related to the standard deviation in both the colourspaces; it works for all the classes but “Back/Top”, as already mentioned, its characteristics aren’t equal to the relative counterparts, moreover, 11 over the 21 tests resulting in an acceptance of the null hypothesis have been caused by t-tests involving the standard deviation features.

4 Discussion and Conclusion

This thesis faces a deep learning application case that could be found when trying to adopt machine learning technologies in an ongoing manufacturing process. The proposed method sees the combination of deep learning and classical machine learning techniques to localise and classify defective products in aubergine production lines. As DL model, Yolov8 has been selected as it represents the state of the art for object detection purposes. Among the selected classes to be identified within the acquired dataset, "Speckled" and "Holes/Broken" classes are underrepresented in the "compliant" dataset, they account approximately 7.9% and 1.3% respectively. Yolov8x managed to retrieve more than 90% of the objects belonging to "Back/Top" and "Other/Compliant" classes while it struggles to distinguish the remaining classes ("Speckled" and "Holes/Broken"), when those instances are misclassified, they are assumed to be "Other/Compliant" but the opposite doesn't happen, this performance fits the user's needs in case the interest points towards the retention of "good" samples and the minimization of the not compliant slices isn't a priority; on the localisation perspective, the model is extremely good on locating the targets when applied to "compliant" dataset, the drawn boxes enclose properly the objects while the completely missed targets are only those slices located on picture's borders, those products are barely visible. Yolov8x performances may be enhanced increasing the training pictures quantity, this operation is still time consuming and introduces inconsistencies when carried out by multiple operators; adding or changing the channels may introduce additional useful information, in fact, RGB imaging looks effective in recognizing "Back/Top" aubergine slices while specific wavelength bands

are proved to be in detecting rotten organic matter and fungal contaminations [26] [27] [28], the choice about the wavelengths selection is driven by the manufacturer needs, “Speckled” is one of those classes that is expected to show relevant contrast in some of the NIR wavelengths and is candidate to receive benefits to the addition of further wavelength channels; Yolov8x performances on “Holes/Broken” product is less dependent to the wavelengths/channels selection process until the contrast between the product and the background is guaranteed, in fact, “Holes/Broken” are recognized due to through their shape and size; wavelength selection methods have been successfully applied to aubergine’s pixels in order to retrieve the best wavelength set having high effectiveness in sorting the product under the following conditions [29]:

- Burnt.
- Cooked.
- Fresh product.
- Skin exposed to the camera.

The second part regarding the application of ML techniques sees the comparison of different algorithms to classify images at pixel level in effective parts and background; among the ML techniques tested in this work, Random Forest has been selected to be used in the main workflow due to its higher performances when compared to other ML techniques, moreover, such performances are retained when the number of features (colour channels) are decreased. Random Forest performances are explained by the similarity of the background pixels and reflects on the product, although “Skin” pixels are severely affected by a low Recall value (Figure 3.6), they account for less than 7% (in “compliant” images), as a result, its impact on the weighted performances is low. This problem may be tackled adopting two approaches, the first requires changing the process such a way that the product’s skin is distinguishable to the background, replacing the conveyor belt tape with another one having a different colour addresses the issue only if the new material doesn’t reflect the incident light, a possible solution is the replacement of the lighting system itself, backlighting or

diffusive lighting are possible ways to reduce the reflects. The second approach is based on wavelength selection, in fact, conveyor belt reflectivity may be significantly different to the organic matter's one when acquiring in specific wavelengths [27]. Random Forest provides a further significant result, it ranks colour channels by relative importance, based on this technique, the three most important are: "Cb", "v" and "Cr"; these colour representations aren't easily understandable from a human point of view, but they are the most useful when Random Forest splits the training instances.

This thesis introduces a variant to watershed transformation, it allows to separate overlapped objects even if the targets don't satisfy the implicit requirements put in place by the original watershed transformation paper (2.6). The already explained error sources interact with watershed transformation in "feature extraction" phase generating errors, most of them are partial BLOB extractions since the procedure isn't able to fix the gaps originated by the reflects in all the cases, the same happened for the unperfect product's circularity, in particular, sometimes the modified watershed transformation detects a "fake nuclei" close to sharp angles, those nuclei and all their associated pixels are considered autonomous BLOBs by the watershed transformation causing an area loss and shape deformation to the "real" BLOB representing that specific slice.

Feature values has been successfully extracted; therefore, mean and standard deviation have been calculated, the test results show as the features have good potential when trying to guess the classes. The errors generated in the previous phases impact on t-test reliability since the afore mentioned tests are partially fed with highly deformed BLOBs, as shown in Figure 3.1, a relevant fraction of "Speckled" and "Holes/Broken" are misclassified, hence, a significant quantity of feature values are allocated to the wrong class, therefore is reasonable to think that such errors affect the feature comparison. The workflow deployed in this thesis up to and excluded the feature extraction may be replaced by Yolov8x-seg, a Yolov8 variant based on instance

segmentation, it localises the borders drawing continuous segments instead of rectangles, hence the techniques used to refine the rectangles aren't needed, Yolo segmentation requires a slightly different process to be put in place, the labelling requires different support tools and a significantly larger amount of time which might be not justified since many applications require a roughly localisation.

The workflow proposed in this document (Figure 2.1) may be applied as set up procedure in industrial applications, although the limited effort put in labelling, the model is effective in locating the food products while the poor classification performances can be adjusted in two possible non-exclusive ways:

- Leveraging on operators' knowledge on the features; some features are easy to understand from a human perspective, thus the operators could establish a few thresholds on features' values with the purpose to assign the instance in the classes, this process operates on the workflow upstream fixing the incorrect class predictions that can be used to refeed a training process.
- A possible improvement is the addition of further image material to the already existing one, since a limiting resource is the labelling time, the existing model could be used to speed up this activity because the localisation is correct, and the operator is only responsible for selecting the class.

Future researches may investigate the usage of image channels different to the ones used in this thesis, starting from those wavelengths that showed good potential for food quality inspection, also for detecting quality criticalities different to what seen in in this case, RGB channels still useful to classify some objects (like "Back/Top" class), furthermore, the related sensors/filters are easy to access due to the industrial development experienced in RGB imaging.

Bibliography

- [1] J.-H. Cheng and D.-W. Sun, "Partial Least Squares Regression (PLSR) Applied to NIR and HSI Spectral Data Modeling to Predict Chemical Properties of Fish Muscle," *Food engineering reviews*, vol. 9, pp. 36-49, 2017.
- [2] D. Caballero, T. Antequera, A. Caro, M. L. Duran and T. Perez-Palacios, "Data Mining on MRI-Computational Texture Features to Predict Sensory Characteristics in Ham," *Food and Bioprocess Technology*, vol. 9, pp. 699-708, 2016.
- [3] W. Zhang, Y. Zhang, Z. Jia, Z. Dehai, X. Liang, Z. Liehan, L. Zhongwei and Y. Su, "Multi-source data fusion using deep learning for smart refrigerators," *Computers in Industry*, vol. 95, pp. 15-21, 2018.
- [4] A. D. Arya, S. S. Verma, P. Chakarabarti, T. Chakravarti, A. A. Elngar, M. Nami and A.-M. Kamali, "Asystematic review on machine learning and deep learning techniques in the effective diagnosis of Alzheimer's disease," *Brain Informatics*, vol. 10, no. 1, pp. 1-15, 2023.
- [5] J. Redmon, S. Divvala, R. Girshick and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," in *In Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016.
- [6] R. Girshick, J. Donahue, T. Darrell and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014.
- [7] J. Redmon and A. Farhadi, "YOLOv3: An Incremental Improvement," *arXiv preprint arXiv*, no. 1804.02767, 2018.
- [8] A. Géron, "Hands-On machine learning with scikit-learn, keras & tensorflow," O'Reilly, 2019, pp. 114-124;149-151;177-184;199-200;240-249;256-259.

- [9] T. Dewi, P. Risma and Y. Oktarina, "Fruit sorting robot based on color and size for an agricultural product packaging system," *Bulletin of Electrical Engineering and Informatic*, vol. 9, no. 4, pp. 1438-1445, 2020.
- [10] T. Dewi, Z. Mulya, P. Risma and Y. Oktarina, "BLOB analysis of an automatic vision guided system for a fruit picking and placing robot," *Int. J. Computational Vision and Robotics*, vol. 11, no. 3, pp. 315-327, 2021.
- [11] W. Chen, Z. Li, J. Zhang and B. Yi, "A Study of Workpiece Targeting Based on Improved Blob Analysis," in *In 2020 5th International Conference on Automation, Control and Robotics Engineering (CACRE)*, 2020.
- [12] Y. Ekawaty, Indrabayu and I. S. Areni, "Automatic Cacao Pod Detection Under Outdoor Condition Using Computer Vision," in *2019 4th International Conference on Information Technology, Information Systems and Electrical Engineering (ICITISEE)*, 2019.
- [13] H. Zhou, L. Llewellyn, L. Wei, D. Creighton and S. Nahavandi, "Marine object detection using background modelling and blob analysis," in *2015 IEEE International Conference on Systems, Man, and Cybernetics*, Hong Kong, 2023.
- [14] S. Yang, B. Ni, W. Du and T. Yu, "Research on an Improved Segmentation Recognition Algorithm.," *Sensors*, vol. 22, no. 10, p. 3946, 2022.
- [15] Y. Cui and N. Zhou, "Blob Analysis Using Watershed Transformation," in *International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems*, Berlin, 2000.
- [16] A. Wang, W. Zhang and X. Wei, "A review on weed detection using ground-based machine vision and image processing techniques," *Computers and electronics in agriculture*, no. 158, pp. 226-240, 2019.
- [17] S. M. Pelkey, J. F. Mustard, S. Murchie, R. T. Clancy, M. Wolff, M. Smith, R. Milliken, J.-P. Bibring, A. Gendrin, F. Polulet, Y. Langevin and B. Gondet, "CRISM multispectral summary products:Parameterizing mineral diversity on Mars from reflectance," *Journal of Geophysical Research: Planets*, vol. 112, no. E8, 2007.
- [18] İ. ÇINAR, M. KOKLU and Ş. TAŞDEMİR, "Classification of Raisin Grains Using Machine Vision and Artificial," *Gazi Mühendislik Bilimleri Dergisi*, vol. 6, no. 3, pp. 200-209, 2020.

- [19] N. Oppelt and W. Mouser, "Hyperspectral monitoring of physiological parameters of wheat during a vegetation period using AVIS data.," *International Journal of Remote Sensing*, vol. 25, no. 1, pp. 145-159, 2004.
- [20] A. A. Khule, M. S. Nagmode and R. D. Komati, "Automated Object Counting for Visual Inspection," in *2015 International Conference on Information Processing (ICIP)*, 2015.
- [21] G. Feng and C. Quixin, "Study on color image processing based intelligent fruit sorting system.," in *Fifth world congress on intelligent control and automation*, Hangzhou, 2004.
- [22] R. Azadina, S. Fouladi and A. Jahanbakhshi, "Intelligent detection and waste control of hawthorn fruit based on ripening level using machine vision system and deep learning techniques.," *Results in Engineering*, vol. 17, p. 100891, 2023.
- [23] X. Xue, Z. Guomin, Q. Yun, L. Zhuang, W. Jian, H. Lin, F. Jingchao and G. Xiuming, "Detection of young green apples in orchard environment using adaptive ratio chromatic aberration and HOG-SVM," in *Computer and Computing Technologies in Agriculture XI*, Jilin, 2017.
- [24] D. P. Penumuru, S. Muthuswamy and P. Karumbu, "Identification and classification of materials using machine vision and machine learning in the context of industry 4.0.," *Journal of Intelligent Manufacturing*, vol. 31, no. 5, pp. 1229-1241, 2020.
- [25] C. S. Lauguico, S. R. Conception, D. J. Alejandrino, R. R. Tobias, D. D. Macasaet and P. E. Dadios, "A comparative analysis of machine learning algorithms modeled from machine vision-based lettuce growth stage classification in smart aquaponics.," *Int. J. Environ. Sci. Dev*, vol. 11, no. 9, pp. 442-449, 2020.
- [26] J. Gòmez-Sanchis, G. Camps-Valls, E. Moltò, L. Gòmez-Chova, N. Aleixos and J. Blasco, "Segmentation of hyperspectral images for the detection of rotten mandarins.," in *Analysis and Recognition: 5th International Conference*, Póvoa de Varzim, 2008.
- [27] E. Minieri and E. Milani, "POLITesi," 4 Maggio 2023. [Online]. Available: <https://hdl.handle.net/10589/212282>. [Accessed 7 Novembre 2023].
- [28] J. Sun, R. Künnemeyer, A. McGlone and N. Tomer, "Optical properties of healthy and rotten onion flesh from 700 to 1000 nm," *Postharvest biology and technology*, vol. 140, no. 1, pp. 1-10, 2018.

- [29] M. Di Raimondo, "POLITesi," 7 Ottobre 2021. [Online]. Available: <https://hdl.handle.net/10589/179617>. [Accessed 7 Novembre 2023].

List of Figures

Figure 1.1:Original Yolo network.....	6
Figure 1.2: Yolo's output structure.....	7
Figure 1.3: Intersection over union (IoU).....	8
Figure 1.4: Yolo result postprocessing and output representation.	9
Figure 1.5: Yolo releases comparison, average precision vs number of parameters(https://github.com/ultralytics/ultralytics).	10
Figure 1.6: Decision Tree structure.	12
Figure 1.7: DBscan working principle [8].	16
Figure 1.8: Documents indexed by Scopus on "deep learning" AND "object detection" entries over time.	17
Figure 1.9: Object detection based on background filtering [13].....	18
Figure 1.10: Erosion on rice grains image.	19
Figure 1.11: Isolated BLOB/ aubergine slice.	20
Figure 1.12: Classic BLOB analysis pathway.....	21
Figure 2.1: Thesis workflow.....	23
Figure 2.2: Camera and lightning system setup.	24
Figure 2.3: Camera system location in COTRAPA.....	25
Figure 2.4: "Compliant" image sample.	26
Figure 2.5: "Discard" image sample.	26
Figure 2.6: Class samples: "1" is "Back/Top", "2" is "Speckled", "3" is "Broken/Holes", "4" is "Other/Compliant".....	27
Figure 2.7: <i>LabelImg</i> interface.	28
Figure 2.8: Annotation file in "Yolo" standard.	29
Figure 2.9: Class defining file in "Yolo" standard.	29
Figure 2.10: Pixel level mask drawn through <i>Photoshop.exe</i>	34
Figure 2.11: ML for pixel sorting workflow.	34
Figure 2.12: One-vs-All Support Vector Machine (SVM).	36

Figure 2.13: Random Forest applied to Yolov8x.....	41
Figure 2.14: Distance map on aubergines slice.	42
Figure 2.15: T-test for feature comparison.....	45
Figure 3.1: Yolov8x confusion matrix, values as percentage of the true instances for that class.....	47
Figure 3.2: SGD confusion matrix and performances.	49
Figure 3.3: Softmax Regression confusion matrix and performances.	49
Figure 3.4: Decision Tree confusion matrix and performances.	50
Figure 3.5: Random Forest confusion matrix and performances, 12 features run.....	50
Figure 3.6: Random Forest confusion matrix and performances, 5 features run.....	51
Figure 3.7: DBscan+KNN cluster purity.....	52
Figure 3.8: Silhouette score vs #clusters.....	53
Figure 3.9: K-Means cluster purity.	53
Figure 3.10: T-tests on features mean, alpha equal to 0.05.....	56

List of Tables

Table 2.1: Yolov8x training hyperparameters.	31
Table 2.2: Yolov8x dataset composition.	32
Table 2.3: Yolov8x prediction hyperparameters.	33
Table 2.4: Colour channels description.	35
Table 2.5: SDG parameters.	36
Table 2.6: Softmax parameters.	37
Table 2.7: Decision Tree parameters.	38
Table 2.8: Random Forest parameters.	39
Table 2.9: DBscan parameters.	39
Table 2.10: KNN parameters.	40
Table 2.11: K-Means parameters.	40
Table 2.12: Features list.	44
Table 3.1: Yolov8x predictions on "compliant" dataset vs ground truth.	48
Table 3.2: Features ranked by Random Forest classifier.	51
Table 3.3: Yolov8x predictions excluding elements close to the border.	54
Table 3.4: Features mean by class.	54
Table 3.5: Features standard deviation by class.	55

Acknowledgments

Thanks to the staff of the Politecnico di Milano for their professionalism. Thanks to Prof. Marco Tarabini and Ing. Chiara Conese for the assistance provided to produce this document.

