UNIVERSITÀ
DEGLI STUDI DI BARI
ALDO MORO

Anomaly Detection in the Agroalimentary Supply Chain

Docente: Prof. Michelangelo Ceci

Revisore: Dott. Stefano Polimena

Studente: Giovanni Federico Poli

Matricola: 766577

Index:

The Problem

The motivation for research in non-destructive contactless quality control within the agroalimentary supply chain stems from the critical need to address the perishability of fresh fruits and vegetables.

These products are highly susceptible to various forms of deterioration such as discoloration, dryness, and texture loss, which significantly impact their quality and shelf life.

Traditional methods of quality assessment rely heavily on **manual inspection**, which is not only time-consuming but also prone to inconsistencies and subjectivity.
Therefore, there is a pressing need to develop automated, contactless inspection techniques that can efficiently evaluate the quality of products without damaging them.

By integrating such technology into the supply chain, effective quality control can be ensured from harvest to supermarket shelves, ultimately minimizing waste and enhancing consumer satisfaction.

The research goal aims to develop innovative machine learning approaches to enhance non-destructive, contactless quality control within the agroalimentary supply chain.

By leveraging machine learning algorithms, this research seeks to enhance the accuracy and efficiency of quality control measures, ensuring that only the highest-quality products are delivered to consumers.

Proposed approach

The proposed approach involves the implementation of two unsupervised techniques, namely an autoencoder and a self-organizing map (SOM), with the objective of comparative analysis within the given context.

- The autoencoder, a neural network architecture, focuses on learning efficient representations of input data through an encoding-decoding process.
- The self-organizing map, a type of artificial neural network, specializes in creating low-dimensional representations of high-dimensional data, facilitating tasks such as clustering and visualization.

By employing these methods, the study seeks to evaluate their individual capabilities and contributions, focusing specifically on their effectiveness in anomaly detection within a set of arugula (rucola) images.

The Data

The dataset utilized in this study originates from the repository provided by dott. Polimena and comprises approximately 1200 RGB (.tif) images of arugula, each featuring a black background. The dataset consists of normal and potentially anomalous images.
These images serve as the primary source for training, testing, and validating the unsupervised techniques employed for anomaly detection. In particular 32 of these images were separated in order to make the final test set.

The consistent background ensures uniformity across the dataset, facilitating accurate analysis and comparison of the methods' performance.

For this reason, the data quality is considered very high, as some data preparation has already been conducted, ensuring that the images are standardized and free from significant noise or inconsistencies.

Leveraging this comprehensive and well-prepared dataset, the study aims to develop robust models capable of effectively detecting anomalies in arugula images, thereby contributing to improved quality control practices within the agroalimentary supply chain.

[Data samples]

The Models

Models' set-up:

In the initial attempt to process the images, we opted to use Google Colab.
However, we found this approach limiting as it consistently exceeded the video RAM limit on the GPU, resulting in frequent crashes.

Consequently, we decided to create an environment on a private machine following the guide provided by TensorFlow, which led to the configuration of NVIDIA tools such as CUDA and cuDNN with versions 11.2 and 8100 respectively, along with the setup of a conda environment with a Python 3.9 interpreter.
In this environment, TensorFlow was installed and used to model the first version of the autoencoder.

However, this solution also proved inadequate due to the VRAM memory issue (6 GB), resulting in Out-Of-Memory errors when loading and processing natural-resolution images.
Given the sensitivity of preserving all information within the images, natural-resolution image processing posed a sensible constraint. With the aim of maintaining data integrity, we decided to explore an alternative approach.

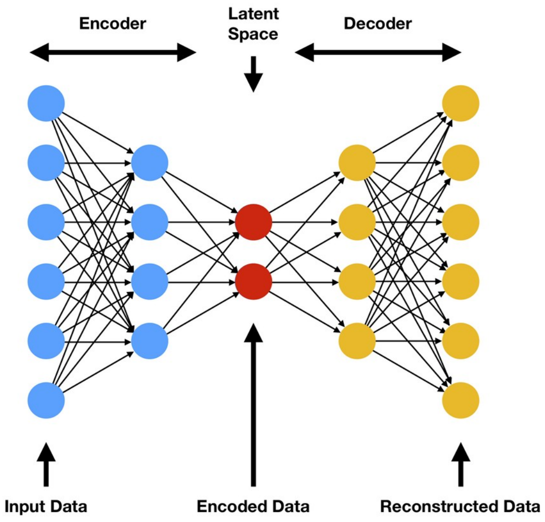
Therefore, we opted to utilize a university server with greater resources, where the same development environment was replicated. In this context, we preferred using PyTorch forgreater control over memory elements, aiming to effectively address the challenges associated with available computational resources.

The Autoencoder:

Autoencoder - Introduction:

Anomaly detection aims to identify patterns in data that deviate from normal behavior. Autoencoders, a type of neural network architecture, can be adapted for unsupervised anomaly detection by reconstructing input data and comparing it with the original data.
Differences between original and reconstructed images indicate the anomalies.

The idea is: by leveraging the autoencoder, we aim to build a model that can automatically learn representations of normal data and detect differences indicative of anomalies.



[Autoencoder visualization]

Autoencoder - Methodology:

Images are loaded using OpenCV which also handles the normalization and then converted to PyTorch tensors.

The autoencoder consists of an encoder and a decoder component.

The encoder compresses the input data into a latent representation, while the decoder reconstructs the original input data from this representation.

We found out that the best **number of filters** for the convolutional layers was 32, 64 and 128.

Autoencoder – Architecture

1. **Encoder**:
    - First Convolutional Layer:
        - output = $\text{ReLU}(\text{Conv2d}(\text{input}, \text{weight\_1}, \text{stride}=1, \text{padding}=1))$
        - maxpooled_output = $\text{MaxPool2d}(\text{output}, \text{kernel\_size}=2, \text{stride}=2)$
    - Second Convolutional Layer:
        - output = $\text{ReLU}(\text{Conv2d}(\text{maxpooled\_output}, \text{weight\_2}, \text{stride}=1, \text{padding}=1))$
        - maxpooled_output = $\text{MaxPool2d}(\text{output}, \text{kernel\_size}=2, \text{stride}=2)$
    - Third Convolutional Layer:
        - output = $\text{ReLU}(\text{Conv2d}(\text{maxpooled\_output}, \text{weight\_3}, \text{stride}=1, \text{padding}=1))$
        - maxpooled_output = $\text{MaxPool2d}(\text{output}, \text{kernel\_size}=2, \text{stride}=2)$

2. **Decoder**:
    - First Convolutional Transpose Layer:
        - output = $\text{ReLU}(\text{ConvTranspose2d}(\text{input}, \text{weight\_4}, \text{stride}=2, \text{padding}=1, \text{output\_padding}=1))$
    - Second Convolutional Transpose Layer:
        - output = $\text{ReLU}(\text{ConvTranspose2d}(\text{output}, \text{weight\_5}, \text{stride}=2, \text{padding}=1, \text{output\_padding}=1))$
    - Third Convolutional Transpose Layer:
        - output = $\text{ConvTranspose2d}(\text{output}, \text{weight\_6}, \text{stride}=2, \text{padding}=1, \text{output\_padding}=1)$

Autoencoder - Parameters:

Several challenges were encountered during the hyperparameters tuning phase:

- The Adam **optimizer** is chosen for gradient descent optimization due to its effectiveness in training deep neural networks and it is known for being relatively robust to the choice of hyperparameters, requiring minimal tuning compared to other optimizers.
- **Learning rate**: 0.0001
  We also encountered some problems in finding the best learning rate since we thought that 0.001 would be sufficient, but it became stuck on a local minimum.
- 30 **epochs** on a **batch size** of 16 images at full resolution.

Autoencoder - Data:

In order to effectively process data within the PyTorch Autoencoder implementation, it's essential to work with tensors. Therefore, the initial step in data manipulation process involves converting the numpy image into a PyTorch tensor.

Subsequently, the normalization process is applied by dividing the pixel values by 255.0. This step is crucial as it ensures that the pixel values fall within the range of 0 to 1.

Autoencoder - Training, Testing and Evaluation:

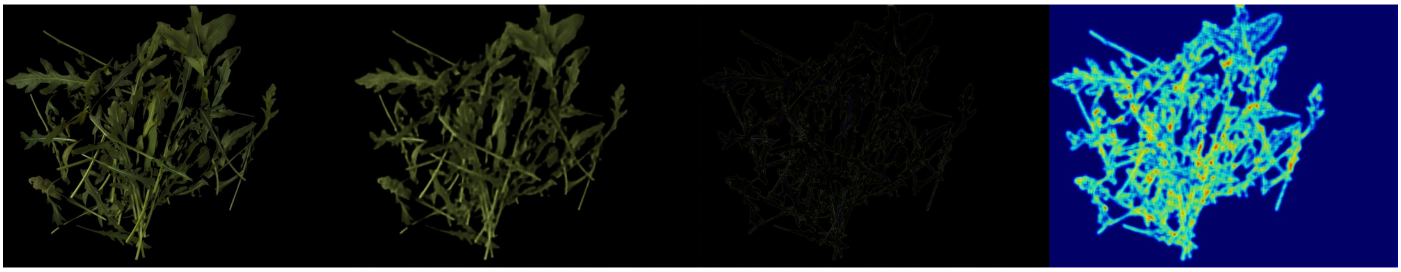
The Mean Squared Error (**MSE**) loss function is utilized to measure the reconstruction error. The MSE loss is commonly used in reconstruction tasks, where the objective is to minimize the difference between the input and reconstructed data.

Following 30 epochs of training, the loss on the training and validation sets is measured at 0.0005 and 0.0004, respectively, achieved over a training duration of approximately two hours.

Post-training, the model undergoes evaluation using an independent test dataset to gauge its capacity for accurately reconstructing normal data.

During this evaluation, the model assesses each reconstructed image from the test set. Any image displaying a reconstruction loss surpassing 0.0005 is identified as anomalous.
This threshold isn't arbitrary; rather, it is based on an average loss of 0.0004.
Thus, 0.0005 is deemed an appropriate threshold value.

Upon detection of an anomaly, a heatmap is generated, depicting the pixel-by-pixel difference between the input and reconstructed images.

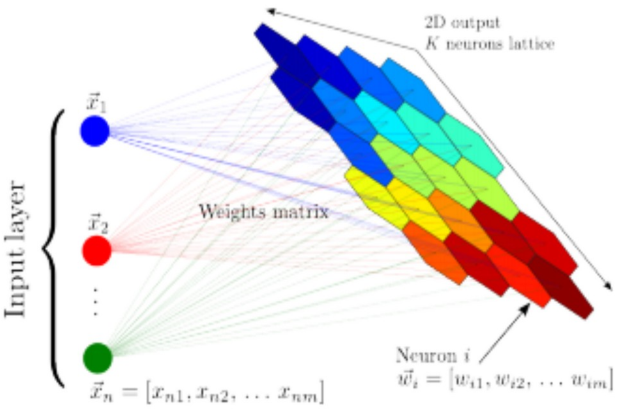


[Autoencoder final output: input, reconstruction, difference and heatmap]

Self-Organizing Map:

Self-Organizing Map - Introduction:

Self-Organizing Maps (SOM) are neural network models used for unsupervised learning. SOMs differ from other artificial neural networks because they apply competitive learning as opposed to error correlated learning, which involves back-propagation and gradient descent. In competitive learning, nodes compete for the right to respond to the input data subset.


2D output
K neurons lattice
Input layer
$\vec{x}_1$
$\vec{x}_2$
Weights matrix
Neuron i
$\vec{x}_n = [x_{n1}, x_{n2}, \dots x_{nm}]$
$\vec{w}_i = [w_{i1}, w_{i2}, \dots w_{im}]$

[SOM visualization]

In this implementation, SOM is utilized for color quantization, a process of reducing the number of colors in an image while preserving its visual appearance. The trained SOM provides a representation of the colors present in the input images. The idea is:

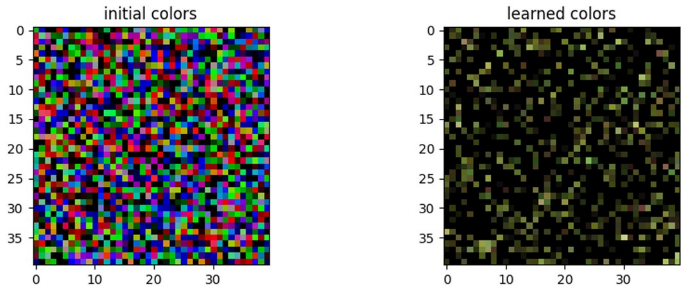
1. Initialize the weights for each node. The weights are set to random values.
2. Choose a vector at random from the training set.
3. Examine every node to calculate which one's weight is most like the input vector.
   This will allow us to obtain the Best Matching Unit (BMU).
   We compute the BMU by iterating over all the nodes and calculating the Euclidean distance between each node's weight and the current input vector.
   The node with a weight vector closest to the input vector is marked as the BMU.
4. Calculate the radius of the neighborhood of the BMU.
   Nodes found within the radius are deemed to be inside the neighborhood of the BMU.
5. Weights of the nodes found in the previous step are adjusted to make them more like the input vector. The weights of the nodes closer to the BMU are adjusted more.
6. Repeat from step 2 for N iterations.

$$d_j(\mathbf{x}) = \sum_{i=1}^{D} (x_i - w_{ji})^2$$

[the squared Euclidean distance between the input vector x and the weight vector wj for each neuron j]

Self-Organizing Map - Methodology:

The SOM is initialized with a grid size of **40x40**, reflecting the desired output color space. Each neuron in the SOM grid represents a color cluster. The SOM is trained on the input images, where the pixels are normalized before training.



[SOM color matrix]

Training involves iteratively updating the SOM's weights to adjust to the input data distribution. The training process utilizes a **neighborhood function** ('bubble' in this case) to adjust the influence of neighboring neurons during weight updates.

Self-Organizing Map - Parameters:

The SOM is initialized with a **sigma value** of 1.0 and a **learning rate** of 0.2. These parameters control the neighborhood size (radius) and the rate at which the SOM adapts to the input data, respectively.

Self-Organizing Map - Data

This SOM implementation follows the JustGlowing's MiniSom implementation that streamlines data processing, requiring minimal manipulation.

It is designed to accept images in a standardized format, emphasizing correct shape and normalization.
In particular since the images are 3D matrices we convert them into a 2D array by multiplying their width and length. Furthermore, by ensuring pixel values fall within the 0 to 1 range, the normalization grants consistency across the images.

Self-Organizing Map - Training, Testing and Evaluation:

The SOM undergoes incremental training on each input image.
The training process involves updating the SOM's weights to better represent the color distribution of the input image.
The number of training iterations per image is set to 100,000, which takes about six hours.

The **Quantization Error** is utilized to measure the reconstruction error.
It indicates the degree of discrepancy between the original colors in the input images and the quantized colors represented by the SOM neurons. Despite its primary focus on color quantization, the SOM's quantization error indirectly influences the quality of image reconstructions and aids in assessing the model's performance in anomaly detection based on color discrepancies.

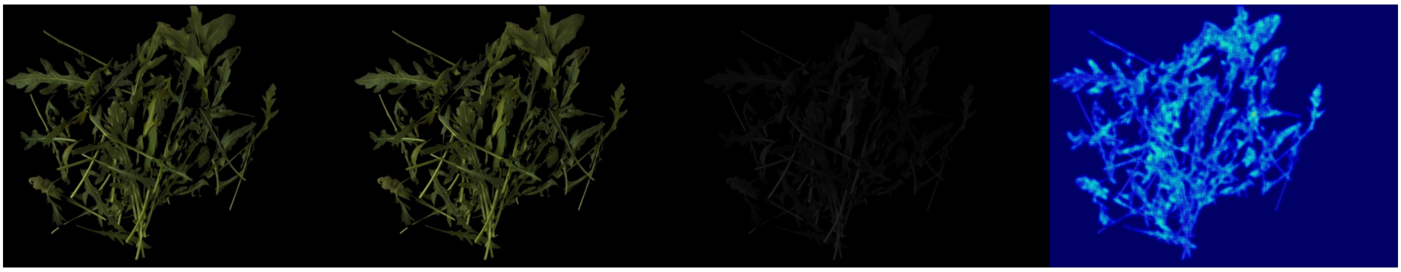
In the testing phase, each image in the test set undergoes a three-step process for reconstruction:

- Initially, the image is quantized, meaning it is mapped into a lower-dimensional space, which reduces the complexity of color representation.
- Subsequently, the quantized values are clustered, associating them with specific neurons within the SOM.
  These SOM neurons represent centroids in the learned color space.
- Then the reconstruction is performed by mapping the quantized vectors obtained from the SOM back to their original pixel locations in the image.

Upon reconstruction, the input image is transformed into a reconstructed image based on the assigned SOM neurons. While the reconstructed image maintains the original shape and structure of the input, it differs in color due to the quantization process.

This color variation, though almost imperceptible to the human eye, is revealed through subtraction between the reconstructed and input images.
Consequently, the resulting image illustrates discrepancies in color between the input and reconstruction.

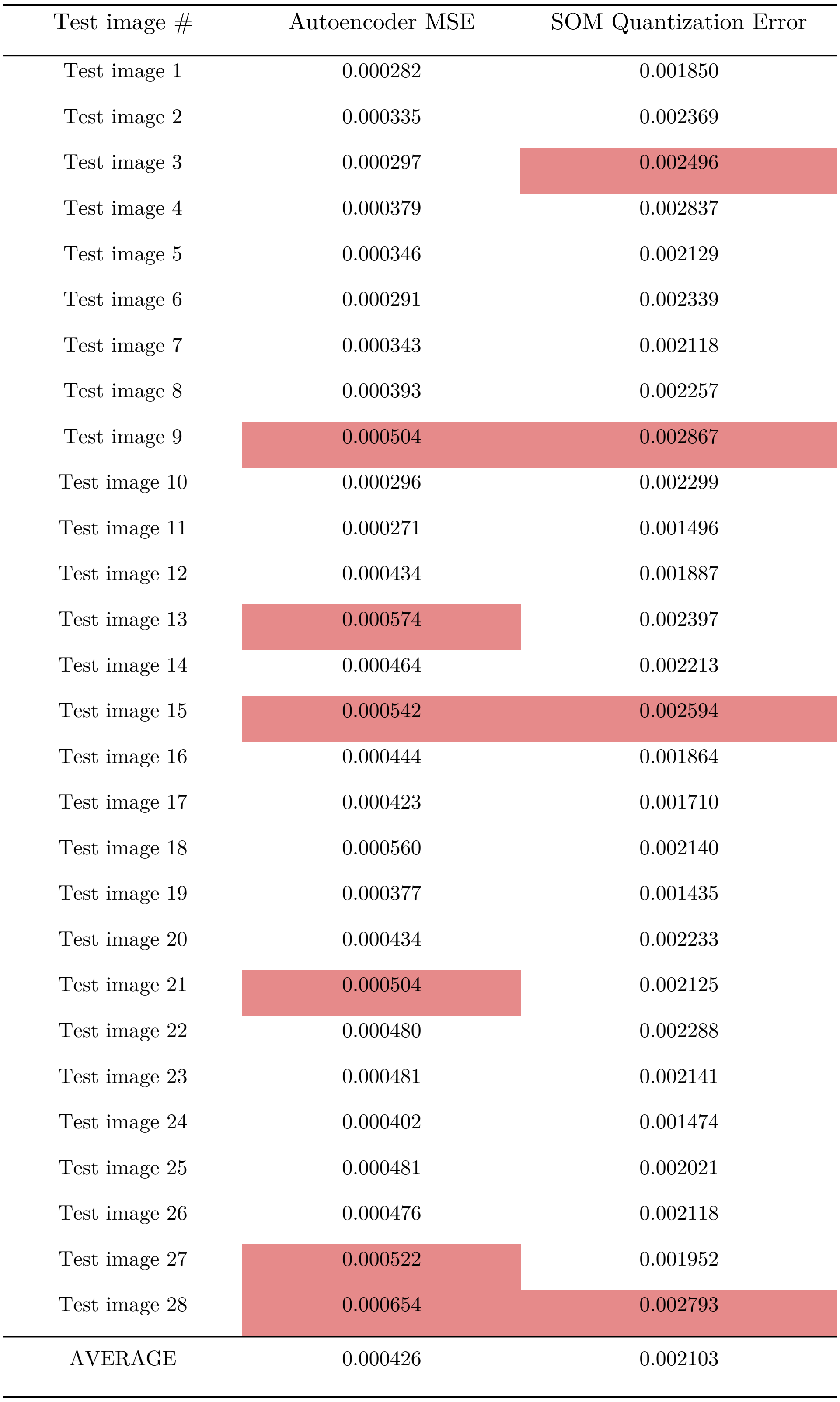
To emphasize these subtle color differences, the heatmap is applied to the gray-scaled subtracted image.
This heatmap visualization focuses on luminance variations, enabling the identification of color gaps between the input and reconstructed images.



[SOM final output: input, reconstruction, difference and heatmap]

Test resuts

Highlighted in red the images that carry an anomaly according to the specific model.



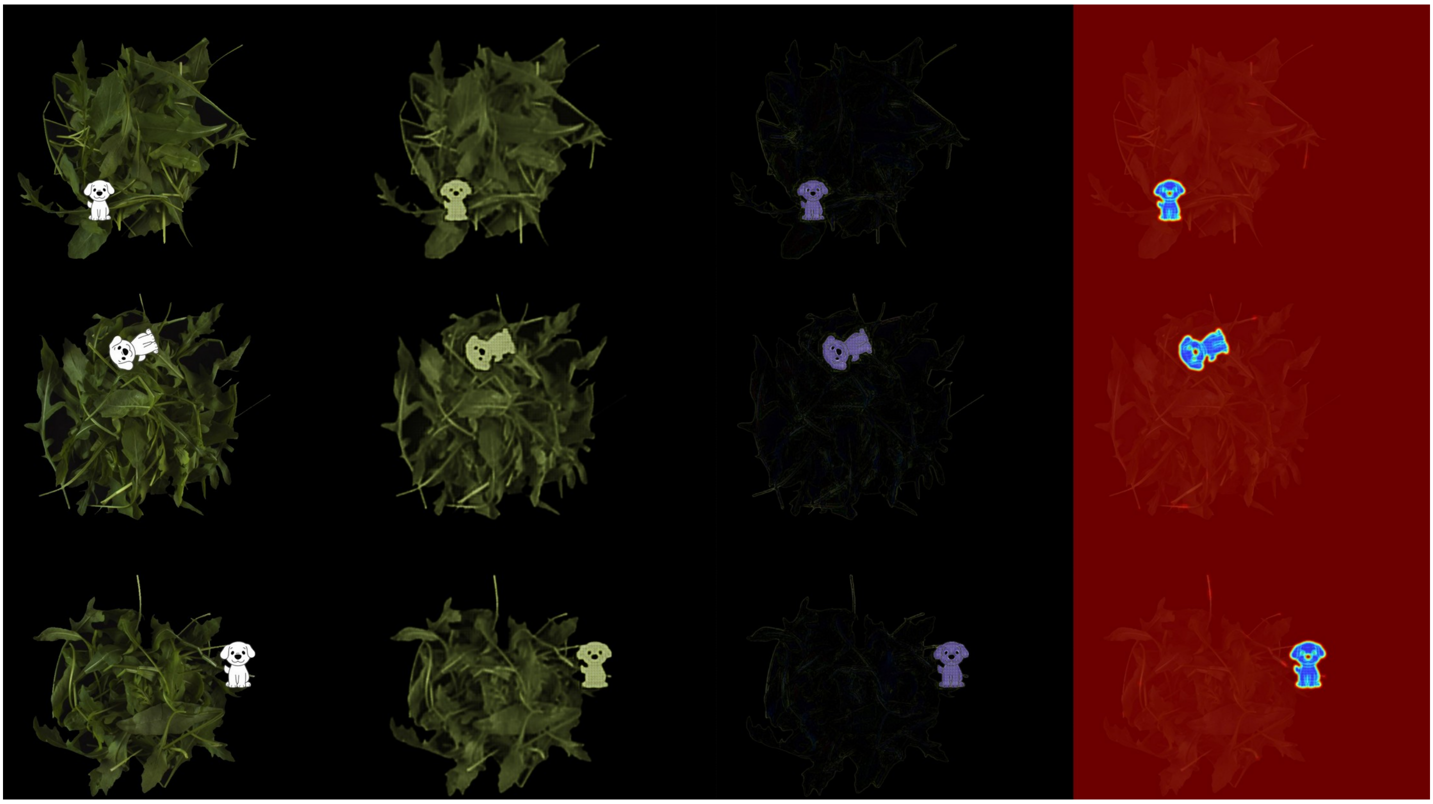
Test image #	Autoencoder MSE	SOM Quantization Error
Test image 1	0.000282	0.001850
Test image 2	0.000335	0.002369
Test image 3	0.000297	0.002496
Test image 4	0.000379	0.002837
Test image 5	0.000346	0.002129
Test image 6	0.000291	0.002339
Test image 7	0.000343	0.002118
Test image 8	0.000393	0.002257
Test image 9	0.000504	0.002867
Test image 10	0.000296	0.002299
Test image 11	0.000271	0.001496
Test image 12	0.000434	0.001887
Test image 13	0.000574	0.002397
Test image 14	0.000464	0.002213
Test image 15	0.000542	0.002594
Test image 16	0.000444	0.001864
Test image 17	0.000423	0.001710
Test image 18	0.000560	0.002140
Test image 19	0.000377	0.001435
Test image 20	0.000434	0.002233
Test image 21	0.000504	0.002125
Test image 22	0.000480	0.002288
Test image 23	0.000481	0.002141
Test image 24	0.000402	0.001474
Test image 25	0.000481	0.002021
Test image 26	0.000476	0.002118
Test image 27	0.000522	0.001952
Test image 28	0.000654	0.002793
AVERAGE	0.000426	0.002103

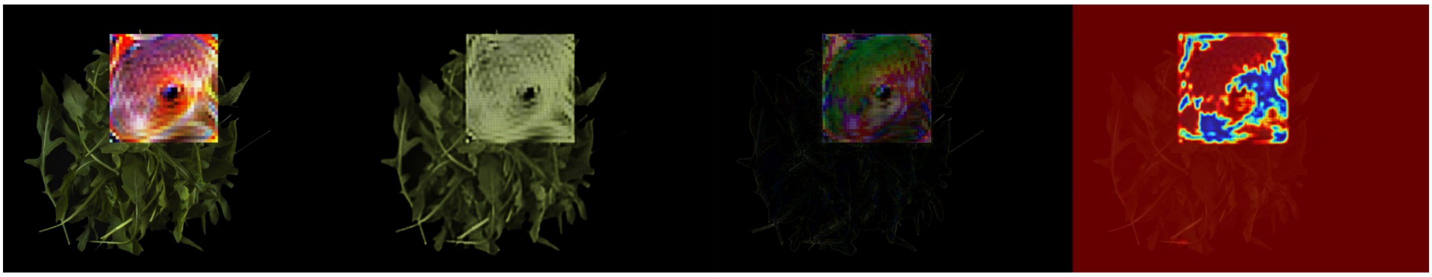
This is not a direct comparison, it is just to show how different models can find different anomalies in the same testing set.

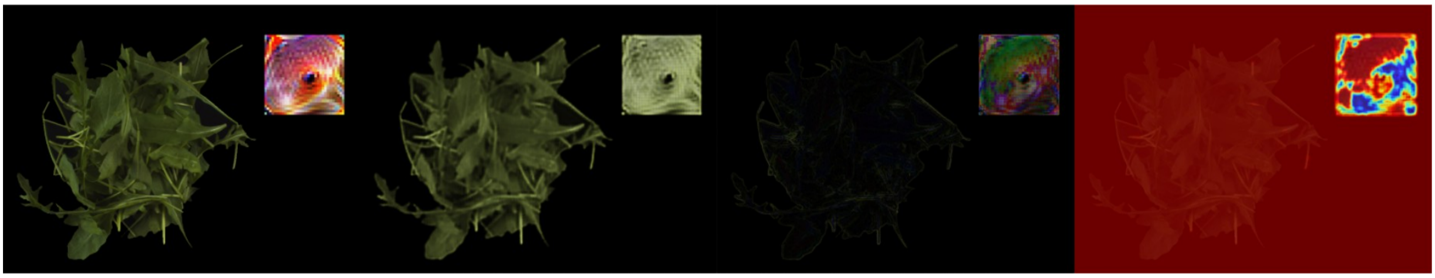
Final Experiments:

To put both methodologies to the test, a small separate test set was created.
This set contained images of arugula with added noise, achieved by overlaying other images or positioning them beside the arugula.
The results obtained from both methodologies are as follows:

Autoencoder:

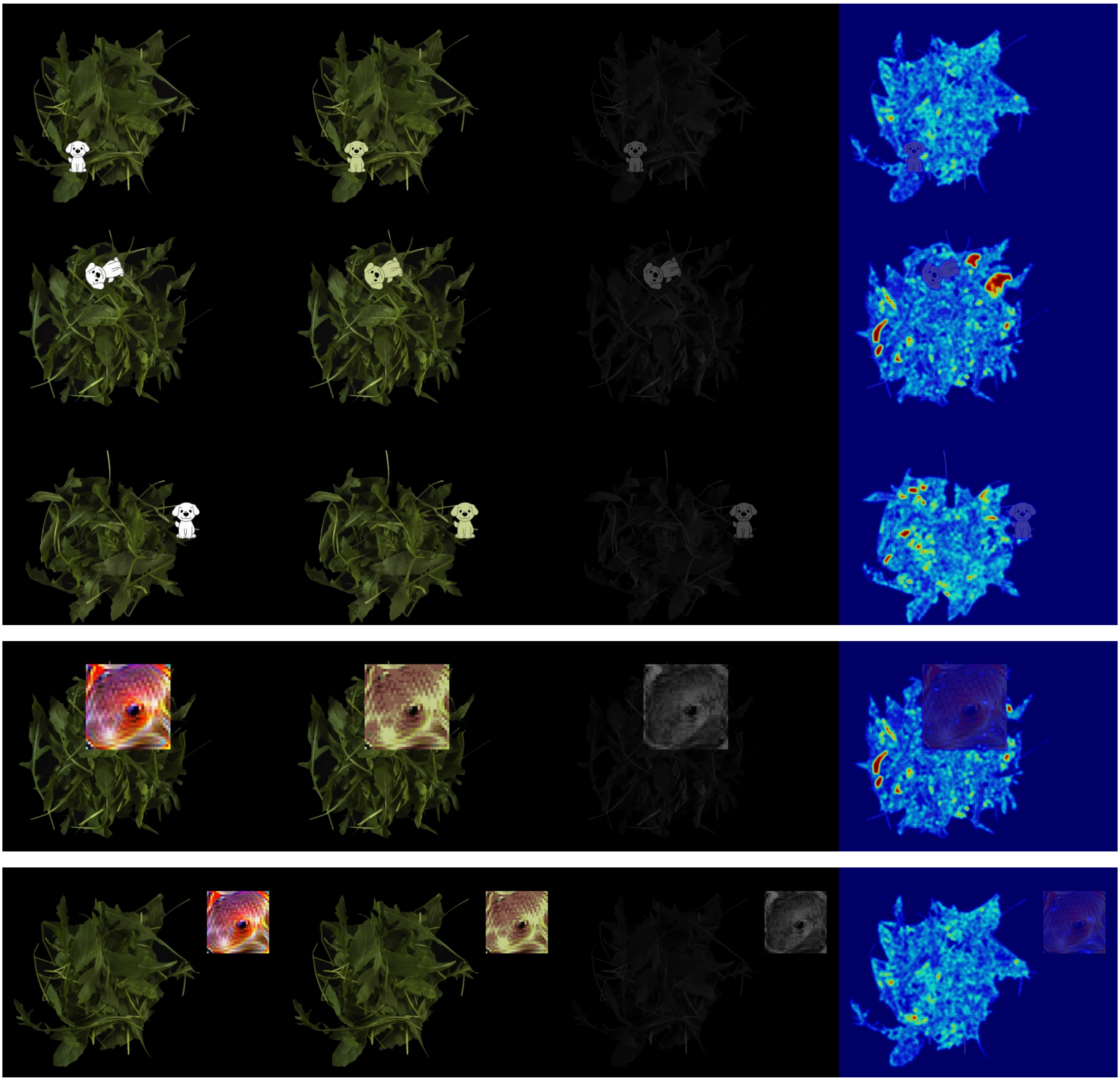






The autoencoder precisely captures the anomaly, as seen from the heatmap and the higher loss value (~ 0.0022) compared to the average on the test set (0.0004) and a threshold of 0.0005.

Self-Organizing Map:



The SOM doesn't reveal the anomaly through the heatmap because it evidently manages to reproduce a similar luminance (likely associated with element 35,30 in the learned matrix for the case of the dog image).

Nevertheless, it signals the anomaly through the quantization error significantly exceeding the threshold by about three times (in the “dog” case ~ 0.0073 versus the threshold of 0.0024 and an average on the test of 0.0021).

Experiment Results

Highlighted in red the images that carry an anomaly according to the specific model. Threshold set at 0.0005 for the autoencoder and 0.0024 for the SOM.

Experiment image #	Autoencoder MSE	SOM Quantization Error
Dog image 1	0.001815	0.006245
Dog image 2	0.003213	0.007151
Dog image 3	0.002245	0.007308
Abstract image 1	0.005289	0.029536
Abstract image 2	0.001364	0.072122

Conclusion:

In the pursuit of enhancing non-destructive, contactless quality control within the agroalimentary supply chain, the comparison between the Autoencoder and Self-Organizing Map (SOM) reveals complementary strengths and contributions towards anomaly detection.

The Autoencoder, with its focus on structural fidelity through encoding-decoding processes, excels in capturing intricate patterns and deviations within input data. By reconstructing images and evaluating the mean squared error, it efficiently identifies anomalies indicative of quality discrepancies.

On the other hand, the SOM prioritizes color quantization, offering a different perspective on anomaly detection. Through the training process, it learns a low-dimensional representation of color space, enabling efficient clustering and visualization. Despite its primary focus on color, the SOM's quantization error indirectly influences the quality of reconstructions, aiding in anomaly detection based on color discrepancies.

The integration of these techniques within the agroalimentary supply chain presents a comprehensive solution for quality control. While the Autoencoder specializes in identifying structural anomalies, the SOM complements this by detecting color-based irregularities. By leveraging the strengths of both approaches, a holistic quality control system can be established, ensuring the delivery of only the highest-quality products to consumers.

In essence, rather than viewing the Autoencoder and SOM as competing methodologies, we should embrace them as complementary tools that, when used in conjunction, offer a holistic and robust solution for non-destructive quality control within the agroalimentary supply chain.