VLAIO-TETRA
Machine Vision for Quality Control
Case 6 - Detection of a sufficient presence of nuts and sauce on Cornettos

Contents

1 Introduction **2**

2 Literature Study **4**
2.1 Classification 4
2.2 Segmentation 5

3 Methodology **10**
3.1 Data Capturing 11
3.2 Data Pre-processing 13
3.3 Model Choice 16
3.4 Model Training 19
3.5 Model Evaluation 20
3.6 K-means Segmentation 21

4 Results **22**
4.1 Binary Classification 22
4.2 Multi-label Classification 31
4.3 K-Means Segmentation 36

5 Conclusion **42**

1 Introduction

Due to recent technological advancements and the emergence of more efficient production methods, both semi-finished and finished products are being produced on a larger scale. Depending on the production speed, it can be challenging for operators to monitor the quality of the products constantly.

To address this issue, machine vision can be used. A camera with appropriate lighting can capture images of the products, which are then analyzed and interpreted by a model. Designing a machine vision system that is tailored to the specific case is crucial. On the hardware side, there are many different types of cameras and lighting types to choose from. On the software side, both data-driven and non-data-driven techniques can be applied.

Implementing a machine vision system in a production line offers several advantages. The most obvious is that the quality control is less biased than when performed by a human operator. A machine vision system is not affected by fatigue or lack of concentration, which can be issues for human operators who perform the same task repeatedly. Additionally, an experienced operator may be more in touch with the quality control of a certain product than a newly hired operator. Due to the speed of the production process, it may be impossible to check each product individually. This can lead to samples being taken by the operator to test the quality. However, a machine vision system that can check and classify each product individually also can offer a remedy. The advantage of individual quality control is that bad quality products are much less likely to reach the costumer and consumer. This reduces financial losses and helps maintain the company's image.

This report examines the quality of ice cream cones, which is mainly determined by the topping consisting of sauce and/or nuts/curls. The production process of such ice cream cones is shown in Figure 1.1. The spread of the topping is an important factor in determining the quality of the ice cream cone. Due to the complexity of this determination, we have been assigned to use data-driven techniques for this classification task.

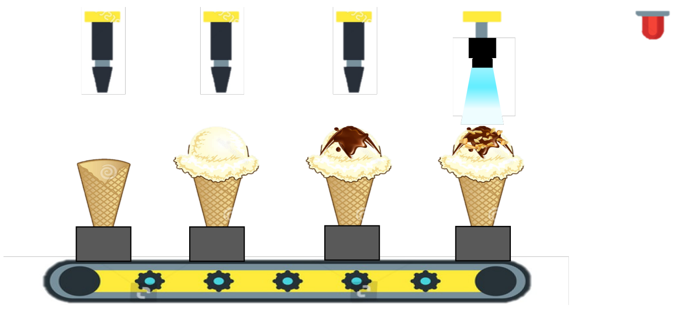


Figure 1.1: A production line of ice cream cones equipped with a machine vision system.

First, we review some existing solutions for similar types of classification/segmentation problems in Section 2. Next, we discuss our methodology in Section 3. We start with how we captured and analysed the image data. Then, we show how we pre-processed the raw data to obtain a dataset for training a Convolutional Neural Network (CNN). We also present some models that are suited for our use case. We use transfer learning to train them. Two different approaches are suggested here. The first approach is a binary classification problem, where the two labels relate to the general quality, namely *good* and *bad*. The second approach is a multi-label classification problem. Here, the model predicts two labels, one for the sauce quality and one for the nuts/curls quality. To introduce more explainability, we also investigate segmentation methods that we can use in combination with our trained models. We conclude our methodology with the metrics that we use to determine the performance of our trained models. These metrics need to be able to deal with an imbalanced dataset. The results of both approaches and segmentation are presented and discussed in Section 4. Finally, we give a conclusion of this report in Section 5.

2 Literature Study

This literature study examines two research papers that are relevant to our classification problem. Unfortunately, to the best of our knowledge, no research has been conducted on the classification of ice cream cones based on their quality. Therefore, in Section 2.1, we discuss the application of a Convolutional Neural Network (CNN) for the quality classification of Copra. This case is similar to ours based on the relevance of visual features that determine the quality of the product. In Section 2.2, we explore how we can segment pizza toppings using the stick growing and merging algorithm. Just like the pizzas, the ice cream cones also contain overlapping toppings and are very similar in this sense.

2.1 Classification

In this section, we investigate the application of a CNN for the quality classification of copra. Copra is the dried kernel of a coconut that produces coconut oil and copra meal as a by-product. The process to arrive at such a CNN, that is able to classify the quality of copra, is divided into three main parts, namely image acquisition, image pre-processing and network training. These are discussed in the following sections [1].

2.1.1 Image Acquisition

Based on several characteristics, such as the color of the copra, the extraneous matter and the aflatoxin-related molds (ARM), the copra can be divided into three different grades of quality. Table 2.1 shows these grades with their corresponding characteristics.

Characteristics	**Grade 1**	**Grade 2**	**Grade 3**
Color of meat	White to pale yellow	Brown to dark brown	Brown to dark brown
Extraneous matter (% max)	0.25	0.75	1.00
ARM (% max)	0	10	20

Table 2.1: Classification based on characteristic quality of copra [1].

A camera is mounted parallel to the surface at a distance of 0.3 m. The copra samples are placed on a solid blue background and the raw images are stored in the subfolders corresponding to their quality grade. For each quality grade, 450 images are obtained from different copra samples. This results in a dataset of 1350 images.

2.1.2 Image pre-processing

The raw images are pre-processed before they are fed to the neural network. The pre-processing steps are:

1. **Image enhancement:**
   The raw images are cropped and subsequently resized from 1280 x 1280 pixels to 512 x 512 pixels. This is done to limit the computational cost needed for training the model.
2. **Image segmentation:**
   Due to the use of a solid blue background, the background colour is thresholded. This results in a binary mask of the copra sample. This is done to clearly define the region of interest. The result of the background removal is shown in Figure 2.1.

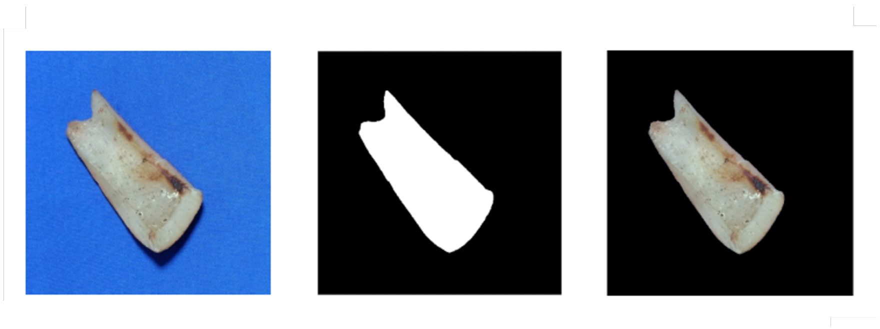


Figure 2.1: Background removal by segmenting the background and the copra using thresholding [1].

2.1.3 Network training

The dataset consisting of the pre-processed images is divided into a training set consisting of 80% of the data and a validation set consisting of 20% of the data.

The CNN architecture is made from scratch and consists of only three convolutional layers and max pooling is used after every convolution to reduce the learned features size. A dropout layer is also used to generalise the trained model. Stochastic gradient descent with momentum is used as optimisation method during training. The initial learning rate is 0.01 and the batch size is 32.

The best trained model is able to classify 90.74% of the validation set correctly. This model is only 16 layers deep and trained with 30 epochs.

2.2 Segmentation

In this section, we discuss the application of food image segmentation, more specifically of pizzas. This is a very specific application where traditional segmentation usually falls short of. Therefore, this section explains the stick growing and merging (SGM) algorithm, which is capable of segmenting complex food images. SGM consists of 4 major steps: stick initialisation, stick merging, subregion merging and boundary modification. These steps as well as some experimental results are discussed in the following sections [2].

2.2.1 Stick Initialisation

The goal of this step is to obtain the initial sticks. A stick is defined as a horizontal line where the pixels are homogeneous. The ends of these sticks correspond to edge points.

This step starts with traversing the image from the top-left to bottom-right. Adjacent pixels on a single row are compared and the intensity differences are put in a histogram as shown in Figure 2.2. The area under the curve can be divided into a stick area and a non-stick area based on the homogeneity criterion C_1.

$$C_1 : |x_1 - x_2| < T_1, \tag{1}$$

where x_1 and x_2 are the intensities of adjacent pixels and T_1 is the global information, which partitions the area. Within a row, the value of $T_1(row)$ can be different.

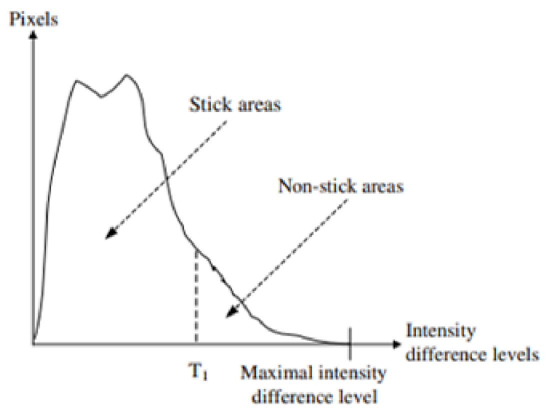


Figure 2.2: Difference histogram [2].

Based on C_1, the algorithm traverses the image again and stores the stick lengths with the corresponding number of pixels in another histogram. Based on this, the maximum stick length L_{max} can then be determined. This is a limitation to ensure that a stick does not cross multiple regions.

The algorithm then obtains an edge image and starts with growing sticks from homogeneous pixels. Edge points can function as starting seeds for the sticks. The stick is confirmed if the length is bigger than L_{min}. The growing process is stopped when the stick reaches another edge point or C_1 is not satisfied. The information about the start and stop indices of the sticks is stored in stick arrays while the up and down relationships of the sticks are stored in the adjacent stick lists.

2.2.2 Stick Merging

In this step, sticks are merged into initial subregions. A stick of a down-row will be merged with its most homogeneous stick of an up-row if the stick merging criteria C_{2min} is satisfied. C_{2min} consists of the following criteria:

1. The stick-stick homogeneity criterion:

$$C_{2ssh} : |intensity(upstick) - intensity(current)| < min(T_1, T_2), \tag{2}$$

   where T_1 is the global information and T_2 is the threshold between sticks that are homogeneous and inhomogeneous as shown in Figure 2.3.

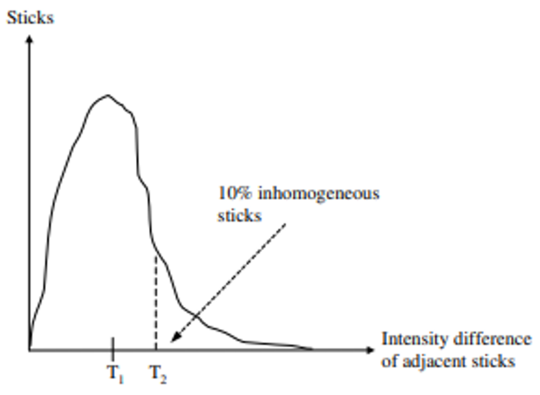


Figure 2.3: Stick homogeneity histogram [2].

2. The stick-stick minimal adjacent length:

$$C_{2ssa} : L_{ss} = min(T_3, L_{min}), \tag{3}$$

where L_{min} is the minimal stick length and T_3 is the threshold between adjacent sticks and least homogeneous parts as shown in Figure 2.4.

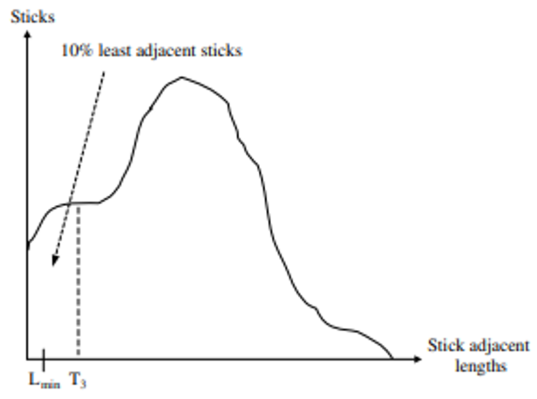


Figure 2.4: Stick adjacent length histogram [2].

3. The subregion-stick homogeneity criterion:

$$C_{2srsh} : |intensity(subregion) - intensity(current)| < K_4 * min(T_1, T_2), \tag{4}$$

where K_4 is an adjusting coefficient.

The algorithm starts with assigning each stick in the first row to a new subregion. Subsequently, the image is traversed to merge adjacent sticks based on the stick merging criteria C_{2min}. Figure 2.5 illustrates two possible scenarios that can occur. In the first scenario, the current stick is merged with a single subregion. In the second scenario, the current stick is merged with two subregions, resulting in the merging of those two subregions.

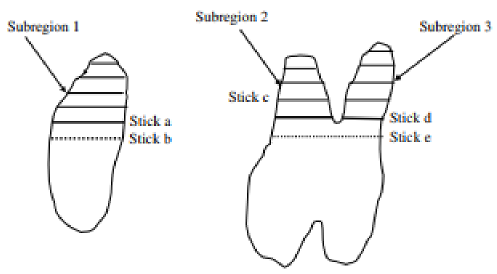


Figure 2.5: Stick-stick merging cases: only merging to one subregion or to two subregions [2].

2.2.3 Subregion Merging

The next step is to merge smaller subregions into larger subregions. Before this is carried out, four assumptions are made:

1. Very small subregion should be merged first.
2. Smaller subregions are more likely to be merged first than larger subregions.
3. A subregion should be merged with the subregion where it is most homogeneous too and has the longest adjacent boundaries with.
4. A subregion should be merged with its larger adjacent subregion if it satisfies a minimal subregion merging criterion.

The subregion merging step is similar to the stick-stick merging step, but now considering both up and down adjacent subregions instead of sticks. The subregions are sorted by size and they are merged when the subregion merging criteria C_3 is satisfied. C_3 consists of the following scores:

1. The homogeneity score:

$$S_{hregion}(a,b) = |I_{regiona} - I_{regionb}|/T_{rrh}, \tag{5}$$

where I_{region} is the average intensity of a region and T_{rrh} is equal to $K_6 * min(T_1, T_2)$.

2. The boundary score:

$$S_{conn}(a,b) = (1.0 - B_{shared})/min(B_a, B_b), \tag{6}$$

where B_{shared} is the length of the shared boundaries and B_{region} is the boundary length of the subregion.

3. The size score:

$$S_{size}(a,b) = (S_a/S_b)^{0.5}, \tag{7}$$

where S_{region} is the size of the subregion.

C_3 is the multiplication of these scores. A score of 0 indicates a definite merging of the subregions, while a score of 1 indicates no merging of the subregions.

2.2.4 Boundary Modification

Non-stick areas appear as noise or edges, leading to boundary roughness. To mitigate this roughness, boundary modification can be employed. Figure 2.6 showcases two possible approaches. The first approach involves merging a non-stick area m that is situated between two horizontal sticks a and b. In this case, one part of the non-stick area assumes the average intensity of stick a, while the other part assumes the average intensity of stick b. The second approach aims to fill a larger gap between 2 sticks a and b by growing pixels towards the center based on a aggregation criterion C_4.

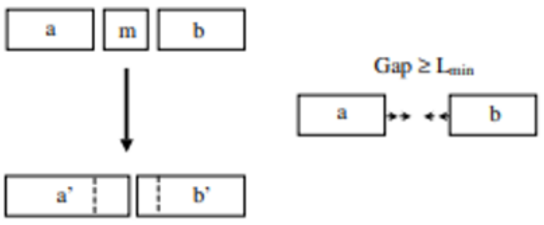


Figure 2.6: Boundary modification: attempt 1 (left) and attempt 2 (right) [2].

2.2.5 Experimental Results

The SGM algorithm, when applied to pizza topping segmentation, yields highly satisfactory results. The algorithm successfully segments the ham, red and green peppers, cheese shreds and tomato sauce as shown in Figure 2.7.

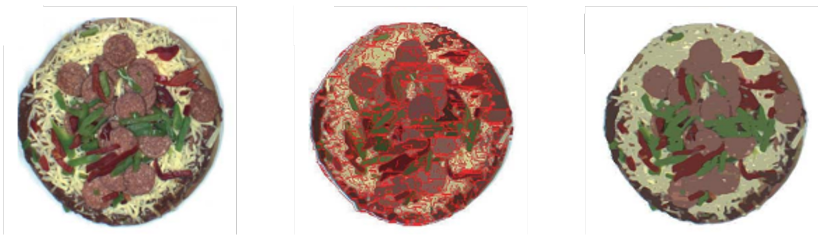


Figure 2.7: SGM used for pizza topping segmentation: the original image, the stick image and the result after boundary modification [2].

3 Methodology

The methodology used for this case is explained in detail in this section. Since this case is a conventional example of a classification problem, a general deep learning methodology framework is employed. The framework consists of several steps, which are illustrated in Figure 3.1. The first step involves capturing the data. Section 3.1 discusses how the data is captured and analysed. The second step, data pre-processing, focuses on preparing the dataset for training. This is discussed in more detail in Section 3.2. Section 3.3 provides more detailed information on various pre-trained models that are suitable for our classification problem. The retraining of the pre-trained models is explained in Section 3.4. Finally, a proper evaluation measure is chosen and discussed in Section 3.5. As you can see, the methodology framework used in our use case is similar to the one discussed in Section 2.1.

To enhance the explainability of our models, the K-means segmentation algorithm is explained in Section 3.6.

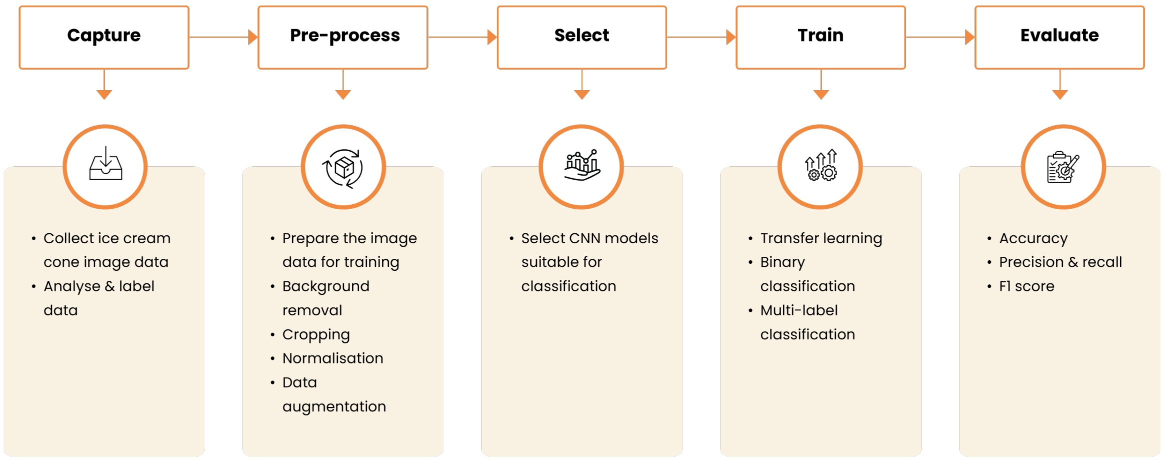


Figure 3.1: General deep learning methodology framework.

3.1 Data Capturing

Image data of ice cream cones is captured by Ysco and labelled by experts at the company. First, an in-depth look will be provided in the process of capturing the image data, including the hardware used and the content of the image data. Subsequently, the process of labelling the data will be explained.

The data is captured using a *Panasonic Lumix DMC-SZ10* camera in combination with a *PULUZ lighting box* that has a ring light with LED's. Both the camera and lighting box are shown in Figure 3.2. The captured images are RGB images with a width of 4608 pixels and a height of 2592 pixels. A single image requires 2.72 MB of storage capacity. The data contains both good and bad quality samples of three ice cream cones variants, namely:

1. Vanilla ice cream with chocolate sauce and bresilienne nuts.
2. Strawberry ice cream with strawberry sauce.
3. Chocolate ice cream with chocolate sauce and curls.

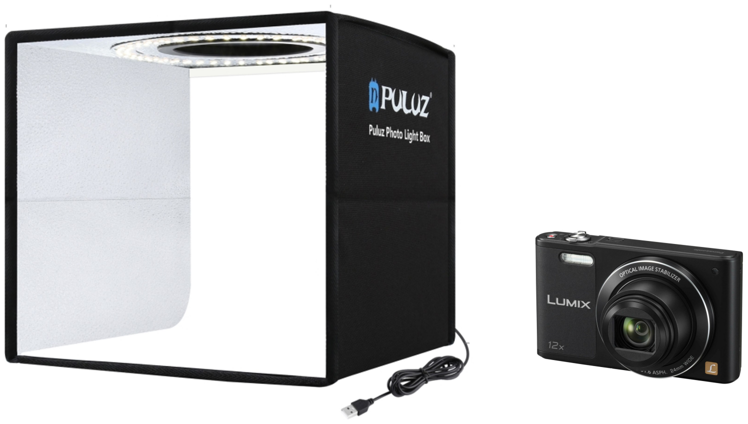


Figure 3.2: *PULUZ lighting box* with LED ring light and *Panasonic Lumix DMC-SZ10* camera [3] [4].

The ice cream cones are handcrafted in laboratory conditions to showcase the different possible quality stages of the ice cream cones. Figure 3.3 shows some examples of the original raw images for the three variants, with good and bad quality samples in the left and right columns, respectively.

After obtaining the image data, the next step is to label those images. The main label provided by the experts is the quality rating of the ice cream cones: *good* or *bad*. The two main indicators that influence the quality is the quantity and the coverage of the sauce and the nuts/curls. Therefore, the experts also provided us sub-labels based on the coverage of the two indicators. A summary of all the different images for the three variants with their respective labels is shown in Table 3.1.

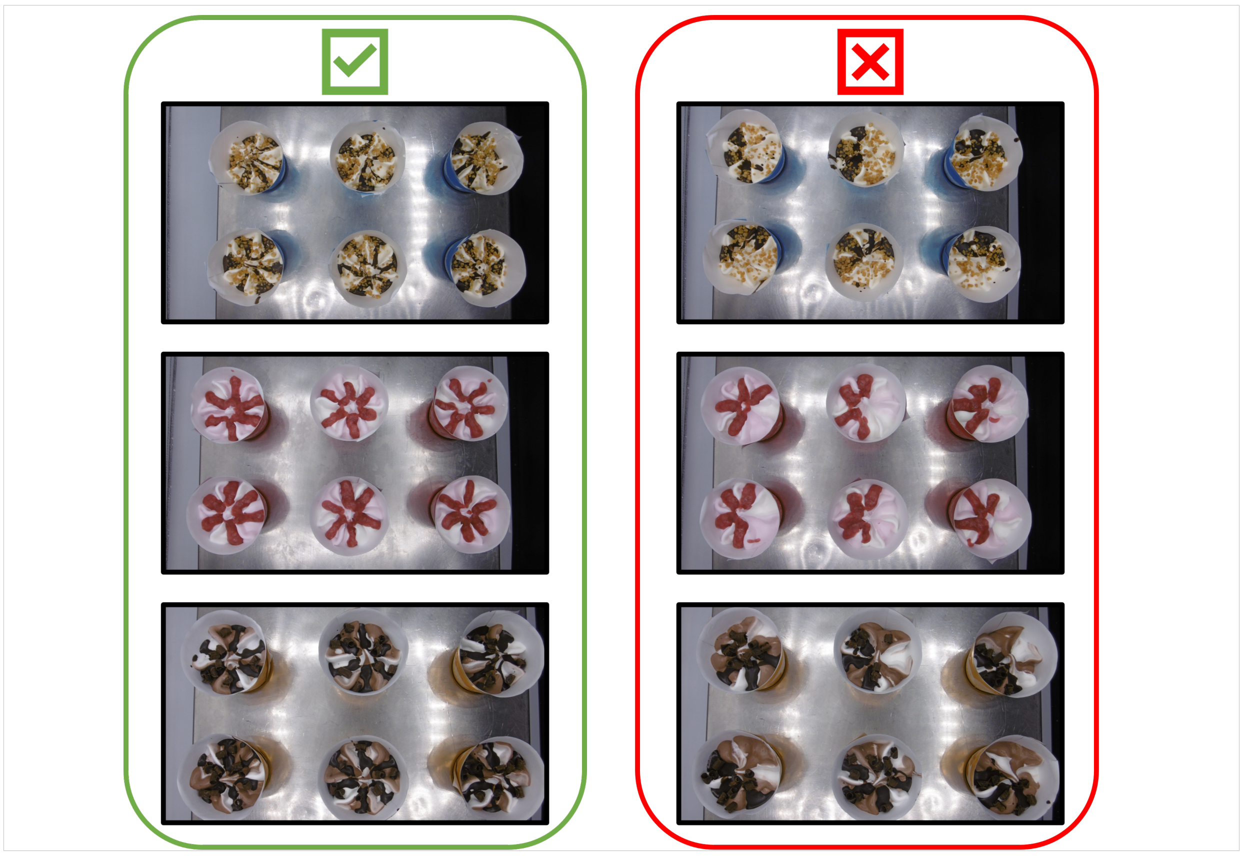


Figure 3.3: Raw data: good and bad quality samples.

This data is suitable for experimental purposes and exploring the ease of training a neural network for this case. However, when looking to deploy the trained model, image data directly from the production line is required. The ice cream cones in the production line are viewed by the camera at a fixed point of view and have a fixed size on the image data. When this is different, as in the provided image data by Ysco, it may result in a lot of faulty predictions and therefore a poor performance of the trained model. Therefore, it is crucial to capture the training data as close as possible to the real circumstances under which the real data will be captured from to ensure optimal performance of the trained model.

Variant	Description	Amount	Quality	Coverage sauce	Coverage nuts/curls
1	0g sauce, 0g nuts	2	Bad	Bad	Bad
	2g sauce, 2g nuts	2	Good	Good	Good
	2g sauce, 2g nuts	2	Bad	Good	Bad
	2g sauce, 2g nuts	2	Bad	Bad	Good
	2g sauce, 1.5g nuts	2	Good	Good	Good
	2g sauce, 0g nuts	2	Bad	Good	Bad
	0g sauce, 2g nuts	2	Bad	Bad	Good
2	0g sauce	2	Bad	Bad	n/a
	4g sauce	4	Good	Good	n/a
	4g sauce	4	Bad	Bad	n/a
	4g sauce	2	Bad	Bad	n/a
	2.5g sauce	2	Bad	Bad	n/a
3	0g sauce, 0g curls	2	Bad	Bad	Bad
	2g sauce, 1.5g curls	2	Good	Good	Good
	2g sauce, 1.5g curls	2	Bad	Bad	Bad
	2g sauce, 1g curls	2	Good	Good	Good
	2g sauce, 1.5g curls	2	Bad	Good	Bad
	2g sauce, 0g curls	2	Bad	Good	Bad
	0g sauce, 0g curls	1	Bad	Bad	Good

Table 3.1: Original raw data with labels.

3.2 Data Pre-processing

Data pre-processing is a crucial step in vision applications that involves transforming raw image data into a form that can be used to enhance the training of a neural network. This process can help to improve the performance and the robustness of the trained vision model, resulting in good predictions for new and unseen data. The pre-processing in this case involves several steps, which are discussed in more detail in the following sections.

3.2.1 Background Removal

As shown in Figure 3.3, some unwanted reflections can be observed in the background of the raw data. To fix this issue, the background needs to be removed. Background removal is important because it clearly defines the object of interest (the ice cream cone) and removes all the irrelevant information and noise. This will be beneficial for the speed of the model training and also for the performance of the trained model.

Thresholding cannot be used to remove the background, as in Section 2.1.2, because the background colour is not distinctively different from the ice cream colour. As a result, an AI-based tool named *rembg* is used to detect and to remove the background in the image [5]. An example of background removal on a raw data image is presented in Figure 3.4. The background is removed generally well. Some ice cream cones still have some background, such as the coloured side of the packaging, present. In such cases, manual removal of the background can be performed.

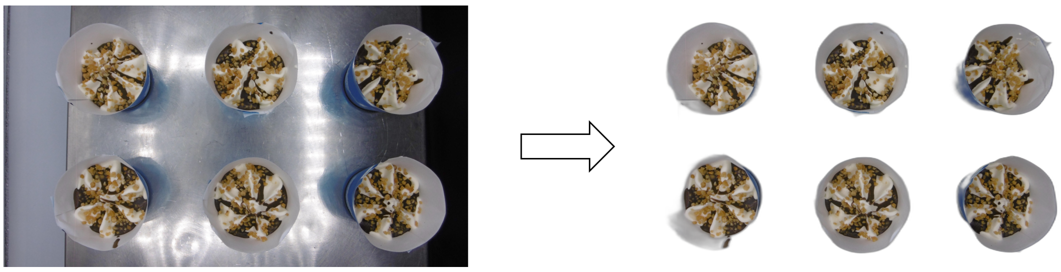


Figure 3.4: Background removal of raw data of the vanilla variant.

3.2.2 Cropping

Each image contains 6 individual ice cream cones. However, the images are captured from different positions and angles, resulting in variable sizes and positions of the cones on the image. Therefore, to obtain these samples, they must be cropped manually, leading to different sizes of the cropped images.

In Figure 3.5, an example of cropping is shown for the first ice cream cone sample. Table 3.2 provides a summary of the number of individual samples with their respective labels.

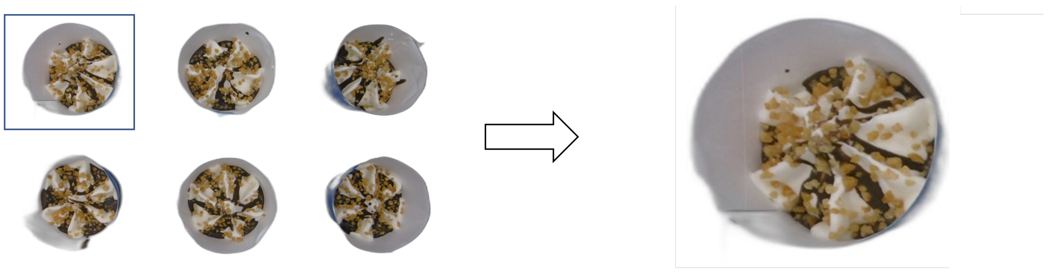


Figure 3.5: Cropping of the first ice cream cone sample.

Variant	Original samples	Quality	Coverage sauce	Coverage nuts/curls
1	12	Bad	Bad	Bad
	24	Bad	Good	Bad
	24	Bad	Bad	Good
	24	Good	Good	Good
2	60	Bad	Bad	n/a
	24	Good	Good	n/a
3	12	Bad	Bad	Bad
	24	Bad	Good	Bad
	18	Bad	Bad	Good
	24	Good	Good	Good

Table 3.2: Cropped original samples with labels.

3.2.3 Resizing

The crops obtained in the previous pre-processing step have varying image sizes. Therefore, it is necessary to resize all the crops uniformly. The images of the individual samples are resized to an image size of 256 x 256 pixels.

Resizing is an important pre-processing step for several reasons. Firstly, it ensures that all images have the same size, which is a requirement for most deep learning models. Secondly, resizing can impact the model accuracy and computational cost of training the model. It is important to choose an image size that is small enough so that every feature that is important for the classification problem remains visible [6].

3.2.4 Normalisation

Normalisation is a process that changes the range of pixel intensity values in an image. This operation ensures that images have the same pixel intensity distribution. The normalisation process can be represented by the following formula:

$$p_{ij,norm} = \frac{p_{ij} - mean}{std}, \; mean = 0.5 \; and \; std = 0.25$$

This formula is applied to each channel of the RGB image.

Normalising image data has several advantages. It induces consistency in the image data, which is essential for training deep learning models. Secondly, it makes computation more efficient [7].

3.2.5 Data Augmentation

To overcome the issue of limited data availability per variant, data augmentation can be performed. This technique creates new image samples by applying transformations to the previous image samples. This not only increases the dataset size but also makes it more diverse. Another benefit of data augmentation is that it reduces overfitting of the trained model [8].

It is important to apply the right data transformations that are relevant to this case. Two data transformations are used here:

1. Rotation: different rotations provide different distributions of the sauce and/or nuts/curls.
2. Brightness and contrast: different brightness and contrast settings simulate different lighting conditions.

An example of three data augmented variants of a cropped image sample is shown in Figure 3.6.

Variant	Original samples	Augmented samples	Total samples	Quality
1	60	600	660	Bad
	24	240	264	Good
2	60	600	660	Bad
	24	240	264	Good
3	54	540	594	Bad
	24	240	264	Good

Table 3.3: Total samples: original + augmented samples.

To increase the dataset size, ten data augmented variants were created for each ice cream cone sample. Table 3.3 provides an overview of the total number of original and augmented samples.

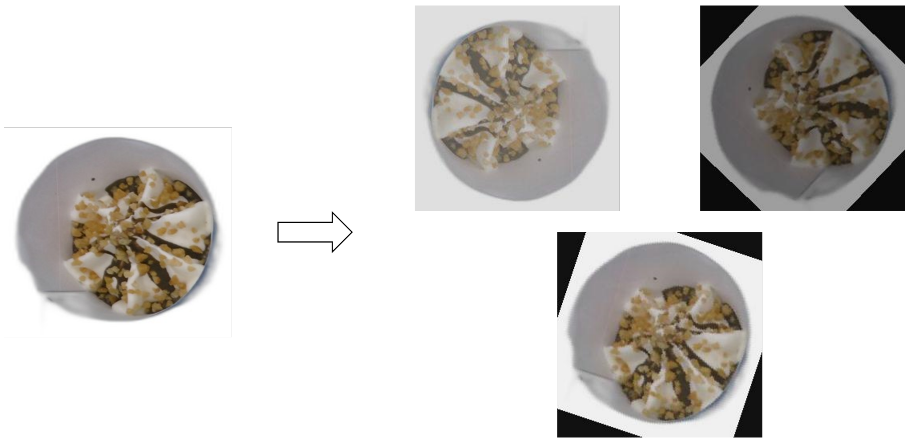


Figure 3.6: Data augmentation: varying rotation, brightness and contrast.

3.3 Model Choice

In this section, we present a selection of Convolutional Neural Network (CNN) models suitable for classification. This selection is based on the performance and complexity of the model architectures.

Since PyTorch is used as a framework in this paper, the selection consists of models available in *torchvision.models*. The documentation of *torchvision.models* shows us the different classification models with their corresponding top-1 and top-5 accuracy on the ImageNet dataset. The four CNN models that we will use here as well as their top-1 and top-5 accuracy are presented in Table 3.4.

Next, we will discuss the architectures of the selection of models consisting of the ResNet 34, ResNet 50, VGG 19 and EfficientNet B7 models.

CNN model	**Top-1 acc**	**Top-5 acc**
VGG 19	72.376	90.876
ResNet 34	73.314	91.420
ResNet 50	76.130	92.862
EfficientNet B7	84.122	96.908

Table 3.4: CNN classification models with the top-1 and top-5 accuracy [9].

3.3.1 VGG 19

VGG 19 is an image classification model pre-trained on the ImageNet dataset. This model is proposed in the paper ”Very Deep Convolutional Networks for Large-Scale Image Recognition” [10].

In this paper, the relationship between the neural network depth and accuracy is studied. The results show that increasing the depth of the network while using relatively small convolution filters (3x3 filters) leads to an improvement in accuracy. The VGG 19 architecture shown in Figure 3.7 was developed as a result of this work. This model comprises 16 convolutional layers with 3x3 filters and 3 fully connected layers. The last fully connected layer produces 1000 values, each corresponding to a class in the ImageNet dataset. By applying a softmax operation to these values, a probability score is obtained for each class.

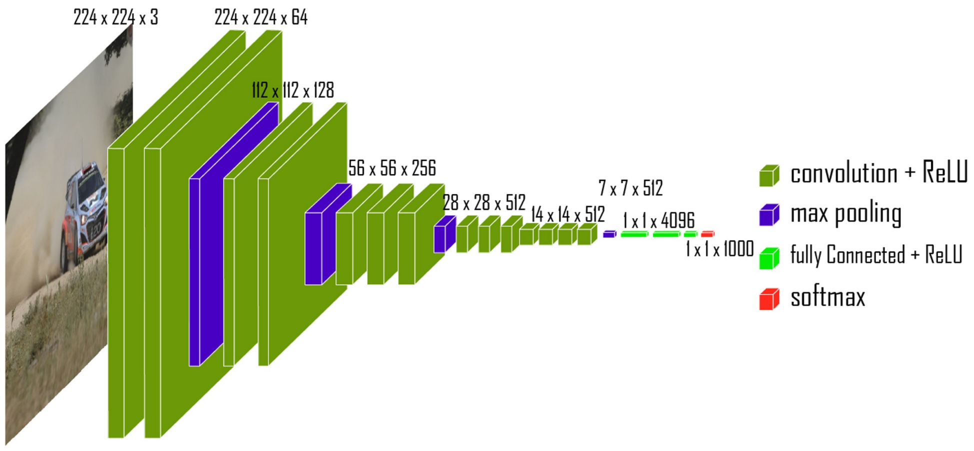


Figure 3.7: VGG 19 architecture [11].

Despite its improved accuracy, VGG 19 has some disadvantages. The model has 144 million trainable parameters, which makes training such a network from scratch very slow. Furthermore, storing these weights requires a significant amount of storage space.

3.3.2 ResNet 34

ResNet 34 is an image classification model that is also pre-trained on the ImageNet dataset. The architecture is described in the paper ”Deep Residual Learning for Image Recognition” [12].

In general, it is believed that the deeper a CNN, the better its performance. This is because deeper CNNs have more layers and therefore more parameters and features that can be learned. However, further research has shown that this is not always true. Suppose that you have a network of n layers. Copying the n layers and adding an extra identity mapping layer will result in network with n + 1 layers. We would expect that the network with n + 1 layers would have at least the same accuracy as the network with n layers. But the training and test error curves for the CIFAR-10 network in Figure 3.8 contradict this.

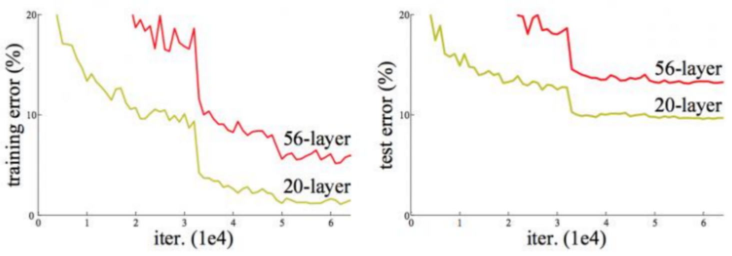
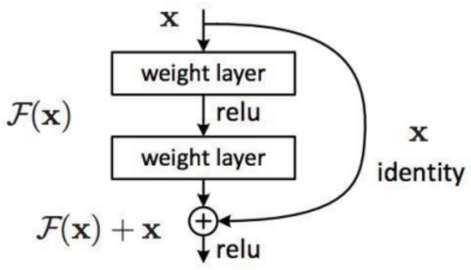


Figure 3.8: Training and test error curves for a CIFAR-10 network with 20 and 56 layers [12].

The reason for the worse performance is because direct mappings are hard to learn. Instead of learning the mapping from x to H(x), it is better to learn the residual F(x) = H(x) - x. To calculate H(x), we just add the residual F(x) to the input. This results in the ResNet block in Figure 3.9.



x
weight layer
$\mathcal{F}(\mathbf{x})$
relu
weight layer
x
identity
$\mathcal{F}(\mathbf{x}) + \mathbf{x}$
relu

Figure 3.9: ResNet block [12].

Figure 3.10 illustrates the architecture of ResNet 34. Each ResNet block is composed of a series of layers and a shortcut connection. The ResNet 34 model has 34 layers, with the last one being a fully connected layer which is responsible for the classification of the image.

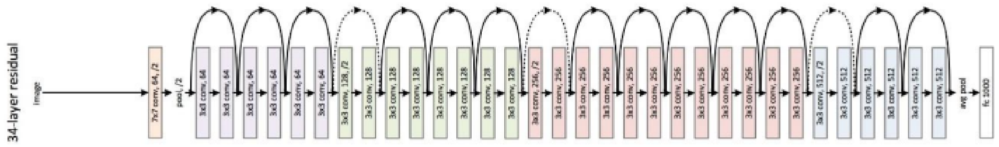


Figure 3.10: ResNet 34 architecture [12].

3.3.3 ResNet 50

ResNet 50 is a variant of the ResNet model that has 50 layers, making it deeper than the ResNet 34 variant. The addition of more shortcut connections in ResNet 50 should lead to a better performance compared to ResNet 34.

3.3.4 EfficientNet B7

EfficientNet B7 is a CNN that is developed by Google AI and pre-trained on the ImageNet dataset. The model is proposed in the paper ”EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks” [13].

The paper introduces a method called compound scaling that scales depth, width and image resolution of a CNN to ahcieve better performance and efficiency. Figure 3.11 illustrates this concept. The paper also introduces a new baseline model called EfficientNet. EfficientNet B7 is an upscaled version of the base model that is 8.4x smaller and 6.1x faster than the best existing CNN at the time.

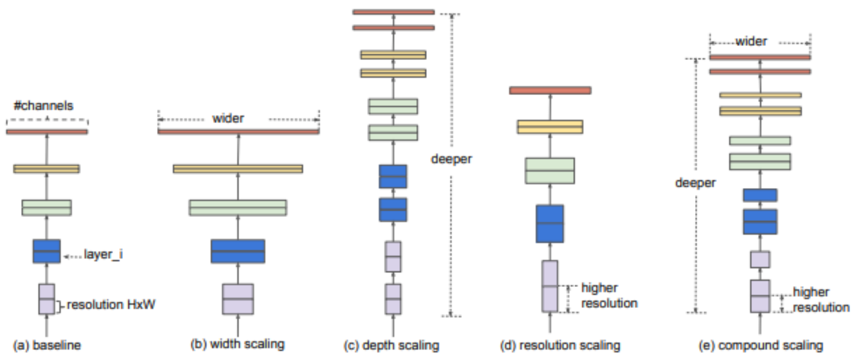


Figure 3.11: Upscaling of a CNN model: a) baseline network example, b) - d) conventional scaling and e) compound scaling [13].

3.4 Model Training

In this use case, we do not have to build and train classification models from scratch. Instead, we use transfer learning on the pre-trained models presented in Section 3.3. Transfer learning is a method that adapts a model trained on a specific problem to your own custom problem. Here, we copy the convolutional layers and their weights but change the fully connected layers so that they output probability values for the classes in our problem. When we retrain our model, the convolutional layer weights are frozen while the fully connected layer weights are trained. To prevent overfitting, the model that performs best on the validation data during the entire training procedure is stored. Because we apply stochastic gradient descent, we also have to define a batch size. This determines the number of data samples that are randomly chosen to update the gradient per epoch. We use a learning rate step scheduler which decreases the learning rate with a factor δ after a predefined amount of epochs.

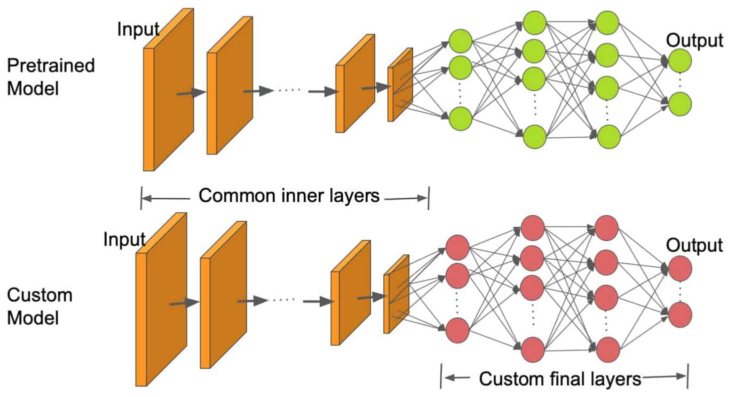


Figure 3.12: Transfer learning: the convolutional layers are copied while the fully connected layers are changed to match our custom problem [14].

For this use case, we propose two different approaches. The first approach is rather simple and intuitive. The main solution of this case lies in detecting whether the ice cream cone is of good or bad quality. This can be translated into a **binary classification** problem. The second approach tries to give more insights into why the ice cream cone is of poor quality, namely a fault based on the sauce or nuts/curls distribution. Therefore, a **multi-label classification** model is trained as a for this approach. The multi-label classification model will predict a label for the sub-classes being the sauce and the nuts/curls class. The quality prediction is then determined by using the logical AND operation on the labels for both sub-classes. All the different possibilities are shown in Table 3.5. This approach is a way of introducing explainability in our AI models.

Quality	**Sauce**	**Nuts/Curls**
Good	Good	Good
Bad	Bad	Good
Bad	Good	Bad
Bad	Bad	Bad

Table 3.5: Quality labels based on the sauce and nuts/curls sub-labels .

The dataset obtained by the pre-processing steps (see Table 3.3) is divided into training and validation sets. Initially, this is done in a 70-30 ratio. However, in this case, we also vary the size of the training set to determine how much data is required to train a good model.

To determine how hyperparameters such as the learning rate, batch size and amount of training epochs affect our model, we vary these parameters. The best values are then used to train a final model.

3.5 Model Evaluation

Accuracy is often used to evaluate the performance of a model. The formula for the accuracy is:

$$Accuracy = \frac{(TP + TN)}{(TP + TN + FP + FN)} \tag{8}$$

However, when the dataset is imbalanced, accuracy can give a distorted view of the model's performance. Therefore, additional metrics should be considered.

The first metric that we use is precision. Precision is the percentage of correctly predicted positives out of the total number of predicted positives [15]. The formula is:

$$Precision = \frac{TP}{(TP + FP)} \tag{9}$$

The second metric that we use is recall. Recall is the percentage of the correctly predicted positives out of the total number of actual positives [15]. The formula is:

$$Recall = \frac{TP}{(TP + FN)} \tag{10}$$

At last, we use the F1 score which is the harmonic mean of both the precision and recall values [15]. the formula is:

$$F1\ score = 2.\frac{Precision.Recall}{Precision + Recall} \tag{11}$$

These metrics have the advantage of focusing only on the relevant group (positives). This ensures that we do not get a distorted view due to correct predictions of irrelevant samples (negatives) that have a dominant share in our dataset.

3.6 K-means Segmentation

In addition to training a CNN, we also want to investigate the added value that segmentation can provide. We use K-means segmentation instead of setting the RGB threshold for both sauce and nuts/curls manually. K-means segmentation divides the image into k segments of similar colour. The following are steps involved in K-means clustering [16]:

1. Choose the number of clusters k.
2. Randomly initialise the cluster centers.
3. Assign each pixel to a cluster based on the RGB value distance.
4. Update the cluster centers.
5. Repeat steps 3 and 4 until convergence.
6. Create the segmented image.

Dividing the image into multiple segments makes it easier to locate the sauce and nuts/curls. Using their respective segment mask, we can calculate several different metrics such as the centroid, size and position relative to the ice cream cone center.

4 Results

In this section, we present the results of our experiments. First, we show the results of our binary classification approach in Section 4.1. Next, we present the results of our multi-label classification approach in Section 4.2. Finally, we show the results of the K-means segmentation in Section 4.3. We do this for all three variants. It is important to note that in our case, the bad quality ice cream cones are considered as positives, as this group is the most relevant for us to detect. The good quality ice cream cones are considered as negatives. In practice, the positives will be more frequent than the negatives. However, this is not reflected in Table 3.3.

4.1 Binary Classification

The results of our binary classification models for the three variants are presented in the following sections. We demonstrate the influence of the hyperparameters, namely epochs, batch size, and learning rate, using metrics defined in Section 3.5. Additionally, we present accuracy and loss curves for both the training and validation sets, as well as confusion matrices.

We vary the hyperparameters for the vanilla variant for each model architecture in Section 4.1.1. We then select the model architecture and the tuned hyperparameters with the best results to train a model for the strawberry and chocolate variants in Sections 4.1.2 and 4.1.3, respectively. If the values of the hyperparameters are not specified, then we use default values in our training procedure. Specifically, we use 10 training epochs, a batch size of 10, and a learning rate of 0.01.

4.1.1 Variant 1 - Vanilla

First, we train models for the four architectures for a varying number of training epochs. This results in three models per architecture. The results for these models are presented in Table 4.1. The trained models perform well across all architectures. The model based on the ResNet 50 architecture achieved the best F1 score (96.93%) when trained for 20 epochs. However, we observe that training for a bigger number of epochs does not always result in the best model. For example, we see that with the ResNet 34 and EfficientNet B7 architectures, we have already trained a model after 5 epochs that is better than for 10 or 20 epochs. The results also demonstrate that a more complex and deeper model is not always the better option. For instance, EfficientNet B7 has a more complex and deeper architecture but achieved an F1 score of 96.91%, which is about 2% lower than the simplest architecture here, namely VGG 19.

Next, vary the batch size to observe its impact on model training. We expect that increasing the batch size will have a positive impact on the model training, as more data samples are used to update the gradient. However, this will make the training slower. The results of the varied batch size are presented in Table 4.2. We indeed see that a larger batch size has a positive effect on the model. For the VGG 19, ResNet 34 and EfficientNet architectures, we obtain the best F1 score when training with a batch size of 20. This is not the case for the ResNet 50 architecture, where the best F1 score is obtained when training with a batch size of 5. A possible reason for this is that the batches of 5 consisted of more informative data samples than the batches of 20 for the ResNet 50 architecture. In Table 4.3, we show for each architecture the training loss and accuracy curves for the different batch sizes. The loss and accuracy for a batch size of 20 is already initially lower than the other batch sizes. Furthermore, the loss curves for the batch size of 20 remain lower than curves for other batch sizes, while the accuracy curves for the batch size of 20 remain higher than curves for other batch sizes. This shows that a larger batch size leads to a model with a better performance.

Model	**# epochs**	**Precision** (%)	**Recall** (%)	**F1 score** (%)
VGG 19	5	98.96	97.95	98.45
	10	98.96	97.95	98.45
	20	**98.97**	**98.97**	**98.97**
ResNet 34	5	**97.98**	**99.49**	**98.73**
	10	96.98	98.97	97.97
	20	97.00	**99.49**	98.23
ResNet 50	5	98.40	94.87	96.61
	10	97.98	**99.49**	98.73
	20	**98.98**	**99.49**	**99.23**
EfficientNet B7	5	97.41	**96.41**	**96.91**
	10	**97.88**	94.87	96.35
	20	97.84	92.82	95.26

Table 4.1: The precision, recall and F1 score for a variation (5, 10 and 20) of training epochs.

Model	**Batch size**	**Precision** (%)	**Recall** (%)	**F1 score** (%)
VGG 19	5	98.46	**98.46**	98.46
	10	**98.97**	**98.46**	**98.71**
	20	**98.97**	**98.46**	**98.71**
ResNet 34	5	96.04	**99.49**	96.91
	10	**97.46**	98.46	**97.96**
	20	**97.46**	98.46	**97.96**
ResNet 50	5	97.00	**99.49**	**98.23**
	10	**98.96**	97.44	98.19
	20	98.45	97.95	98.20
EfficientNet B7	5	95.00	97.44	96.20
	10	96.92	**96.92**	96.92
	20	**97.93**	**96.92**	**97.42**

Table 4.2: The precision, recall and F1 score for a variation (5, 10 and 20) of batch sizes.

Finally, we vary the learning rate to observe its impact on model training. Usually, in most cases, a default value of 0.01 is taken as the learning rate. The results of the varied learning rate are presented in the Table 4.4. We see that a learning rate of 0.01 provides a better F1 score for the ResNet 34, ResNet 50 and EfficientNet B7 architectures. The VGG 19 architecture does not follow the majority as it has a better F1 score for a learning rate of 0.1. In Table 4.5, we show for each architecture the training loss and accuracy curves for the different learning rates. In general, the curves show that a learning rate of 0.1 results in a higher loss but not necessarily a lower accuracy. The curves for a learning rate of 0.01 consistently have a loss approaching 0 and the highest accuracy.

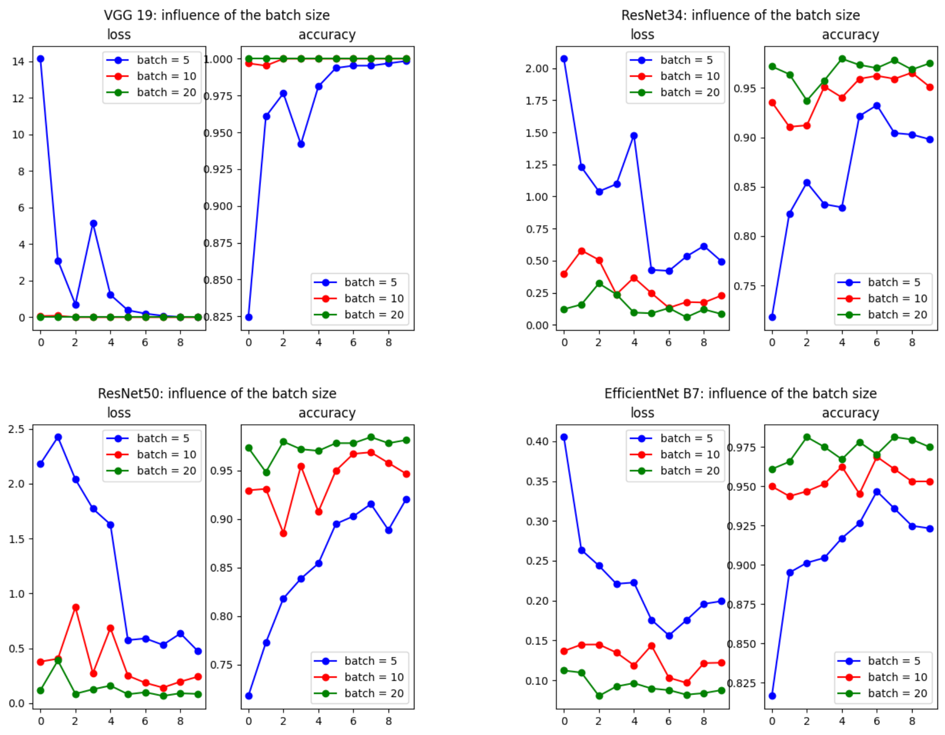


Table 4.3: Influence of the batch size on the training loss and accuracy curves.

Model	Learning rate	Precision (%)	Recall (%)	F1 score (%)
VGG 19	0.001	98.45	97.44	97.94
	0.01	97.98	**99.49**	98.73
	0.1	**98.48**	**99.49**	**98.98**
ResNet 34	0.001	**97.44**	97.44	97.44
	0.01	95.57	**99.49**	**97.49**
	0.1	97.41	96.41	96.91
ResNet 50	0.001	97.97	98.97	98.47
	0.01	**98.48**	**100.0**	**99.24**
	0.1	97.49	99.49	98.48
EfficientNet B7	0.001	96.32	93.85	95.06
	0.01	96.48	**98.46**	**97.46**
	0.1	**97.42**	96.92	97.17

Table 4.4: The precision, recall and F1 score for a variation (0.001, 0.01 and 0.1) of the learning rate.

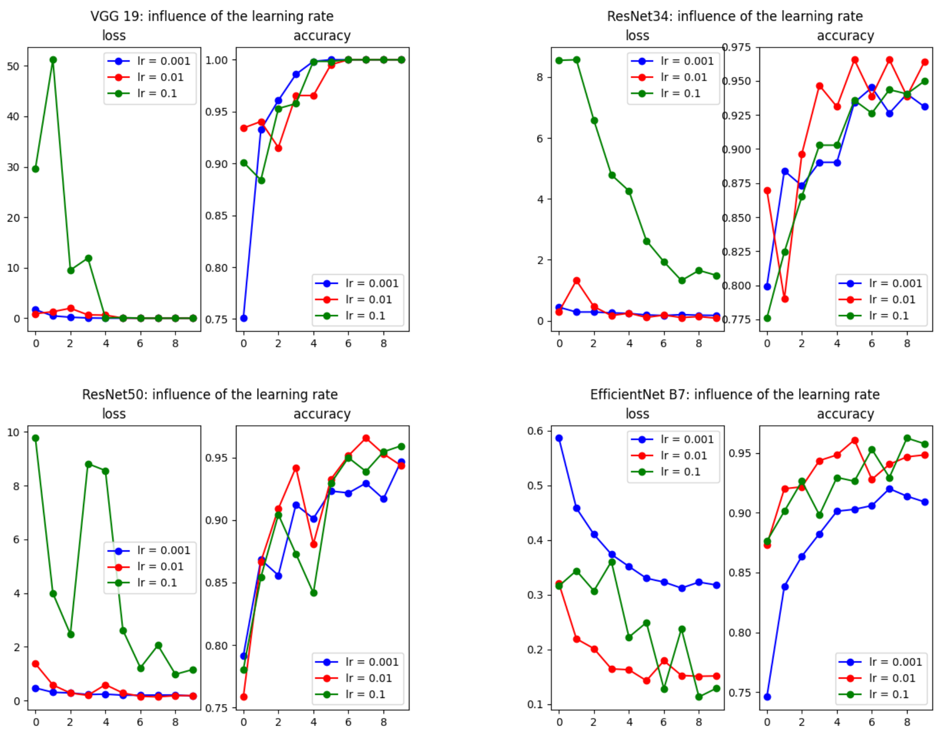


Table 4.5: Influence of the learning rate on the training loss and accuracy curves.

An additional important insight that we want to show here is the amount of data we really need to obtain a model with a decent performance. We use the same training and validation datasets as before. Here, we successively take certain subsets in decreasing size of the training set and evaluate the performance of the models trained on these subsets. The results are shown in Table 4.6. We observe that training a model on a smaller dataset has a negative impact on its performance. Although there is a decrease in performance, we see that this decrease is rather limited in our case. It is then important to compare the cost of obtaining additional labelled data and weigh it against the cost of incorrect classifications.

Model	**Training set size**	**Precision** (%)	**Recall** (%)	**F1 score** (%)
ResNet 50	159 (25%)	95.41	95.90	95.65
	319 (50%)	97.41	96.41	96.91
	478 (75%)	97.01	100.0	98.48
	638 (100%)	98.48	100.0	99.24

Table 4.6: The precision, recall and F1 score for a varying size (25%, 50%, 75% and 100%) of the training set.

Based on our experiments, we conclude that the models based on the ResNet 50 architecture produces the best results. We only need 10 epochs to train a model that meets our expectations. The values we ultimately choose for the batch size and learning rate are 20 and 0.01, respectively. We train a ResNet 50 model with these hyperparameters to obtain the best model for the binary classification of the vanilla variant. A confusion matrix of the predictions on the validation set is shown in Fig. 4.1. The model only predicts 3 ice cream cones incorrectly, which can be classified as false positives. This results in a high F1 score of 99.24%.

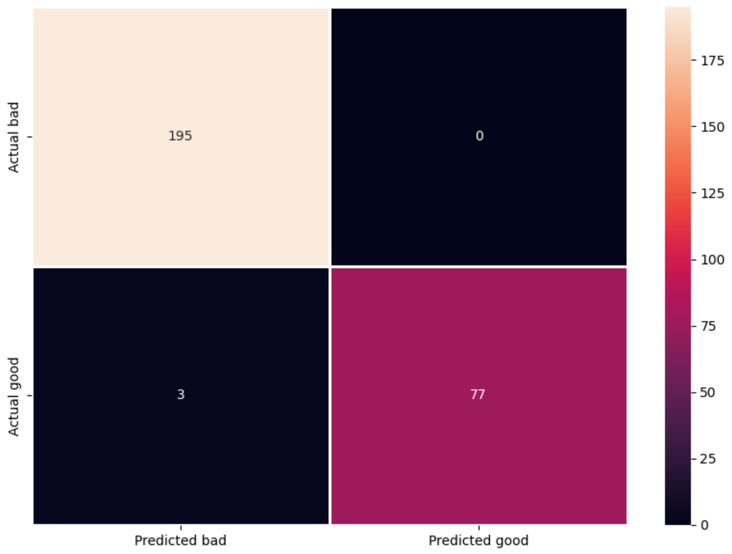


Figure 4.1: Confusion matrix of the ResNet 50 binary classification model for the vanilla variant, trained with a learning rate of 0.01 and batch size of 20..

4.1.2 Variant 2 - Strawberry

First, we start by training a ResNet 50 model with a batch size of 20 and a learning rate of 0.01 for this variant. However, the model performs less well for the strawberry variant than for the vanilla variant. Other architectures and hyperparameters are tested without obtaining an improvement. Therefore, we use the ResNet 50 model.

Further analysis indicates that the model struggles to classify ice cream cones of good quality. Figure 4.2 illustrates this. The presence of bad quality ice cream cones in the dataset, which have a good sauce coverage but an insufficient amount of it, is a possible reason for this. The model is unable to detect the amount of sauce and hence uses the coverage as an indicator of bad quality. As a result, some good quality ice cream cones are misclassified. We conclude that the limiting factor is nor the architecture nor the hyperparameters but rather the information provided by the image data.

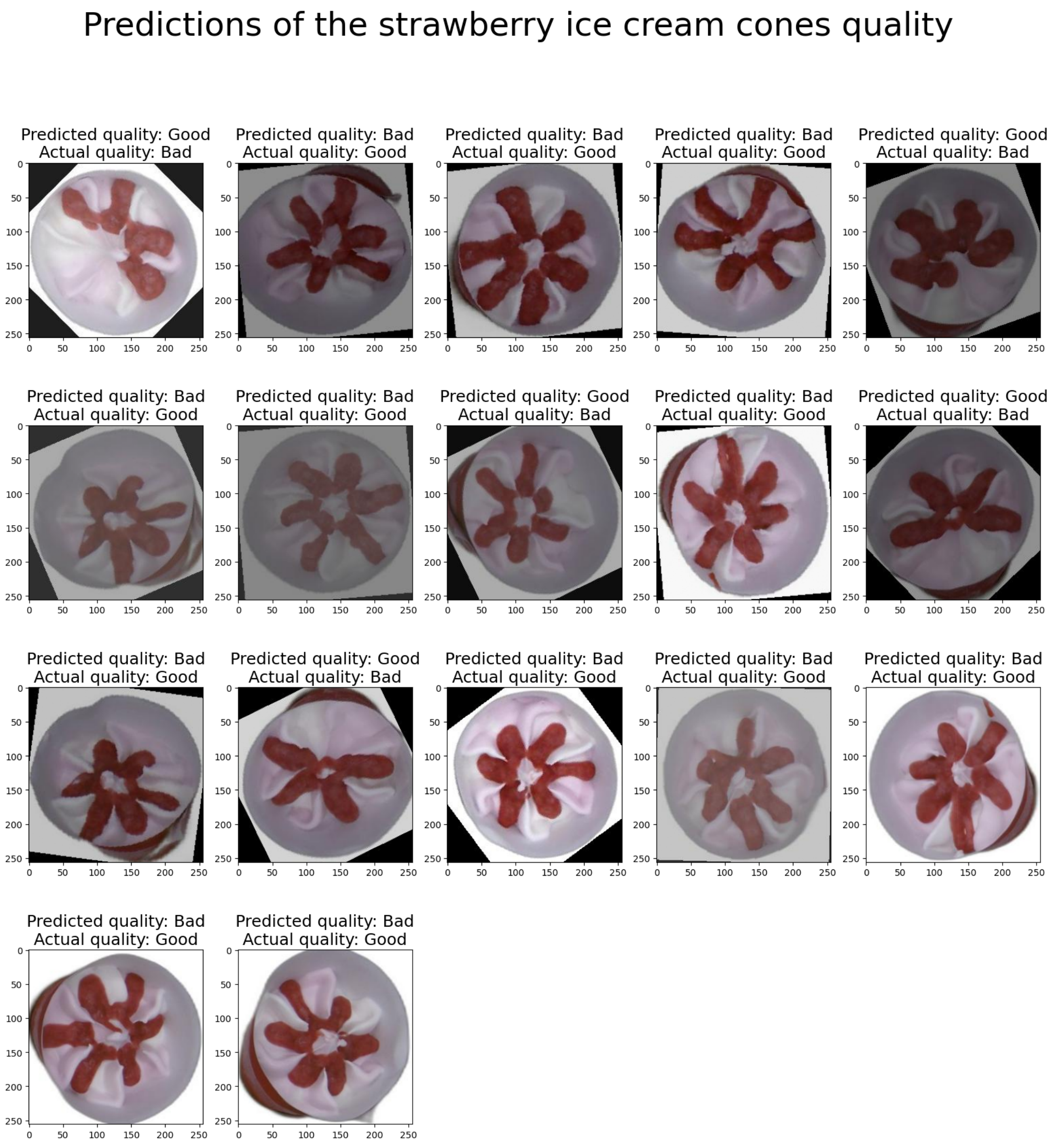


Figure 4.2: Incorrect predictions by the ResNet 50 model for the validation set of the strawberry ice cream cones

The best model obtained for the strawberry variant is a ResNet 50 model trained with a batch size of 20 and a learning rate of 0.01. This model has an F1 score of 95.80%. The performance metrics for this model are presented in Table 4.7. Additionally, a confusion matrix of the predictions on the validation set is shown in Fig. 4.3. The model mainly predicts ice cream cones of good quality incorrectly.

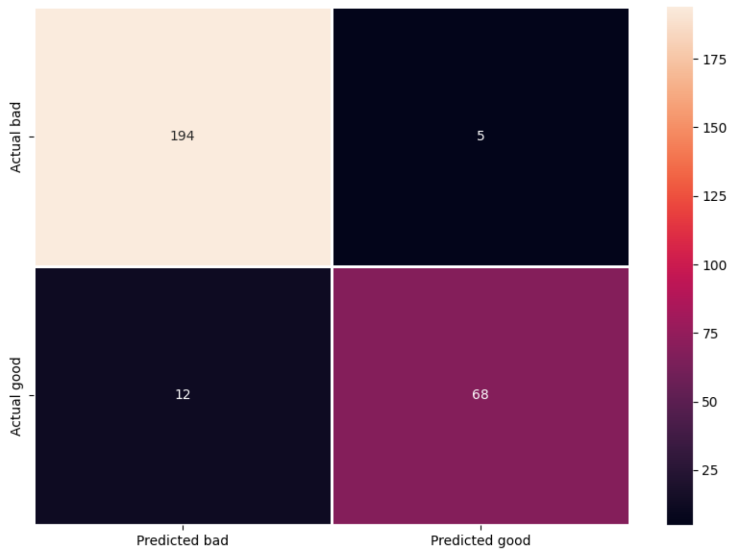


Figure 4.3: Confusion matrix of the ResNet 50 binary classification model for the strawberry variant, trained with a learning rate of 0.01 and batch size of 20.

4.1.3 Variant 3 - Chocolate

First, we start by training a ResNet 50 model with a batch size of 20 and a learning rate of 0.01 for this variant. The model performs less well for the chocolate variant than for the previous two variants. Other architectures and hyperparameters are tested without obtaining an improvement. Therefore, we use the ResNet 50 model trained for 15 epochs instead of 10.

Further analysis indicates that the model struggles to classify both good and bad quality ice cream cones. Figure 4.4 illustrates this. It is evident that the chocolate curls are challenging to distinguish when they are on top of the chocolate sauce, particularly when the brightness is low. This highlights the importance of proper lighting when capturing data. Doing so can ensure that the distinction between the chocolate sauce and curls is clearer, resulting in better model performance.

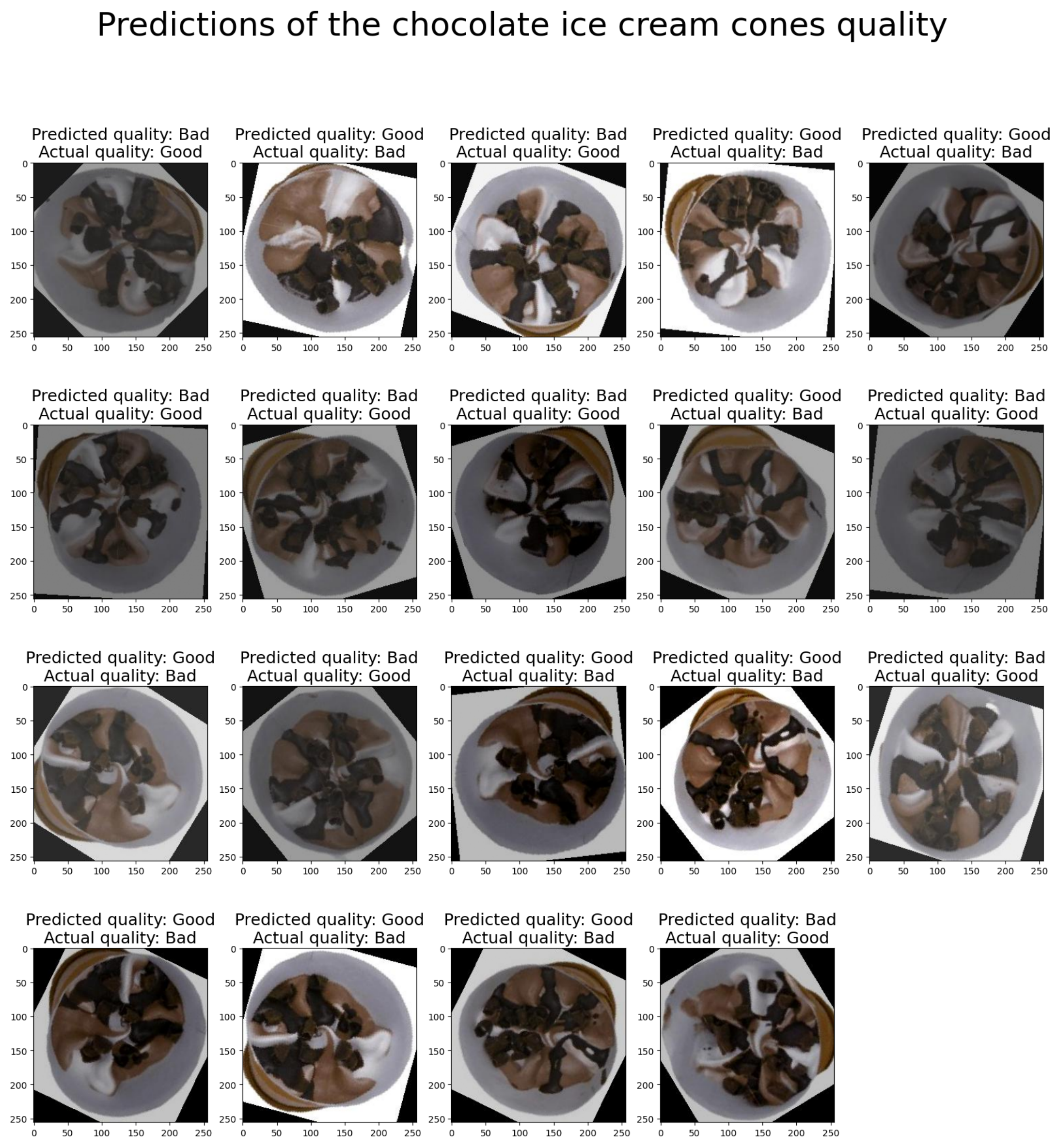


Figure 4.4: Incorrect predictions by the ResNet 50 model for the validation set of the chocolate ice cream cones

The best model obtained for the chocolate variant is a ResNet 50 model trained with a batch size of 20 and a learning rate of 0.01. This model has an F1 score of 94.68%. The performance metrics for this model are presented in Table 4.7. Additionally, a confusion matrix of the predictions on the validation set is shown in Figure 4.5.

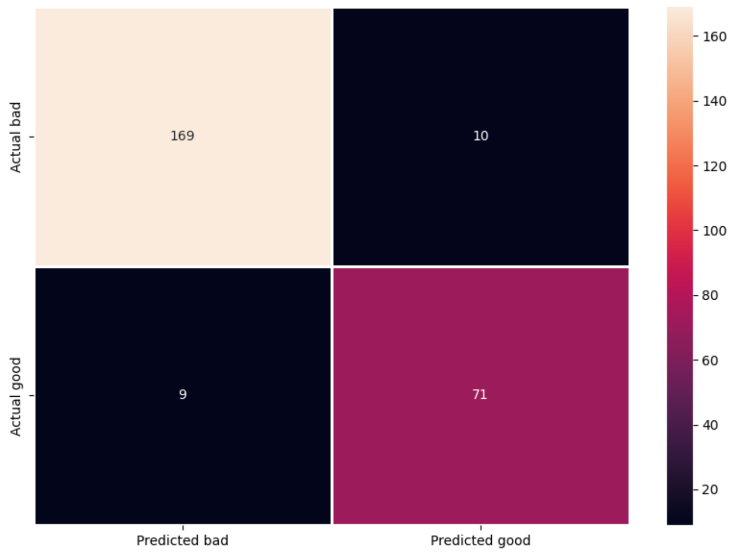


Figure 4.5: Confusion matrix of the ResNet 50 binary classification model for the chocolate variant, trained with a learning rate of 0.01 and batch size of 20.

Variant	**Model**	**Precision** (%)	**Recall** (%)	**F1 score** (%)
1	ResNet 50	98.48	100.0	99.24
2	ResNet 50	94.17	97.94	95.80
3	ResNet 50	94.94	94.41	94.68

Table 4.7: The best binary classification models for the three variants.

4.2 Multi-label Classification

The results of our multi-label classification models for the vanilla and chocolate variants are presented in Sections 4.2.1 and 4.2.2, respectively. The strawberry variant is excluded from this approach because its ice cream cones contain only sauce. The performance is evaluated using the metrics defined in Section 3.5 and visualised in the form of confusion matrices. We primarily discuss the performance of the multi-label classification models and compare it to that of the binary classification models of the same variant.

For both variants, we use a multi-head ResNet 50 model with 10 training epochs, a batch size of 10 and a learning rate of 0.01. It is worth noting that these hyperparameters are similar to the best hyperparameters found in Section 4.1.

4.2.1 Variant 1 - Vanilla

The performance metrics of the best multi-label classification model obtained for the vanilla variant are shown in Table 4.8. The predictions for both sub-classes have a high F1 score, indicating that our model can predict the labels of the sauce and nuts classes very well. Consequently, the predictions of the quality class also have a high F1 score. Figure 4.7 shows the confusion matrices of the predictions on the validation set for all three classes. The two sub-classes have a total of five individual incorrect predictions, resulting in the same number of incorrect predictions for the quality class.

Model	**Label**	**Precision** (%)	**Recall** (%)	**F1 score** (%)
ResNet 50	Sauce	98.31	99.15	98.72
	Nuts	99.07	99.07	99.07
	Quality	98.45	98.96	98.71

Table 4.8: The precision, recall and F1 score for the sauce, nuts and quality class.

The incorrect predictions for the sub-classes are shown in Figure 4.6. The reason for the model's incorrect predictions is not immediately visible since the ice cream cone images are unambiguous. However, the lower brightness of three out of five images may be a possible reason why the model has difficulty predicting the correct label for both sub-classes.

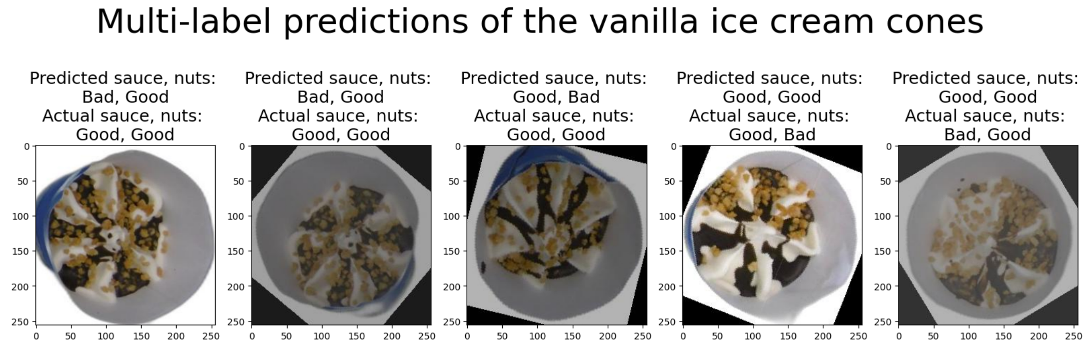

Multi-label predictions of the vanilla ice cream cones
Predicted sauce, nuts: Bad, Good Actual sauce, nuts: Good, Good
Predicted sauce, nuts: Bad, Good Actual sauce, nuts: Good, Good
Predicted sauce, nuts: Good, Bad Actual sauce, nuts: Good, Good
Predicted sauce, nuts: Good, Good Actual sauce, nuts: Good, Bad
Predicted sauce, nuts: Good, Good Actual sauce, nuts: Bad, Good

Figure 4.6: Incorrect predictions by the ResNet 50 multi-label classification model for the validation set of the vanilla ice cream cones

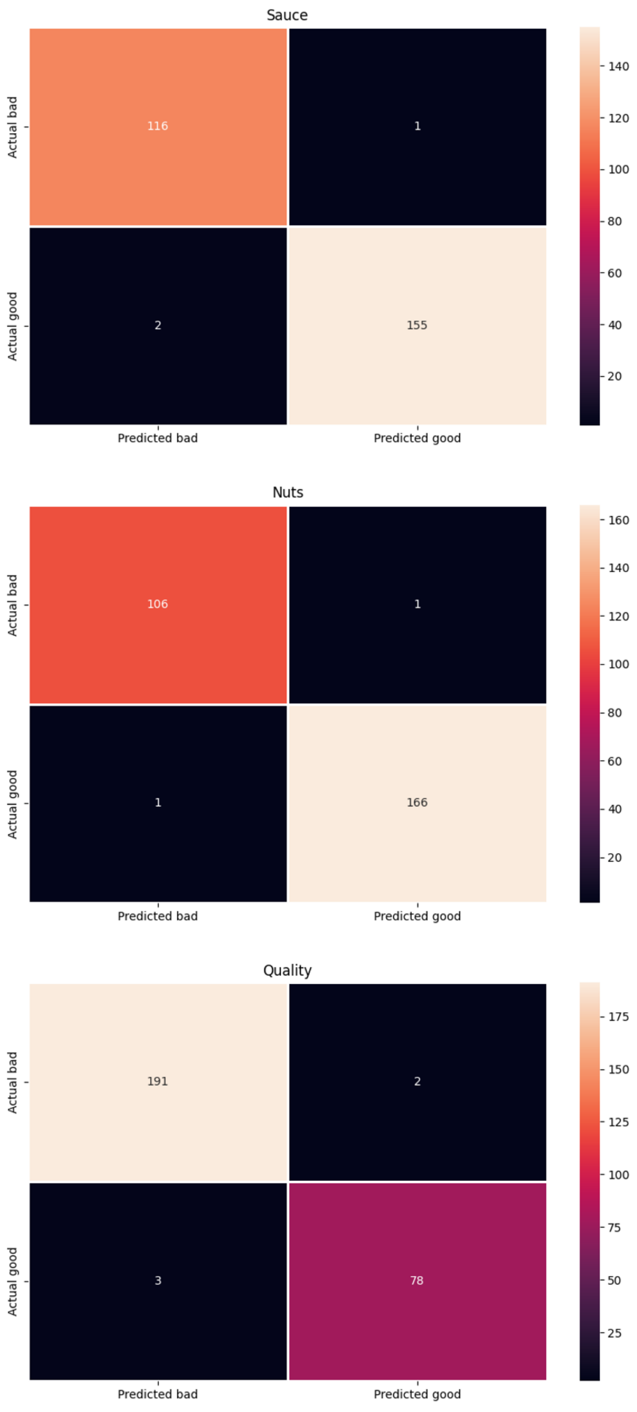


Figure 4.7: Confusion matrices of the ResNet 50 multi-label classification model for the vanilla variant.

Subsequently, we compare the performance of the binary and multi-label classification models by comparing the F1 score of the quality class for both models. The F1 score for the binary classification model is 99.48%, while the F1 score for the multi-label classification model is 98.71%. Note that the F1 score for the binary classification model is higher than what the Table 4.7 represents. This is because the binary classification model here is applied to the validation dataset that was used when training the multi-label classification model. While the binary classification model performs slightly better, using a multi-label classification model offers additional benefits such as finding the cause of the poor quality. This way, the problem can be tackled en resolved more quickly.

4.2.2 Variant 3 - Chocolate

The performance metrics of the best multi-label classification model obtained for the chocolate variant are shown in Table 4.9. The predictions of the sauce class have a significantly higher F1 score than the predictions of the curls class. As a result, the F1 score of the quality class predictions is limited by the lower F1 score of the curls class predictions. Figure 4.8 shows the confusion matrices of the predictions on the validation set for all three classes. The two sub-classes have a total of thirty-four individual incorrect predictions, but this does not necessarily result in the same number of incorrect predictions for the quality class. To explain this phenomenon, let's consider an ice cream cone with 2 bad labels for both sub-classes, resulting in a bad quality label. Suppose that the model predicts a good label for the sauce class and a bad label for the curls class, then this will also result in a bad quality label. This example demonstrates that an incorrect prediction in one or both sub-classes does not necessarily lead to an incorrect prediction of the quality class.

Model	**Label**	**Precision** (%)	**Recall** (%)	**F1 score** (%)
ResNet 50	Sauce	96.23	94.44	95.33
	Curls	94.84	90.18	92.45
	Quality	94.09	95.63	94.85

Table 4.9: The precision, recall and F1 score for the sauce, curls and quality class.

A sub-selection of the incorrect predictions for the sub-classes are shown in Figure 4.9. Here, we can observe again that the chocolate curls are difficult to distinguish from the chocolate sauce in general. A lower brightness enhances this effect. Therefore, the labels of many lower brightness ice cream cone images are difficult to predict, as shown in the Figure 4.9.

Similar to the vanilla variant, we also compare the performance of the binary and multi-label classification models based on their F1 scores. The F1 score for the binary classification model is 97.28%, while the F1 score for the multi-label classification model is 94.85%. Note that the F1 score for the binary classification model is higher than what Table 4.7 represents. This is because the binary classification model here is applied to the validation dataset that was used when training the multi-label classification model. The binary classification model performs significantly better than the multi-label classification model. A trade-off must be made here between the additional benefits that a multi-label classification model offers and the better performance of the binary classification model.

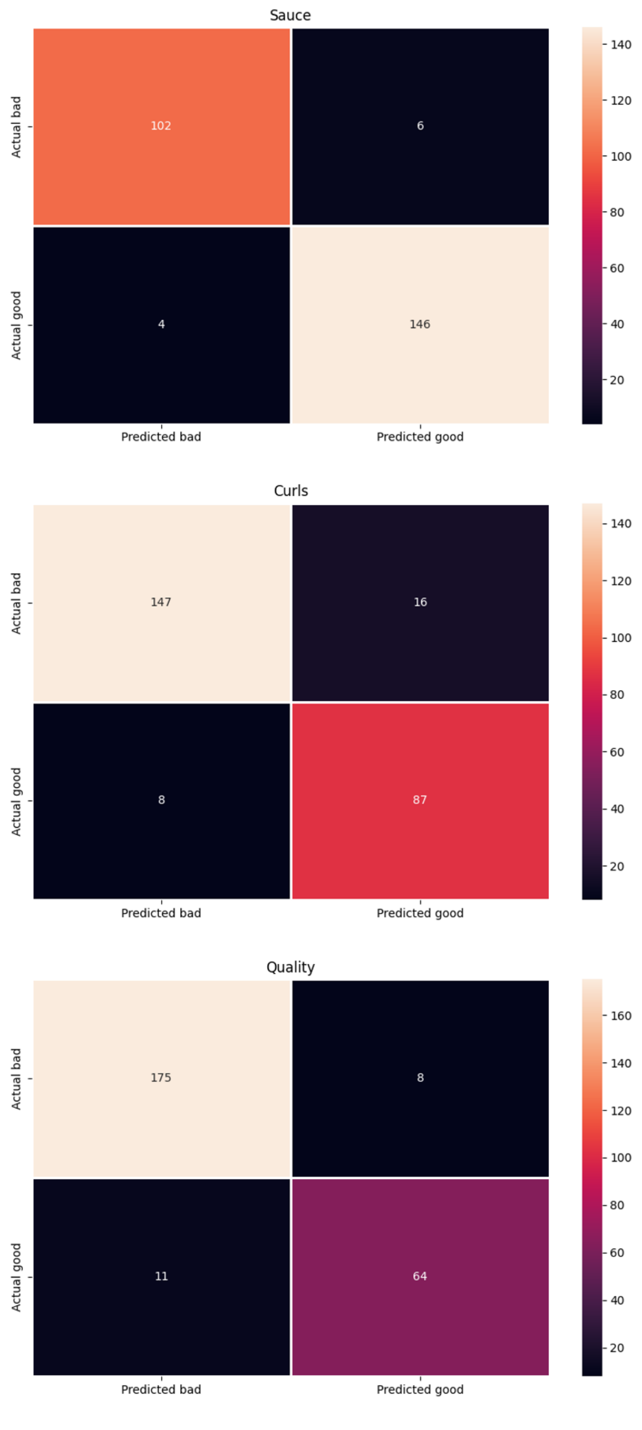


Figure 4.8: Confusion matrices of the ResNet 50 multi-label classification model for the chocolate variant.

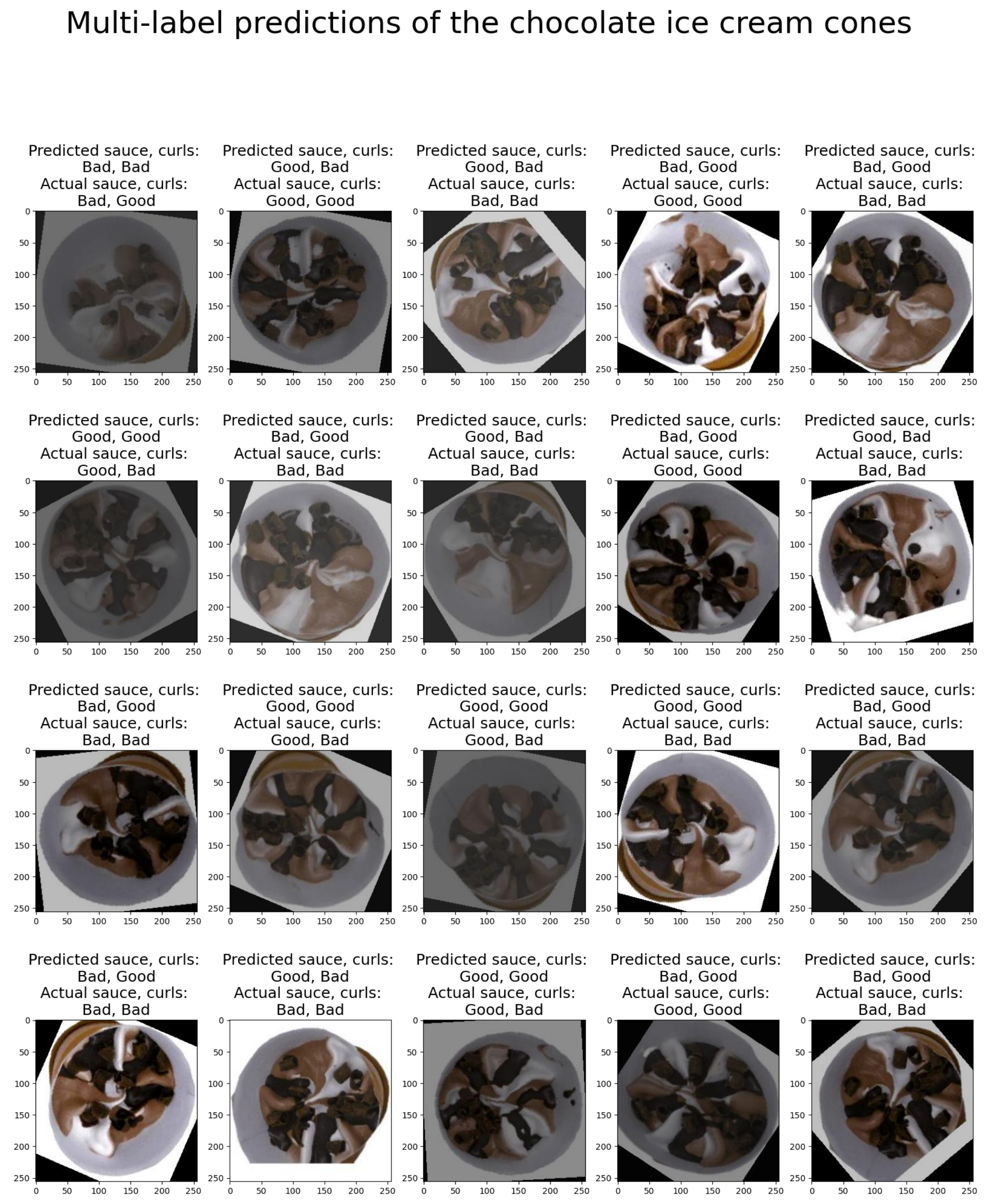

Multi-label predictions of the chocolate ice cream cones
Predicted sauce, curls: Bad, Bad Actual sauce, curls: Bad, Good
Predicted sauce, curls: Good, Bad Actual sauce, curls: Good, Good
Predicted sauce, curls: Good, Bad Actual sauce, curls: Bad, Bad
Predicted sauce, curls: Bad, Good Actual sauce, curls: Good, Good
Predicted sauce, curls: Bad, Good Actual sauce, curls: Bad, Bad
Predicted sauce, curls: Good, Good Actual sauce, curls: Good, Bad
Predicted sauce, curls: Bad, Good Actual sauce, curls: Bad, Bad
Predicted sauce, curls: Good, Bad Actual sauce, curls: Bad, Bad
Predicted sauce, curls: Bad, Good Actual sauce, curls: Good, Good
Predicted sauce, curls: Good, Bad Actual sauce, curls: Bad, Bad
Predicted sauce, curls: Bad, Good Actual sauce, curls: Bad, Bad
Predicted sauce, curls: Good, Good Actual sauce, curls: Good, Bad
Predicted sauce, curls: Good, Good Actual sauce, curls: Good, Bad
Predicted sauce, curls: Good, Good Actual sauce, curls: Bad, Bad
Predicted sauce, curls: Bad, Good Actual sauce, curls: Bad, Bad
Predicted sauce, curls: Bad, Good Actual sauce, curls: Bad, Bad
Predicted sauce, curls: Good, Bad Actual sauce, curls: Bad, Bad
Predicted sauce, curls: Good, Good Actual sauce, curls: Good, Bad
Predicted sauce, curls: Bad, Good Actual sauce, curls: Good, Good
Predicted sauce, curls: Bad, Good Actual sauce, curls: Bad, Bad

Figure 4.9: Incorrect predictions by the ResNet 50 multi-label classification model for the validation set of the chocolate ice cream cones

4.3 K-Means Segmentation

In the following sections, we show the K-means segmentation results for the three variants. Our first aim is to segment the sauce and the nuts/curls separately. We then compute the centroid, the size and the position of the segmentation mask relative to ice cream cone center. We do this for a good quality sample first. Afterwards, we apply this on different degrees of bad quality samples to display the extra explainability.

4.3.1 Variant 1 - Vanilla

We first apply K-means segmentation to a good quality sample. The original image and the segmented result are shown in Figure 4.10. The K value, that gives the best result, is 4. The 4 segmentation clusters are:

1. Background
2. Ice cream and packaging
3. Sauce
4. Nuts

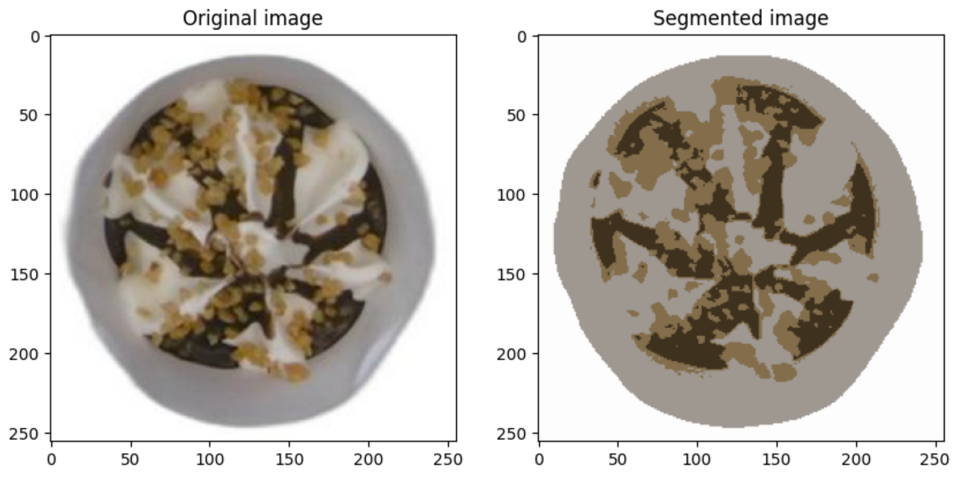


Figure 4.10: Segmented result of a good quality vanilla sample.

To remove the noise in the segmentation masks, we use morphological transformations. More specifically, we use the opening transformation. This means that the mask is first eroded and that result is then dilated. Both operations are done with a 3x3 kernel. Figure 4.11 shows the opening transformation on both the sauce and nuts masks. It has a better effect on the nuts mask due to the removing of the sauce contours that are clearly present in the original nuts mask. However, some non-nut parts remain in the nuts mask. To remove them, we would need a bigger kernel but that would make the masks less accurate and more blocky.

With the segmentation masks for sauce and nuts, we can now compute different features that can help us explain why the quality is poor. We start with the calculation of the centroids of both masks. The centroids are represented by their x and y pixels as seen in Table 4.10. Ideally, the

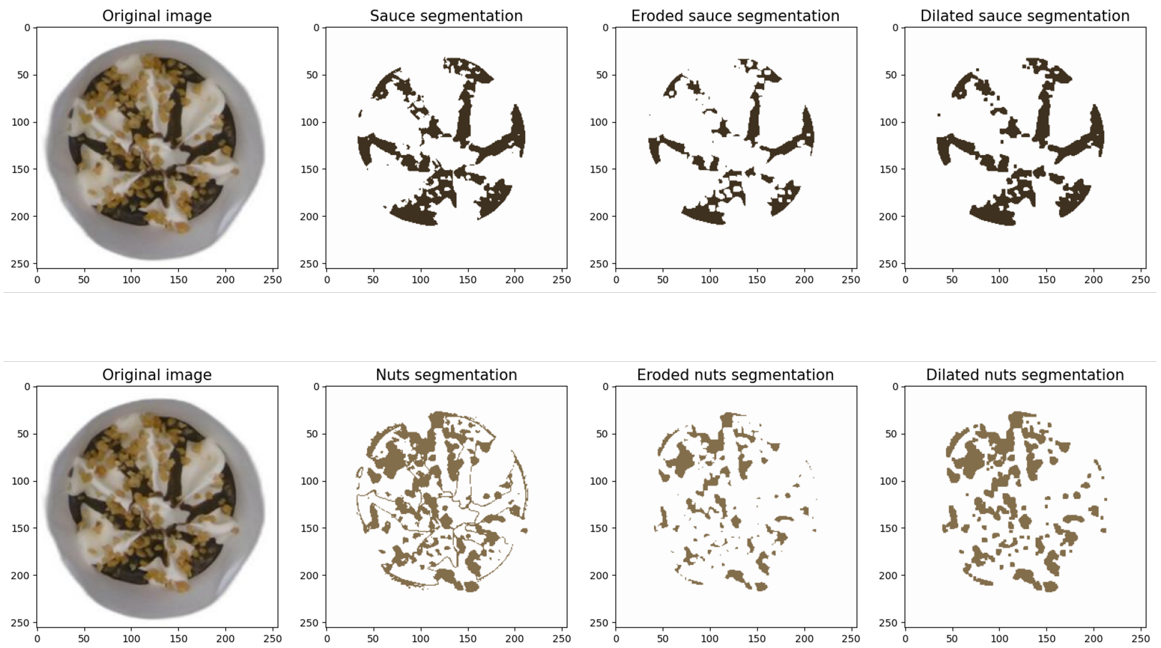


Figure 4.11: Opening of the sauce and nuts masks for the vanilla variant.

centroids of both masks should be close to the ice cream cone center. Next, we measure the size (width and height) of the masks in pixels as an indication of the spread. Finally, we calculate the Euclidean distances and angles between the centroids and the center. This is visualised in Figure 4.12. The green dot represents the center while the red and blue dots represents the sauce and nuts centroids, respectively.

We repeat the same steps for vanilla ice cream cones of bad quality. Table 4.10 shows the results. On the second ice cream cone, the nuts are mostly present at the bottom on the left. The distance of 66 pixels and the angle of 260° show this. The nuts mask size is remarkably large here. This is because there are also some nuts at the top of the ice cream cone, which makes the nuts mask bigger. On the third ice cream cone, the sauce is mostly present at the top on the left. The distance of 40 pixels and the angle of 146° show this. The last two ice cream cones only contain either sauce or nuts. It is remarkable that when the sauce is well spread, the distance is up to 9 pixels from the ice cream cone center. The nuts have a larger distance when spread well because the nuts masks are more porous than the sauce masks. This is due to the small size of the nuts and the fact that they are sprinkled more randomly.

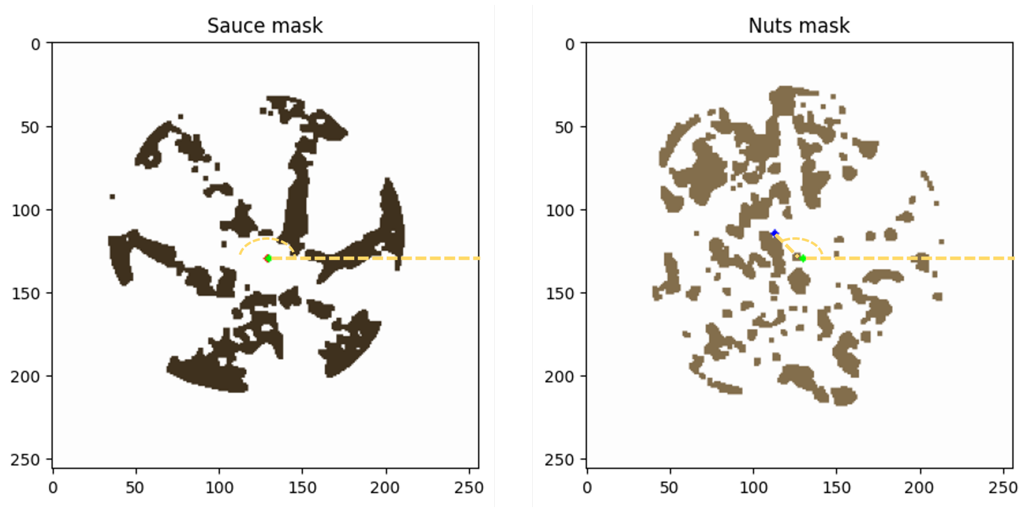


Figure 4.12: Center (green) and centroids (red, blue) of the sauce and nuts masks. The angle between both are indicated in yellow.

Ice cream cone	Sauce mask			Nuts mask		
	CT(x,y)	**S(w,h)**	**POS(d,a)**	**CT(x,y)**	**S(w,h)**	**POS(d,a)**
	(129,130)	(178,178)	(1,180°)	(113,115)	(192,175)	(23,139°)
	(125,124)	(200,175)	(1,90°)	(114,190)	(176,157)	(66, 260°)
	(97,114)	(146,146)	(40,164°)	(141,152)	(184,188)	(28,283°)
	(126,116)	(227,211)	(9,84°)	n/a	n/a	n/a
	n/a	n/a	n/a	(133,144)	(186,193)	(19,264°)

Table 4.10: The centroids, sizes and positions of the vanilla segmentation masks relative to the centers of ice cream cones of different qualities.

4.3.2 Variant 2 - Strawberry

Similar to the vanilla variant, we begin with performing K-means segmentation on a good quality sample. The original image and segmented image are shown in Figure 4.13. The K value, that gives the best result, is 3. The 3 segmentation clusters are:

1. Background
2. Ice cream and packaging
3. Sauce

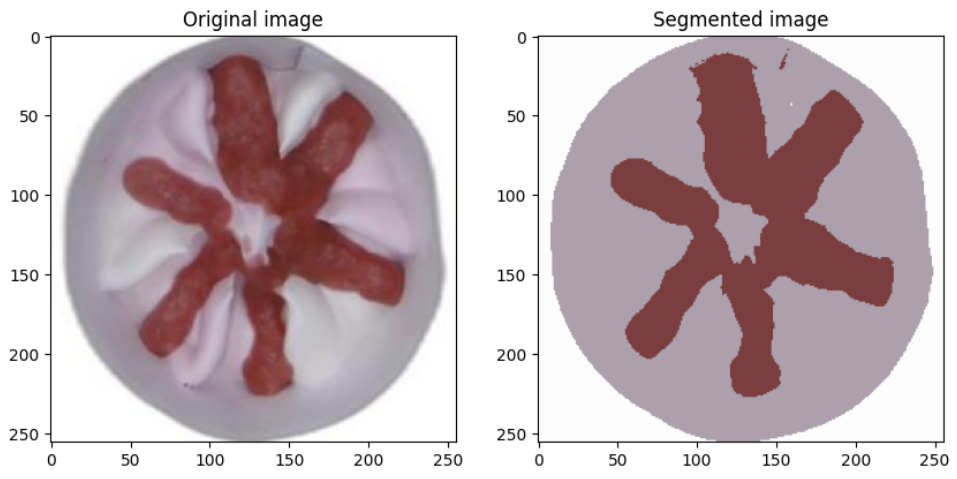


Figure 4.13: Segmented result of a good quality strawberry sample.

We also apply the opening transformation with a 3x3 kernel. Figure 4.14 shows the opening transformation on the sauce mask. It removes the little bit of noise that is present in the mask.

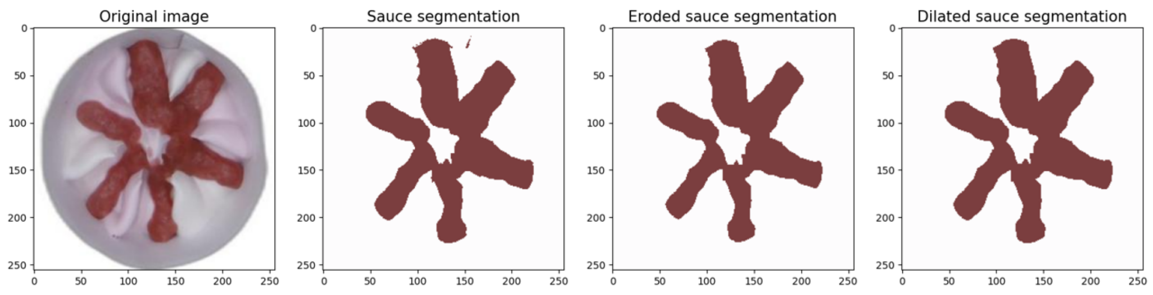


Figure 4.14: Opening of the sauce mask for the strawberry variant.

We again compute the different features for the sauce mask. The angle between the centroid and the center is visualised in Figure 4.15. The green dot represents the center while the red dot represent the sauce centroid. We repeat the same steps for vanilla ice cream cones of bad quality. Table 4.10 shows the results.

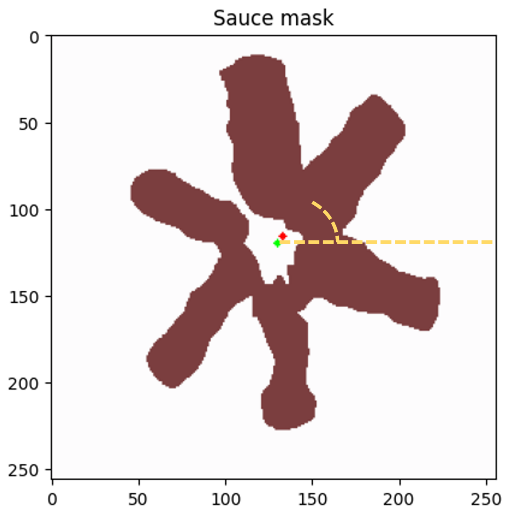


Figure 4.15: Center (green) and centroid (red) of the sauce mask. The angle between both are indicated in yellow.

We repeat the same steps for strawberry ice cream cones of bad quality. Table 4.11 shows the results. On the second ice cream cone, the sauce is spread too much around the center, causing a thick accumulation of sauce. This is indicated by the smaller size of the mask. On the third ice cream cone, the sauce is located on the left half of the cone. The distance of 33 pixels and the angle of 183° confirm this observation. The mask size also has a small width but large height as can be seen on the image of that ice cream cone.

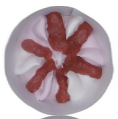
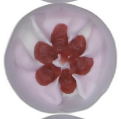
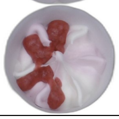
Ice cream cone	**Sauce mask**		
	CT(x,y)	**S(w,h)**	**POS(d,a)**
	(133,116)	(178,216)	(5,53°)
	(134,130)	(130,161)	(1,180°)
	(102,132)	(114,212)	(33,183°)

Table 4.11: The centroids, sizes and positions of the strawberry segmentation masks relative to the centers of ice cream cones of different qualities.

4.3.3 Variant 3 - Chocolate

Similar to the other variants, we begin with performing K-means segmentation on a good quality sample. Unfortunately, this does not work as well as with the previous variants. A noticeable difference between the sauce, nuts and ice cream was present with these variants. Due to the similar colour of the chocolate sauce and curls, they are segmented as one cluster. Even when increasing the K value, no clear distinction can be made. For instance, when using a K value of 10, the segmented image is compared to the original image in Figure 4.16.

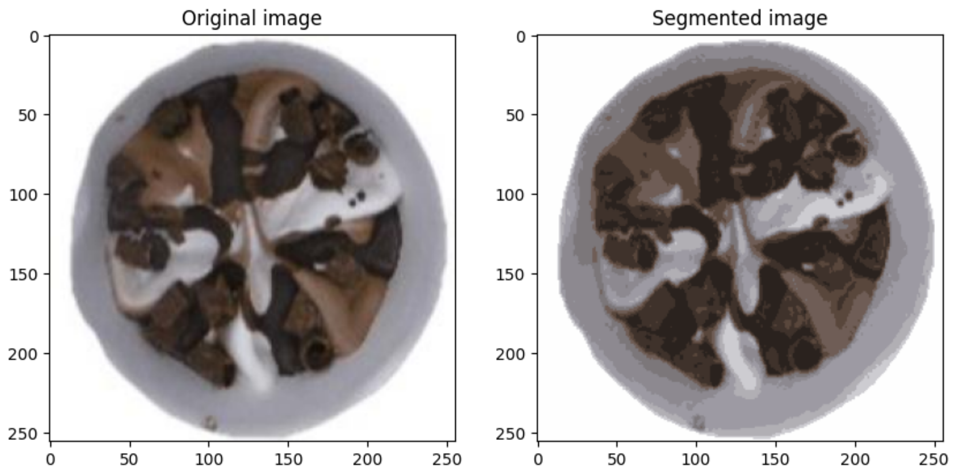


Figure 4.16: Segmented result of a good quality chocolate sample.

Due to the inability of segmenting the sauce and curls as separate masks, we only will look at chocolate ice cream cones which contain only either the sauce or the curls. In this way, we will be still able to compute the different features as with the previous variants. Table 4.12 shows the results. On the first ice cream cone, the sauce is spread well around the center. This is indicated by the large size of the mask, the distance of 7 pixels and the angle of 270°. On the second ice cream cone, the curls are also spread well around the center. The large mask and small distance confirm this observation.

Ice cream cone	Sauce or Curls mask		
	CT(x,y)	S(w,h)	POS(d,a)
	(130,142)	(189,190)	(7,270°)
	(136,119)	(163,198)	(6,9°)

Table 4.12: The centroids, sizes and positions of the chocolate segmentation masks relative to the centers of ice cream cones of different qualities.

5 Conclusion

In this report, we have explored the application of Convolutional Neural Networks (CNNs) for classifying the quality of ice cream cones. This use case is unique as, to our knowledge, no research has been conducted on it thus far. We have developed a framework that encompasses the creation of datasets and two different model approaches.

To train a CNN effectively, it is crucial to have an extensive dataset. In cases where the number of original raw images is limited, as in our case, data augmentation can be used to generate additional images. The two data augmentation transformations that were found to be relevant for us are rotations and adjustments in brightness.

Both model approaches have been validated on datasets for three different variants. The binary classification models of each variant have an F1 score higher than 90%. This also applies to the multi-label classification models. This performance is good considering that these models are trained on imbalanced datasets. It is remarkable that a decreasing F1 score can be observed in the following order: vanilla, strawberry and chocolate. The main reason for this is because the different toppings are easier to distinguish from the ice cream and each other. When we compare the performance of the binary classification model with that of the multi-label classification model of the same variant, we notice a few percent decrease in the performance of the multi-label classification model. Introducing more explainability in our models is associated with a decrease in performance.

We also explored the use of K-means segmentation to obtain topping segmentation masks. The K-means segmentation algorithm is able to create segmentation masks for the toppings of the vanilla and strawberry variants. Although these masks are relatively rough, they still provide additional insights into the position and distribution of the toppings. These insights can also be used as an additional validation of the model's predictions.

The classification models and the segmentation algorithm encounter challenges when dealing with the chocolate variant. This is primarily due to the similarity in colour between the chocolate curls and the chocolate sauce, particularly in the captured image data. To address this issue, it is crucial to employ a different type of lighting, both in type and colour. By doing so, it will be possible to capture the distinctions between the curls and sauce more effectively. Consequently, both the classification models and the K-means segmentation algorithm are expected to exhibit improved performance.

In future work, capturing image data from the production line could facilitate the training of a classification model capable of predicting the quality of ice cream cones in industrial environments. Additionally, the segmentation process can be linked with the multi-label classification model. To accommodate this integration, the labels should be expanded to include an *absent* label. This adjustment is necessary as the segmentation algorithm must be modified when an ice cream cone does not contain one of the two toppings.

References

[1] Crizzan Bel Colubio, Von Patrick Denorte, and Rica Panga. Quality classification of copra using convolutional neural network. Accessed 12-April-2023.

[2] Da-Wen Sun and Du Cheng-Jin. Segmentation of complex food images by stick growing and merging algorithm. May 2003. Accessed 12-April-2023.

[3] Puluz 40cm folding portable ring light usb photo lighting studio shooting tent box with 6 x dual-side color backdrops, size: 40cm x 40cm x 40cm(black). Available online at: https://www.puluz.com/p/PU5041B/PULUZ-40cm-Folding-Portable-Ring-Light-USB-Photo-Lighting-Studio-Shooting-Tent-Box-with-6-x-Dual-side-Color-Backdrops-Size-40cm-x-40cm-x-40cm-Black-.htm. Accessed 21-Mar-2023.

[4] Lumix digitale camera dmc-sz10. Available online at: https://www.panasonic.com/be/nl/consumer/archive-products/cameras-camcorders/dmc-sz10.html. Accessed 21-Mar-2023.

[5] Daniel Gatis. rembg, 2022. Available online at: https://github.com/danielgatis/rembg. Accessed 18-Apr-2023.

[6] Hossein Talebi and Peyman Milanfar. Learning to resize images for computer vision tasks, 08 2021. Available online at: https://arxiv.org/pdf/2103.09950.pdf. Accessed 18-Apr-2023.

[7] Why do we normalize image data?, 07 2022. Available online at: https://www.timesmojo.com/why-do-we-normalize-image-data/. Accessed 27-Apr-2023.

[8] Suorong Yang, Weikang Xiao, Mengcheng Zhang, Sehan Guo, Jian Zhao, and Furao Shen. Image data augmentation for deep learning: A survey, 04 2022. Available online at: https://arxiv.org/pdf/2204.08610.pdf. Accessed 27-Apr-2023.

[9] PyTorch. Torchvision.models. Available online at: https://pytorch.org/vision/0.11/models.html. Accessed 11-May-2023.

[10] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. 04 2015. Accessed 16-May-2023.

[11] What is the vgg 19 neural network?, 03 2022. Available online at: https://pytorch.org/vision/0.11/models.html. Accessed 16-May-2023.

[12] Kaming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. 12 2015. Accessed 11-May-2023.

[13] Mingxing Tan and Quoc V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks. 09 2020. Accessed 16-May-2023.

[14] OpenCV. transfer-learning, 05 2019. Available online at: https://learnopencv.com/wp-content/uploads/2019/05/transfer-learning.jpg. Accessed 16-May-2023.

[15] Eugenio Zuccarelli. Performance metrics in machine learning — part 1: Classification, 12 2020. Available online at: https://towardsdatascience.com/performance-metrics-in-machine-learning-part-1-classification-6c6b8d8a8c92. Accessed 22-May-2023.

[16] GeeksforGeeks. Image segmentation using k means clustering, 02 2023. Available online at: https://www.geeksforgeeks.org/image-segmentation-using-k-means-clustering/. Accessed 05-Sep-2023.