NHL STENDEN
computer vision & data science

Polymer Flake Detection Through Hyperspectral Imaging

NHL Stenden Professorship Computer Vision & Data Science

Benjamin Delaporte, Tjeerd van Gelder, Klaas van der Sluis

Supervisors: Klaas Dijkstra, Maya Aghaei Gavari

Abstract—While the amount of disposed plastic waste is constantly increasing, the issue of recycling plastics largely remains an open problem. This is partially due to the complexity of distinguishing between different types of plastics. With the use of the Specim FX17 camera which operates on the near-infrared region of the light spectrum we are able to capture information unseen to the human eye, this information will be analyzed with the use of deep learning. The aim of this project is to take further steps towards the automation of plastic waste sorting. This research compares the performance of three CNNs (YOLO-V3, EfficientDet and Faster-RCNN) to see which network performs the best when it comes to the detection and classification of polymer flakes. In this research we will perform object detection on NIR hyperspectral images with the use of convolutional neural networks. These hyperspectral images contain 224 spectral layers and thus contain a lot more information to work with that is normally unseen by the human eye. The method used in this project sets up a linear converter at the network input which converts n-channeled hyperspectral images to 3-channel images thus allowing the networks to perform object detection on hyperspectral images. Experiments were conducted to see if the size of the dataset influenced the performance of the networks, if adding data augmentation improved the results and if modifying parameters such as the learning rate and the patience of the networks increased their performance. In the end we concluded that all of the aforementioned experiments did indeed give quantitative results of the best performing network. The final conclusion is that the object detection approach is worth researching further to refine the current results and that it will be a good addition to the automation of plastic waste sorting.

Index Terms—: Plastic sorting, Hyperspectral images, Convolutional Neural Network, Yolov3, EfficientDet, Faster-RCNN, Object detection

1 Introduction

In 1862, during the World's fair in London, Alexander Parkes presented the first plastic in the world: the parkesine. With the growth of the chemical industry in the late 1940's the production of new synthetic plastics grew immensely. Well known plastics that were produced at this time are polystirene (PS), polyvinyl chloride (PVC) and polyethylene (PE). Nowadays, there are numerous types of plastics in addition to PS, PVC and PE such as Polypropylene (PP), Polyethylene terephthalate (PET) and Polyactic acid (PLA).

The discovery of plastic made a real change to everyday life. Plastics are strong, easy to mold and cheap to produce. With this in mind they are mainly used for packaging and manufacturing of single-use products [1] like the ones seen in Figure 1. It is in this context that 9.2 billion tons of plastics have been produced since the 1950's [2]. With this information it can be expected that the amount of trashed plastic has also been increasing.

After being used and thrown in the bin, plastics are mainly landified or burned to recover energy. However, these two options are damaging to the environment: the first one pollutes water and soil while the second one releases greenhouse gases [4]. Plastics are sometimes recycled but this is a cumbersome process. Because the plastic waste is unsanitary it has to be washed. To ensure that every nook and cranny of the waste gets cleaned the plastics are ground into small flakes to facilitate their cleaning process. After that, the different flakes need to be sorted based on the polymer type because recycling is solely possible for a few categories of plastic. This task is very difficult because plastic comes in different types of materials, shapes, colours, sizes and flexibility. Given the aforementioned characteristics it is impossible to recognize the type of polymer with the naked eye. Worldwide less than 10 percent of all plastic ever produced has been recycled [2]. Considering that the end-of-life of plastics is a real threat for the environment, it is urgent to take up this challenge in the next decades.

Fig. 1. Single-use plastics. [3]

Our goal with this work is to implement object detection. Detection as defined in [5] is the task of "Locating the presence of objects with a bounding box and types or classes of the located objects in an image.". The coordinates and labels from the boxes of the detected objects will be used by a sorting machine to sort the polymers accordingly. This will take place near the end of the plastic recycling process when the plastics have already been turned into flakes.

- *Benjamin Delaporte is a Mechatronics Engineering student at INSA Hauts de France Engineering school, E-mail : bdelap1_@etu.uphf.fr.*
- *Tjeerd van Gelder is a Software Engineering student at the NHL Stenden University of Applied Sciences, E-mail: tjeerd.van.gelder@student.nhlstenden.com .*
- *Klaas van der Sluis is a Mechatronics Engineering student at the Hanzehoogeschool Groningen, E-mail: klaas.sluis@student.nhlstenden.com .*
- *Klaas Dijkstra is an Associate Lector at the NHL Stenden Professorship Computer Vision & Data Science, klaas.dijkstra@nhlstenden.com.*
- *Maya Aghaei Gavari is a Researcher at the NHL Stenden Professorship Computer Vision & Data Science, E-mail: maya.aghaei.gavari@nhlstenden.com.*

In this research, we intend to solve the problem of sorting polymers using deep learning based software with Convolutional Neural Networks (CNN). These CNN's are "deep learning neural networks designed for processing structured arrays of data such as images" [6]. The sorting operations will be done using Hyperspectral imaging (HSI) which is "a technique that analyses a wide spectrum of light instead of just assigning primary colours (red, green, blue) to each pixel" [7]. Our main reason to use HSI in this project is it's ability to distinguish between the types of polymer through imaging that the human eye is incapable of.

1.1 Research questions

1.1.1 Main question

- Can polymer flake detection on hyperspectral images aid in the automation of plastic waste sorting?

1.1.2 Subquestions

Our main question can be summarized in finding a solution to three principle questions:

- How can we apply object detection on hyperspectral images?
- What is the best object detection approach when it comes to plastic flakes?
- How can we combine segmentation and object detection within our CNN?

2 State of the art

Previous studies on the plastic sorting with object detection topic have been done in [8] [9], these studies can be divided into the subtopics below:

- Computer Vision
- Hyperspectral Imaging
- Convolutional Neural Networks

These subtopics are also related to the current project of this paper since computer vision is also used as a means to analyse the data. This data consists of images containing 224 hyperspectral channels thus hyperspectral imaging is an important factor for this project. Since a form of deep learning is necessary to get understandable results from all the data contained in these images, convolutional neural networks are used as a base for the software.

In this chapter we will explain more in depth on the state of the art regarding these topics and how and why they are used to approach this problem.

2.1 Computer Vision

Computer vision is a scientific field which deals with several topics such as image processing and analysis. The goal of this discipline is to enable computers to process a visual content similar to the way that humans do and provide an output related to its task. This field includes operations like object detection, instance segmentation and classification.

2.1.1 Object Detection

Object detection is the process of detecting the objects of interest and drawing bounding boxes around them in a given image. This is done by taking the shape, color, and orientation in consideration as variables for an object detection algorithm. Object detection can be used to automate certain kinds of projects, for example pick and place solutions.

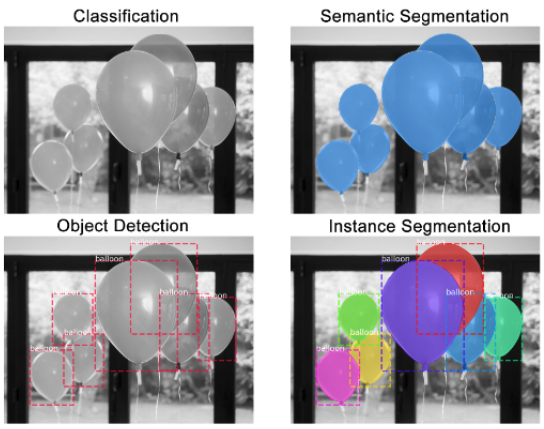

Fig. 2. Different tasks in computer vision. [10]

2.1.2 Instance Segmentation

Instance segmentation is used to make the image in question easier to analyse, for example by labeling them. Segmentation often simplifies the image by partitioning distinctive parts in the image into multiple segments which makes the data easier to understand.

2.2 Hyperspectral Images

Similar to RGB images, hyperspectral images are 2D spatial images. What differs between these two types of images is their spectral range. While regular RGB images cover only three channels (each pixel has three values corresponding to the red, green and blue wavelength), hyperspectral images cover many more spectral channels (224 channels for a specim FX-17 camera). Each layer of a hyperspectral image shows the reflectance of the object shot to a certain wavelength. A hyperspectral camera generates a 3D image which can be referred to as a hyperspectral cube (Figure 3).

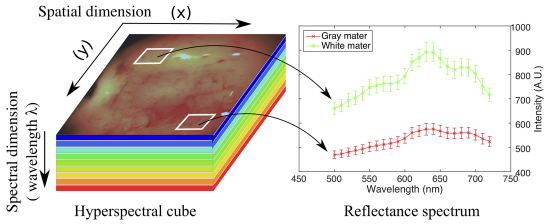

Fig. 3. Hyperspectral cube: each pixel from the spatial dimension. (Plane(x,y)) gives spectrum information [11]

Some researches show that hyperspectral images could be used to distinguish objects which seems to be the same but in fact are made of 2 different materials (Fig: 4). This is because hyperspectral images contain more details of an object, due to the ability to reflect light at more wavelengths than only RGB. [8]

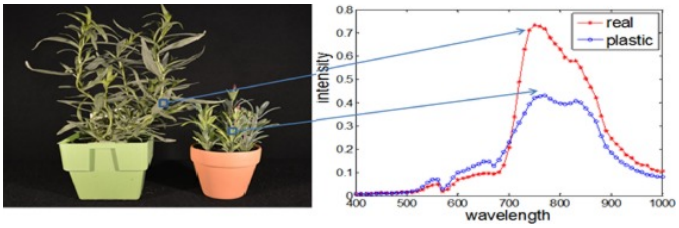

Fig. 4. Pixels from plastic plant and natural plant have different spectrum reflectance. [8]

2.2.1 Object Detection on Hyperspectral Images

According to [8], in order to detect objects with material information, the material composition and proportion need to be analyzed from the hyperspectral images first. For this they used a method called hyperspectral unmixing. From the result of the unmixing they calculated the spectral-spatial distribution so that the object is measured based on the variance of spectral distribution in the spatial domain. After that, the mean is taken from this mapping and two other mappings, namely the spectral Euclidean distance mapping and the spectral angle distance mapping. The mean of these three mappings gives results that can be thresholded afterwards so the objects can be detected individually.

2.3 Convolutional Neural Networks

Convolutional neural networks (CNN) are state-of-the-art algorithms used for computer vision related tasks as for example image classification and object detection. Some of the frontrunners in this topic are companies like Google and Microsoft who have active research groups exploring better CNN architectures. CNN's are a category of Neural Networks that have proven very effective in areas such as image recognition and classification. CNN's have been successful in identifying faces, objects and traffic signs apart from powering vision in robots and self-driving cars [12]. These CNN's can be divided into multiple stages or layers which can consist of convolutional, non-linear and subsampling layers.

So far architecture modifications can be categorized in seven different categories; spatial exploitation, depth, multi-path, width, feature-map exploitation, channel boosting, and attention-based CNNs. (Figure 5).

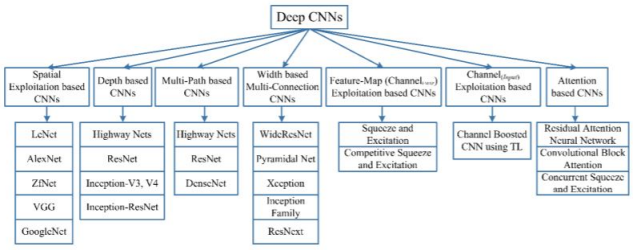

Fig. 5. Taxonomy of deep CNN architectures [13]

Below an explanation is given on some of these categories:

Spatial Exploitation based CNNs:
As convolutional operation considers the neighborhood (locality) of input pixels, therefore different levels of correlation can be explored by using different filter sizes. Different sizes of filters encapsulate different levels of granularity; usually, small size filters extract fine-grained and large size extract coarse-grained information [13].

Depth based CNNs:
Network depth has played an essential role in the success of supervised training. Theoretical studies have shown that deep networks can represent certain classes of function more efficiently than shallow architectures [13].

Multi-Path based CNNs:
Multiple paths or shortcut connections can systematically connect one layer to another by skipping some intermediate layers to allow the specialized flow of information across the layers [13].

In 2019, the most recent architecture was developed by Google and is known as EfficientNet. An efficiency chart of this network compared to other networks can be seen in the picture below.

The EfficientNet model shows high accuracy on datasets like ImageNet with an accuracy of 88.5% and CIFAR-100 with an accuracy of 91.7% [14] as seen in (Figure 6) and thus for these proved capabilities EfficientNet is a good candidate model for our research and is therefore used to detect plastic flakes.

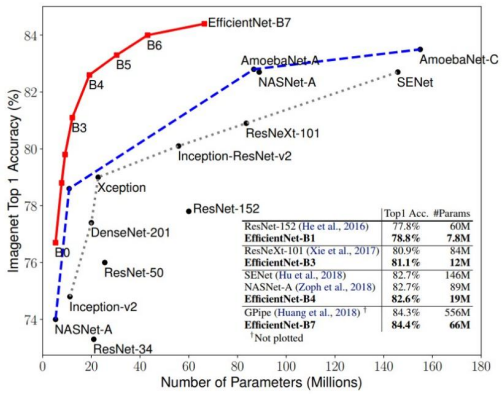

Fig. 6. Evolution of accuracy of different neural networks. [14]

3 Materials and Methods

In this section the materials and methods are described that are used in this research.

3.1 Computing Hardware

This research was done on a Virtual Machine (VM), below in Table 1 is a list of specifications regarding this VM.

Table 1. Computing hardware

CPU	Intel i9-7960X 2.8GHz with 12 cores
RAM	16 GB
GPU	1x NVIDIA RTX 2070 SUPER 8 GB GDDR6 memory and 2560 CUDA cores
OS	Debian 10.1 "Buster"

3.2 Camera

For this research a Specim FX-17, which can be seen in figure 7, was used to acquire the hyperspectral images. This near-infrared camera has a range between 900nm and 1700nm. This makes the camera ideal for acquiring data that differentiates between different chemical compositions. This camera is known as a line-scan camera as it scans one line of 640 pixels at a time instead of taking a full picture. This means that in order to get data of a full image the object in question needs to be passed underneath the camera at a constant speed that is in sync with the shutter-time of the camera. Eventually, the so called datacubes obtained from this camera have the dimensions of 1500 x 640 pixels and 224 spectral channels.

Fig. 7. Specim FX-17 camera [15]

3.3 CNN Architectures

In this section all the different architectures of neural network that are used in this research are presented.

3.3.1 YOLO-V3

Third version of the You Only Look Once architecture [16], YOLO-V3 is a fully convolutional network used in object detection application. This method is a suitable choice for real-time detection, without loss of too much accuracy.

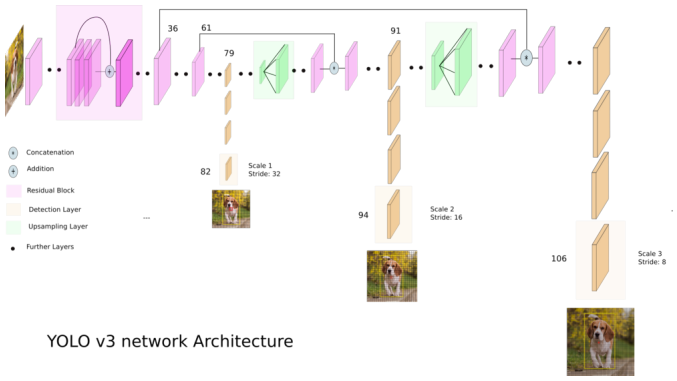

Fig. 8. YOLO-V3 network Architecture [17]

This neural network is based on the Darknet-53 architecture which is a 53 layer convolutionnal neural network trained for classification on Imagenet. This backbone is stacked on a 53 layer neural network which will be trained for object detection.

3.3.2 EfficientDet

EfficientDet [18] is an efficient object detection network which employs as backbone the EfficientNet network. In this project, EfficientDet architecture was chosen because of its high accuracy score as shown in the introduction (Figure 6) . Following this backbone, several weighted BiFPN's (Bi-directional Feature Pyramid Network's) are used to predict and classify the boxes. (Figure 9)

3.3.3 Faster-RCNN

Region-based Convolutional Neural Network (RCNN) ,see Figure 9, is a method that takes a selective amount of regions in an image as region proposals and feeds them to the CNN. Faster RCNN is the latest version of this method that is better optimized to do this process as fast as possible and thus lowering computational cost.

For this research Faster-RCNN was used to train several models and test these models on the dataset. However the results from Faster-RCNN are not promising enough to consider this network a success yet and further research needs to be done to find out how to improve this network, more on Faster-RCNN and the experiments and results of this network can be found in appendix A.

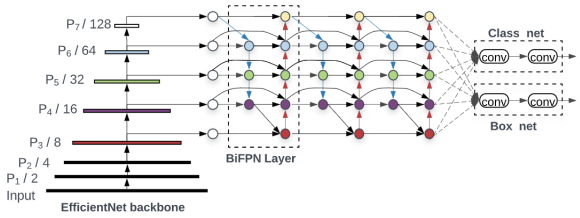

Fig. 9. EfficientDet Architecture [18]

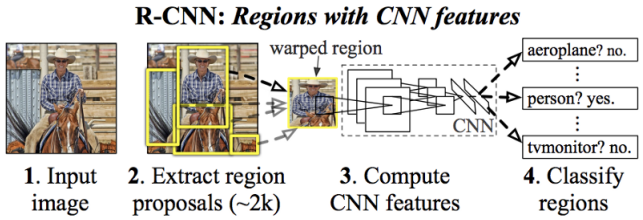

Fig. 10. R-CNN [19]

3.4 Modifications to the CNN architectures

All these architectures are designed for RGB pictures: they only accept pictures with three spectral channels. In the current state, hyperspectral images cannot be processed by these networks as they have 224 channel. Thus, to utilize the potentials of these networks in our project, it was necessary to adapt them to process hyperspectral images. For that, we added a linear converter as the first layer to the networks which converts pictures with 224 channels into three channel images as shown in Figure 11, here each layer of the hyperspectral image (on the left) is converted to RGB to eventually get a new image with three channels (on the right). This method allows us to use pretrained object detection neural networks.

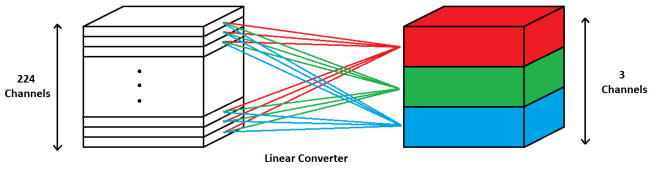

Fig. 11. Linear Converter

To add this linear layer as the first layer of the object detection neural network, it is necessary to create a new network. In this new network, are initialized the linear converter and the pretrained object detection network. Then it is necessary to create a forward function process the input hyperspectral cube in the good order. One trick, with the forward function is the permutation of the axes of the input data. Indeed, hyperspectral cubes come in batch with the following shape (batch size, channels, height, width). The first dimension of the linear converter needs to be the same than the last one of the input data. In order to do that, before and after passing through the converter, the axis of the input data need to be permuted.

Below is presented the source code of this new network for the YOLO-V3 object detector :

```
class MyNetwork(nn.Module):
    def __init__(self, c, network):
        super(MyNetwork3, self).__init__()
        self.network = network
        self.converter = nn.Linear(in_features=c,
                                   out_features=3,
                                   bias=False)
    def forward(self, x):
        x=x.permute(0,2,3,1)
        conversion = self.converter(x)
        conversion=conversion.permute(0,3,1,2)
        new_model=self.network(conversion)
        return(new_model)

def hs_yolov3(num_classes: int):
    channels=224
    network=create_yolov3_pt(num_classes=num_classes)
    model=MyNetwork3(c=channels, network=network)
    return model
```

Furthermore, the dataloader that was used with these architectures originally did not accept hyperspectral images. The dataloader had to be modified so it could read '.npz' files which is the file type for the hypercubes.

This was done by adding the '.npz' extension to the list of loadable images and creating a corresponding function that would extract the hyperspectral cube as an array from the file. In the code fragment below you can see the changes made.

```
class LoadImage():
    loaders = {'.png': load_image,
               '.jpg': load_image,
               '.npy': load_npy,
               '.npz': load_npz}

def load_npz(filename:str):
    npz = np.load(filename)
    return npz['scan']
```

The '.npz' extension was added to the loaders giving the networks the ability to read those file types. The '.npz' file contains more data since these hypercubes consist of 224 layers of information compared to the three layers that are in normal RGB images. A corresponding function had to be created to make sure that only the image data was extracted from the file. Since these '.npz' files contain a lot of data, the dataloader uses a tiling method which divides the image in n*n squares. These tiles are processed seperatly by the network. This was necessary because loading an untiled hyperspectral image required more memory on the GPU then we had available which would cause the software to crash.

3.5 Dataset

For this research a dataset was created with the Specim FX-17 camera. For the dataset six different polymers are used namely: PP, PET, PE, PLA, PVC, PS. These polymers come in the form of flakes and are placed on a conveyor belt that moves underneath the camera.

3.5.1 Image Structure

For the training and validation images, fourteen sets were made, each set consists of flakes from three of the six polymer types. For example a training set can consist of PET, PE and PP flakes or other combinations of three out of the six polymers. For the testing images three sets were made, each set consists of flakes from all six polymer types.

For each polymer type we have used different flake samples to increase the diversity of our dataset. The use of different samples for each type of polymer is an essential step in providing a balanced dataset and to make sure the network learns to detect the polymer and not just a specific sample of said polymer.

For the purpose of this research a large dataset and a smaller subset are created. This is done to see to what extent the size of the dataset influences the performance of the networks. The large dataset consists of images from all fourteen sets while the smaller subset consists of images from eight of the sets.

The Tables 2, 3, 4 and 5 show the amount of samples we used from each polymer and in how many images the polymer is represented.

Table 2. Training and validation large dataset

Polymer Type	Sample count	Image count
PET	5	17
PP	5	20
PS	5	18
PE	5	19
PLA	2	11
PVC	2	11

Table 3. Testing large dataset

Polymer Type	Sample count	Image count
PET	3	6
PP	3	6
PS	3	6
PE	3	6
PLA	2	6
PVC	2	6

Table 4. Training and validation small dataset

Polymer Type	Sample count	Image count
PET	4	8
PP	3	10
PS	3	9
PE	3	9
PLA	2	9
PVC	2	9

Table 5. Testing small dataset

Polymer Type	Sample count	Image count
PET	2	4
PP	2	4
PS	2	4
PE	2	4
PLA	2	4
PVC	2	4

This resulted in a large dataset containing 32 images used for training/validation and six images for testing and a smaller subset containing 18 images for training/validation and four images for testing.

3.5.2 Annotating Method

In order to train CNN architectures to perform flake detection, we need to annotate images in our dataset. The labeling software of our choice only accepts RGB images as input, thus, hyperspectral images of our dataset first need to be converted into RGB images which are then only used for annotation. The method used to convert the images is explained here:

- The hyperspectral image is sliced following the channel axis into three groups of grayscale pictures.
- For each group of grayscale picture, the mean of each pixel is taken to create a new grayscale picture.
- These new gray scale picture are used as the R,G and B channels.

Afterwards, the flakes are annotated with the use of labelimg open source software [20]. This software is a tool with which you can draw bounding boxes on your images and label them as shown in Figure 12, here it can be seen that on the left the source image is annotated by drawing rectangles around them and on the right every box is getting a label with the corresponding polymer type.

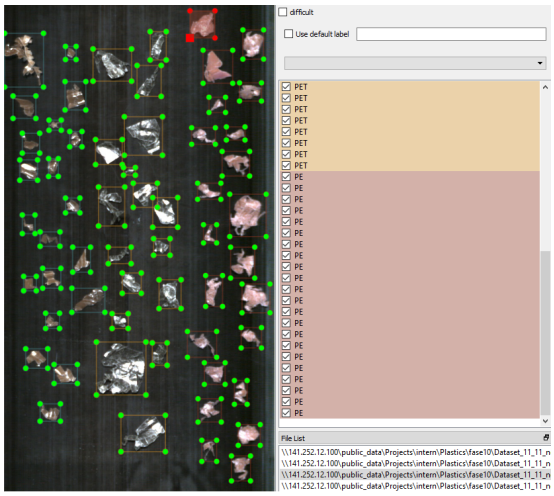

Fig. 12. An image of the dataset being annotated. On the left you see all the polymer flakes and their given bounding boxes. On the right is a list of all the bounding boxes and the class they were assigned.

3.5.3 Data Augmentation

In order to expand the dataset it was enhanced through the use of image augmentation. The augmentations that were used are image rotation and image flipping.

When rotated a rotation angle was generated between 0 and 45 degrees, after which the rotation was applied to the image with the center of the image being the center of rotation. The flipping of the image could happen on the Y and/or X-axis. These different augmentations combined greatly enlarged the amount of data the networks were trained with.

The difficulty with these augmentations was the fact that the bounding boxes that came with the image also had to be transformed. To make sure this worked correctly the augmentations were visualized during some testing. Figure 13 shows the successful flipping along the y-axis.

4 Experiments & Results

This section will give a description of the experiments we executed to answer our research questions. The experiments will be run with the dataset described in section 3.5.1.

The two questions we will be answering are as followed:

- How can we apply object detection on hyperspectral images?
- What is the best object detection approach when it comes to plastic flakes?

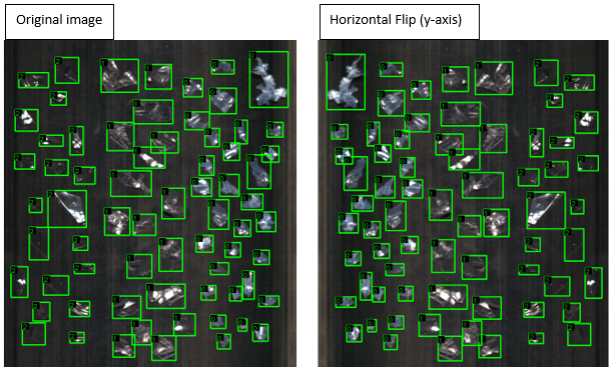

Fig. 13. Image flipped along the y-axis

The hyperparameters used for these experiments are seen in Table 6. Experiment 1 is an excepetion to the amount of classes because for this experiment we train our network on a single class. For experiment 5 there is an exception on the patience because there we gradually increase the patience during training.

Table 6. Hyperparameters

Batch size	1
Num tiles	6
Learning rate	0.001
Patience	8
Num classes	6
Epochs	150

We will be using the following metrics to evaluate each models: Precision, Recall, F1-score, mAP and False Discovery Rate. These metrics and their utility are presented below:

The Precision is a metric which gives us data about the ability of a model to identify only the relevant data points. Precision is computed dividing the TP by the sum of TP and FP as seen in equation 1. The closer the precision is to 1, the better the model is at classifying the flakes correctly.

$$Precision = \frac{TP}{TP+FP} \tag{1}$$

The recall metric shows us the ability of the model to find all the relevant elements in the dataset. Recall is computed dividing the TP by the sum of TP and FN as seen in equation 2. The closer the recall is to 1, the better the model is at detecting all the flakes.

$$Recall = \frac{TP}{TP+FN} \tag{2}$$

The F1-score is computed using the Precision and Recall as seen in equation 3. Precision and recall are average weighted in this metric. This metric is a good indicator of the performance of the model because it takes into account both the False Positives and the False Negatives. Like the precision and the recall, the closer the F1-score is close to 1, the better the model is.

$$F1Score = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{3}$$

The False Discovery Rate gives data about the rate of false positives among the relevant elements. The False Discovery Rate is computed dividing the FP by the sum of FP and TP as seen in equation 4.

$$FDR = \frac{FP}{FP+TP} \tag{4}$$

The use of a confusion matrix is an useful tool to know how well a model is classifying the objects it predicts. The number of correct and incorrect predictions are summarized with count values and broken down by each class. Each row of the confusion matrix is given to the actual class while the columns shows where the classes in which they were classified. Consequently, this matrix gives information on which classes are well classified or not.

The research questions can be answered by conducting the following experiments:

- How do the networks score when asked to detect flakes as a single category?
- How do the networks score when asked to not only locate flakes, but also classify them?
- Does the size of the dataset influence the results?
- Does data augmentation improve the performance of the networks?
- Can the results be further improved by modifying the parameters of the scheduler?

4.1 YOLO-V3 and EfficientDet Experiments

In this subsection the experiments are described that were done with the YOLO-V3 and EfficientDet models, also for each experiment the results will be explained.

4.1.1 Experiment 1: Flake Localization

For the first experiment we will be changing the class of all the annotations so the networks will be trained with one class. This will tell us if the networks are able to localize polymer flakes in hyperspectral images.

In Table 7 the results can be seen for this experiment.

Table 7. Metric scores per network on a single class

	YOLO-V3	EfficientDet
Precision	0.95	0.82
Recall	0.82	0.81
F1-score	0.88	0.81
mAP	0.80	0.79
False Discovery Rate	0.05	0.18

In this table we can see the metrics of the two architectures for the flake detection task. The precision and recall are close to 1. Thus, the F1-Score for the two models is close to 1. These metrics mean the results are good and allows flake detection. Moreover the False Discovery Rate is close to 0 : that means the model doesn't make too much mistakes when it predicts a flake.

In figure 14 we can see that the flakes are almost detected by the EfficientDet Network: These results look very promising in solving the problem.

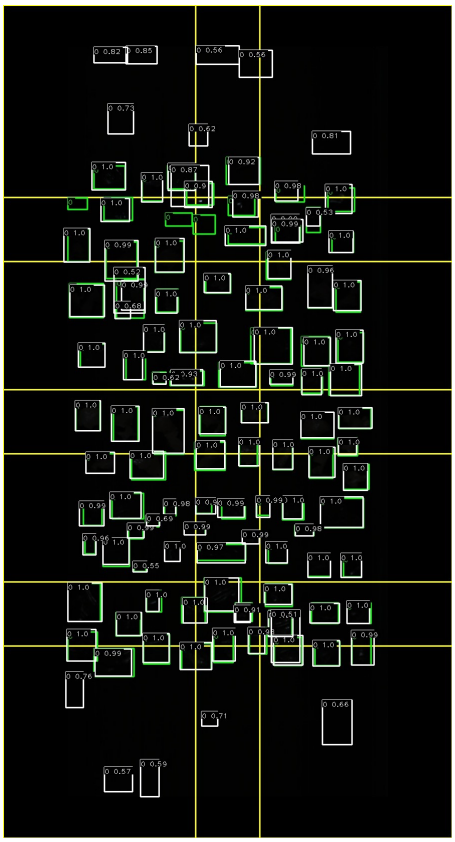

Fig. 14. Flakes Detection with EfficientDet Network. The green boxes are the ground truth, the white boxes are the predictions and the yellow lines are the outlines of the used tiling.

4.1.2 Experiment 2: Flake Localization and Classification on the small dataset

For this experiment, we will be using the images from the smaller subset with their original annotations so the network will be trained to identify flakes of all polymer types.

In Table 8 the results can be seen for this experiment.

Table 8. Overall Metric scores per network on multiple classes with the smaller subset

	YOLO-V3	EfficientDet
Precision	0.12	0.45
Recall	0.27	0.11
F1-score	0.16	0.18
mAP	0.03	0.06
False Discovery Rate	0.88	0.55

All of these metrics are extremely low, especially when compared to the results of experiment 1.

4.1.3 Experiment 3: Flake Localization and Classification on the larger dataset

For this experiment we will run the test from experiment 2 with the large dataset after which we will compare the metrics of both experiments.

In Tables 9, 10 and 11 are the results from this experiment.

Table 9. Overall Metric scores per network on a multiple classes

	YOLO-V3	EfficientDet
Precision	0.69	0.44
Recall	0.29	0.40
F1-score	0.41	0.42
mAP	0.21	0.25
False Discovery Rate	0.31	0.56

Table 10. Metric scores per class for the Yolov3 network

Class	Precision per class	Recall per class	F1 Score per class
PE	0.6	0.04	0.06
PET	0.87	0.49	0.63
PLA	0.52	0.50	0.50
PP	0.58	0.24	0.34
PS	0.58	0.14	0.22
PVC	0.59	0.31	0.40

Table 11. Metric scores per class for the EfficientDet network

Class	Precision per class	Recall per class	F1 Score per class
PE	0.55	0.40	0.46
PET	0.36	0.27	0.31
PLA	0.43	0.46	0.44
PP	0.34	0.36	0.35
PS	0.48	0.56	0.52
PVC	0.53	0.39	0.44

These results show a big performance increase when compared to those in experiment 2, however they are still worse when compared to the ones we got in experiment 1. The metrics we get from this model are very low : only around 0.4 for the F1-Score for the 2 architectures. Given the fact these results were better then those of experiment 2 it is very likely that further increasing the amount of images in our dataset will yield even better results, another possibility is to increase the amount of training itterations.

4.1.4 Experiment 4: Comparing Networks with/without Data Augmentation

For this experiment we will be adding data augmentation to the tests ran in experiment 3 after which we will compare them with each other. In Tables 12, 13, 14, 15 and 16 are the results from this experiment.

Table 12. Overall Metric scores per network on multiple classes with data augmentation

	YOLO-V3	EfficientDet
Precision	0.53	0.53
Recall	0.42	0.43
F1-score	0.46	0.47
mAP	0.23	0.32
False Discovery Rate	0.47	0.47

Table 13. Metric scores per class for the YOLO-V3 network

Class	Precision per class	Recall per class	F1 Score per class
PE	0.73	0.44	0.54
PET	0.46	0.06	0.11
PLA	0.40	0.47	0.43
PP	0.18	0.33	0.23
PS	0.64	0.55	0.59
PVC	0.18	0.08	0.11

Table 14. Confusion matrix for the flakes detected with YOLO-V3 network, the left column is the ground truth

	PE	PET	PLA	PP	PS	PVC
PE	82	0	2	0	0	0
PET	30	38	2	22	0	0
PLA	42	26	30	6	0	0
PP	22	3	0	59	0	0
PS	0	0	0	9	67	1
PVC	1	21	12	12	0	29

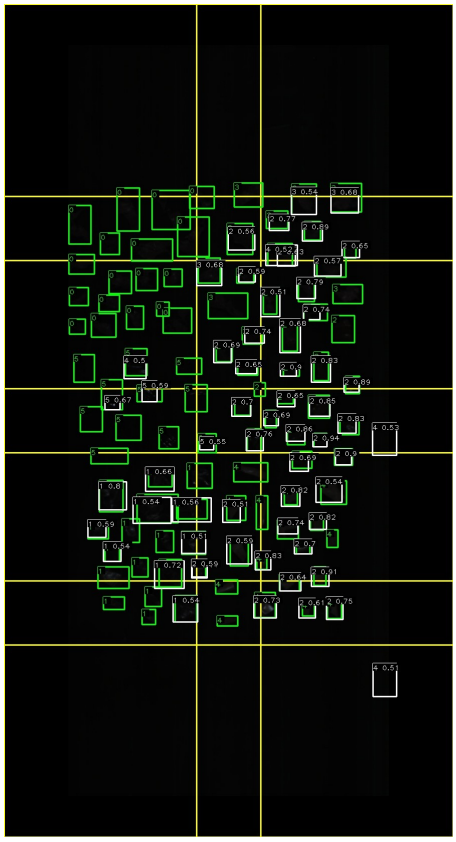

Fig. 15. Flake Detection and Classification with YOLO-V3 network. The green boxes are the ground truth, the white boxes are the predictions and the yellow lines are the outlines of the used tiling.

The performace increased but the metrics are still very low. According to the overall metrics the networks perform better with data augmentation. The F1-score improved for both networks, from 0.41 to 0.46 for YOLO-V3 and 0.42 to 0.47 for EfficientDet. Furthermore, for both networks the results per class begin to be very satisfying for some classes : we can see also see that in the confusion matrices. For example, YOLO-V3 model almost perfectly classified PE. Concerning the EfficientDet model, so far it is the PP and PS that are reasonably classified even if there is still some mistakes made. One thing to note is the False Discovery Rate which is still high for

Table 15. Metric scores per class for the EfficientDet network

Class	Precision per class	Recall per class	F1 Score per class
PE	0.86	0.28	0.42
PET	0.46	0.28	0.35
PLA	0.53	0.33	0.41
PP	0.41	0.75	0.53
PS	0.58	0.79	0.67
PVC	0.69	0.25	0.37

Table 16. Confusion matrix for the flakes detected with EfficientDet network, the left column is the ground truth

	PE	PET	PLA	PP	PS	PVC
PE	24	0	3	38	0	0
PET	0	27	7	35	9	0
PLA	2	12	37	14	4	2
PP	1	10	1	65	0	1
PS	0	1	10	0	62	0
PVC	0	7	7	1	4	28

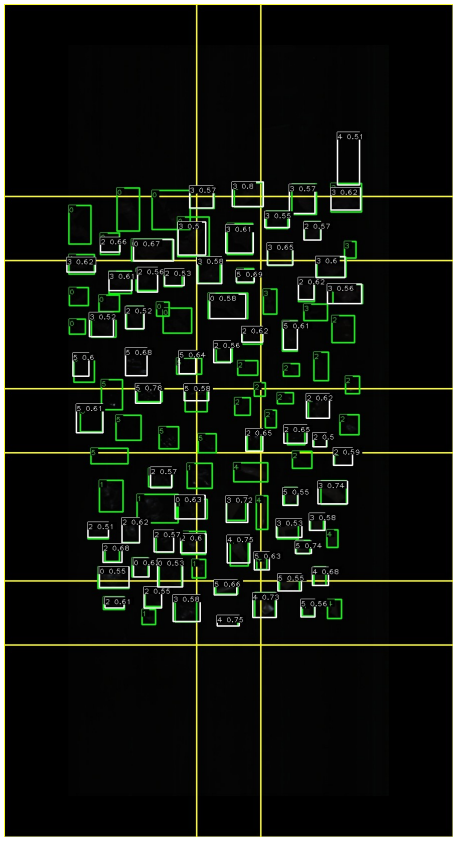

Fig. 16. Flake Detection and Classification with EfficientDet network. The green boxes are the ground truth, the white boxes are the predictions and the yellow lines are the outlines of the used tiling.

both networks and did not decrease a lot.

4.1.5 Experiment 5: Modifying the parameters of the scheduler

After several training steps, it appeared the learning rate was reducing too fast. The optimal moment to decrease the learning rate is when the the model starts overfitting. But looking to the loss curves, after the first reduction the model was not overfitting yet and the learning rate already reduced again. At this moment it was decided to try to modify the hyperparameters again in order to verify this hypothesis. The results below were obtained by modifying some parameters in the scheduler: increasing the patience, and adding a cool down time on the learning rate reduction which stabilized the loss curves.

Table 17. Overall metric scores per network

	YOLO-V3	EfficientDet
Precision	0.68	0.65
Recall	0.60	0.70
F1-score	0.63	0.68
mAP	0.47	0.61
False Discovery Rate	0.32	0.35

Table 18. Metrics scores per class for the YOLO-V3 network

Class	Precision per class	Recall per class	F1 Score per class
PE	0.92	0.91	0.91
PET	0.66	0.88	0.75
PLA	0.96	0.42	0.59
PP	0.50	0.73	0.59
PS	0.59	0.62	0.60
PVC	0.67	0.15	0.25

Table 19. Confusion matrix for the flakes detected with YOLO-V3 network, the left column is the ground truth

	PE	PET	PLA	PP	PS	PVC
PE	77	0	0	0	0	0
PET	0	83	0	0	0	0
PLA	0	25	46	0	18	0
PP	0	0	0	65	0	0
PS	0	0	1	0	48	0
PVC	0	0	0	38	1	16

Table 20. Metrics scores per class for the EfficientDet network

Class	Precision per class	Recall per class	F1 Score per class
PE	0.78	0.99	0.87
PET	0.61	0.74	0.67
PLA	0.59	0.43	0.50
PP	0.56	0.87	0.69
PS	0.75	0.97	0.84
PVC	0.58	0.35	0.44

Table 21. Confusion matrix for the flakes detected with EfficientDet network, the left column is the ground truth

	PE	PET	PLA	PP	PS	PVC
PE	84	0	0	0	0	0
PET	2	66	7	9	8	1
PLA	0	30	49	24	0	5
PP	4	1	0	76	0	0
PS	0	0	0	1	76	0
PVC	0	11	17	13	1	31

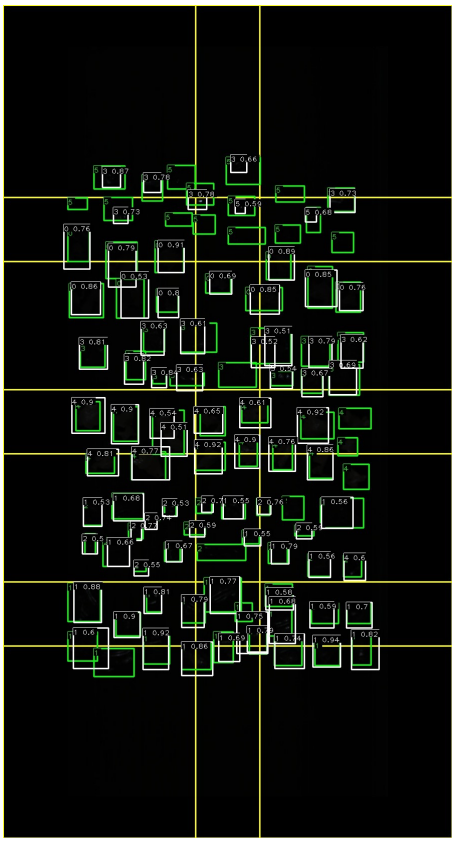

Fig. 17. Flake Detection and Classification with YOLO-V3 network after modifying the parameters. The green boxes are the ground truth, the white boxes are the predictions and the yellow lines are the outlines of the used tiling.

According to the different metrics calculated, modifying these parameters seem to enhance the results. Precision, recall, and F1-Score improved for both networks. The False Discovery Rate reduced a lot: the networks are making less errors in their predictions. As we can see in the resulting images, figure 17 and figure 18, the predictions look good, both networks localize almost all the flakes found in the pictures. According to the confusion matrices of these models, tables 19 and 19, they were also able to classify the flakes quite well. For YOLO-V3 4 out of 6 types of polymers (PE, PP, PET and PS) are classified reasonably. For EfficientDet, it is only 3 out of 6 (PE, PP and PS).

Fig. 18. Flake Detection and Classification with EfficientDet network after modifying the parameters. The green boxes are the ground truth, the white boxes are the predictions and the yellow lines are the outlines of the used tiling.

5 Discussion, Conclusion & Future Work

In this chapter the results of the experiments are discussed and from this a conclusion is given. After the discussion and conclusion section some remaining points will be mentioned that could be done in a future research.

5.1 Discussion

In this research multiple experiments were conducted with different CNN's and these were compared with each other in order to answer our research questions: "How can we apply object detection on hyperspectral images?" and "What is the best object detection approach when it comes to plastic flakes?"

The object detection was made possible by modifying the existing networks with a linear layer which allowed the networks to work with hyperspectral cubes as presented in section 3.5.

The following is with regards to two networks: YOLO-V3 and EfficientDet.

The second question will be answered by looking at the results of the experiments. When looking at the results of our first experiment, most notably the F1-scores which were 0.88 and 0.81 for YOLO-V3 and EfficientDet respectively tell us that the networks are capable of localizing flakes in hyperspectral images.

To expand on this we conducted the second experiment on the smaller dataset in which we asked the networks to detect and classify the flakes. This drastically lowered the performance of the networks which led to the third experiment where we gave the networks the whole dataset. These metrics were still low when compared to those of experiment 1 which is as expected since the models now need to make assumptions for the different classes as well, which would require more data were the metrics to stay the same. In experiment 4 we add data augmentation to the dataloader as discussed in section 3.6.3. This slightly improved the results. In table 22 and 23 you can see the steadily performance increase of both networks.

Table 22. Performance increase YOLO-V3

Metric	expt 2	expt 3	expt 4
F1-score	0.16	0.41	0.46
mAP	0.03	0.21	0.23

Table 23. Performance increase EfficientDet

Metric	expt 2	expt 3	expt 4
F1-score	0.18	0.42	0.47
mAP	0.06	0.25	0.32

The biggest performance increase came from experiment 5 where the parameters of the scheduler were modified. When comparing the metrics with those of experiment 4 all the scores increased significantly which was a pleasant result, most notable was the increase of the F1-Scores which increased by 35% for both networks, from 0.46 to 0.63 for YOLO-V3 and 0.47 to 0.63 for EfficientDet. This performance increase can also be seen when comparing the confusion matrices of these experiments as there is a lot less misclassification. One thing to note in these matrices is that the classification for PLA and PVC is still a problem, which persisted through all experiments. This is likely due to the fact that the amount of samples and images of those two polymers is underrepresented in the dataset as can be seen in Table 2.

5.2 Conclusion

With the findings of the discussion we can conclude our research questions. Object localization and classification on hyperspectral images can be achieved by modifying existing networks to accept hyperspectral images through an added layer which converts the n-channeled image to a 3-channeled image accepted by the network.

The networks showed improving results when data-augmentation was applied and when the scheduler parameters were modified. With the exception of Faster-RCNN, which did improve with the same methods but is still nowhere close to the other two models in terms of performance. Furthermore, they all performed better when given a larger dataset, while still underperforming with regards to PLA and PVC, which made it apparent that the performance of the networks was partly limited by the fact that the polymers PLA and PVC were underrepresented.

This leads us to conclude that plastic flake localization and classification through hyperspectral imaging can be a promising addition to any plastic sorting process that works with flakes when researched and refined further.

5.3 Future Work

The third and last research question, "How do we combine segmentation and object detection within our CNN?" has been left unanswered, the former two questions took all the time and energy making this question fall out of the scope of this project.

Thus for a future research a good topic would be to combine segmentation with object detection to further improve the whole hyperspectral data analysis. An example for why this would be useful would be that with segmentation it would be possible to detect object boundaries and this would eventually be very useful for picking up the plastic flakes with a robot.

Another topic would be to redo the experiments with a larger and more balanced amount of data, this could ensure that the trained models perform better. Regarding the Faster-RCNN model, this could be further looked at in the future as well to improve the results.

Acknowledgements

This project is a collaboration between NHL Stenden Professorship Computer Vision & Data Science and NHL Stenden Professorship Circular Plastics.

References

[1] Science History Institute. The history and future of plastics, 2020-10-05.

[2] Heinrich Böll foundation. 12 brief lessons on plastic and the planet. In *Plastic Atlas*, 2019.

[3] Sarah Moore. Major corporations back factory using enzymes to recycle plastics, 2019-10-28.

[4] Greenpeace.org. Learn about plastic pollution, 2020-10-05.

[5] Brownlee Jason. A gentle introduction to object recognition with deep learning, 2019.

[6] Wood Thomas. Convolutional neural network, 2019.

[7] Schneider Armin and Feussner Hubertus. Hyperspectral imaging. In *Biomedical Engineering in Gastrointestinal Surgery*, 2017.

[8] Lei Tong Jie Liang, Jun Zhou and Bin Wang. Material based salient object detection from hyperspectral images. In *Pattern Recognition*, 2018.

[9] Vigneshkumaran Meeradevi, Sharavana Raju K. Automatic plastic waste segregation and sorting using deep learning model. In *INTERNATIONAL JOURNAL OF SCIENTIFIC & TECHNOLOGY RESEARCH*, volume 9, 2020.

[10] Priya Dwivedi. Semantic segmentation-popular architectures., 2019.

[11] J. Shapey Y. Xie E. Nabavi R. Bradford S. R. Saeed S. Ourselin and T. Vercauteren. Intraoperative multispectral and hyperspectral label-free imaging: A systematic review of in vivo clinical studies.

[12] ujjwalkarn. An intuitive explanation of convolutional neural networks, 2016.

[13] Umme Zahoora Asifullah Khan, Anabia Sohail and Aqsa Saeed Qureshi. A survey of the recent architectures of deep convolutional neural networks. Artificial Intelligence Review, 2020.

[14] Mingxing Tan and Quoc V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks. Cornell University, 2019.
[15] Specim fx17.
[16] J. Redmon S. Divvala R. Girshick and A. Farhadi. You only look once: Unified, real-time object detection.
[17] Ayoosh Kathuria. What's new in yolo v3?
[18] Mingxing Tan Ruoming Pang Quoc V. Le. Efficientdet: Scalable and efficient object detection.
[19] Rohith Gandhi. R-cnn, fast r-cnn, faster r-cnn, yolo — object detection algorithms.
[20] Darrenl Tzutalin. Github : Labelimg software.

A Faster RCNN Experiment and Results

Since Faster-RCNN is the only model out of the three that did not function as expected there will be a broad analysis to illustrate why this is. This will be done by showing evidence from the experiments that have been done during this research. The hyperparameters and other noteworthy functions that were used during each experiment can be seen in Tables 20 and 21 below.

In Table 24 the parameters used for each experiment can be seen. Here is shown for each experiment which dataset is used and what the parameters for batch size, number of tiles, learning rate and patience were used for each experiment. Also information as whether or not augmentation was used, how many classes the model was trained with and how many epochs the model was trained for.

Table 25 shows how many images there are in each section of each dataset, for example the small RGB and HSI datasets both have 3 images in the training, validation and the testing sections. These datasets are a combination of all images taken throughout this project to make the dataset as large as possible since the assumption is that Faster-RCNN needs a lot of images.

Table 24. Experiments Faster RCNN

Experiments	1	2	3	4
Dataset	RGBbig	RGBbig	RGBbig	RGBbig
Batch size	2	2	2	2
Num tiles	3	3	3	3
Learning rate	0.0001	0.0001	0.001	0.001
Patience	10	500	10	10
Augmentation	False	False	False	True
Num classes	1	1	1	1
Epochs	150	100	100	100

Experiments	5	6	7	8
Dataset	RGBsmall	RGBbig	RGBbig	RGBbig
Batch size	2	2	2	2
Num tiles	3	3	3	3
Learning rate	0.001	0.001	0.0001	0.001
Patience	10	5	5	500
Augmentation	True	True	True	True
Num classes	1	1	1	1
Epochs	100	120	120	120

Experiments	9	10	11	12
Dataset	RGBsmall	RGBbig	HSIsmall	HSIbig
Batch size	2	2	2	2
Num tiles	3	3	3	3
Learning rate	0.001	0.001	0.001	0.001
Patience	10	10	10	10
Augmentation	True	True	True	True
Num classes	1	6	1	1
Epochs	120	120	100	100

Experiments	13
Dataset	HSIbig
Batch size	2
Num tiles	3
Learning rate	0.001
Patience	10
Augmentation	True
Num classes	6
Epochs	120

A.1 Experiments

With these experiments we hope to get an answer for the second research question “what is the best object localization approach when it comes to detecting plastic flakes?” with Faster-RCNN. The first three experiments were mainly done changing some hyperparameters to find the combination that would give the best results. After that, augmentation is added for the next experiments 4 and 5 to check if this has a significant impact on these results and if they would improve. After this, three more experiments were done with the changing of hyperparameters to see if this could improve the results again now that augmentation has been added.

Table 25. Datasets used

Dataset	**Training images**	**Validation images**	**Testing images**
RGBsmall	3	3	3
RGBbig	30	8	7
HSIsmall	3	3	3
HSIbig	54	16	13

A.2 Results

A.2.1 Experiment 1

For experiment 1 the model finds the right regions where the targets are but as can be seen from the results in Figure 19 there are a lot of boxes drawn and thus makes it difficult to see what is beneath these boxes. The loss curves for this experiment can be seen in Figure 20.

Fig. 19. Experiment 1 results

Fig. 20. Training and Validation loss curves of experiment 1

A.2.2 Experiment 2

For experiment 2 the learning rate and patience is increased to see if this makes an improvement on the results. As can be seen in Figure 21 there are less boxes drawn around the targets which make is easier to analyse. The loss curves for this experiment can be seen in Figure 22.

Fig. 21. Experiment 2 results

A.2.3 Experiment 3

For experiment 3 the learning rate is kept the same and the patience is lowered again to its previous value to check if this might improve the results. As can be seen in Figure 23 there are once again less boxes cluttered around the targets. The loss curves for this experiment can be seen in Figure 24.

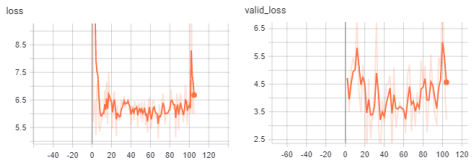

Fig. 22. Training and Validation loss curves of experiment 2

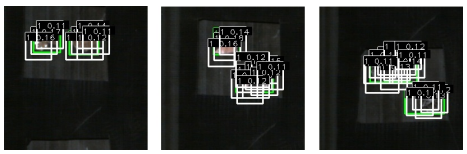

Fig. 23. Experiment 3 results

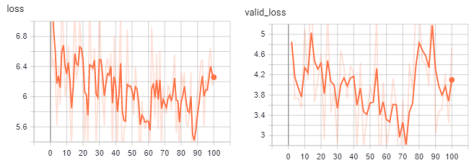

Fig. 24. Training and Validation loss curves of experiment 3

A.2.4 Experiment 4

For experiment 4 image augmentation has been added. Here the images are flipped over the x and y axis and are also rotated from the center to a varying amount of degrees. The results where similar to experiment 2 only it started to draw boxes in other places as well. As can be seen in Figure 25 boxes are drawn in a place where the image is no longer in view due to the rotation. Whether this is a bug or something else is still unclear since the rotations seem to work with the other models pretty well. It might have something to do with Faster-RCNN itself but this is something that has to be looked at in the future. A possible way to fix this would be to crop the image further to remove these edges altogether. But for now it was decided to do augmentation without the rotations from now on and check if this gives better results than without the augmentation at all. The curves from this experiment are shown in Figure 26.

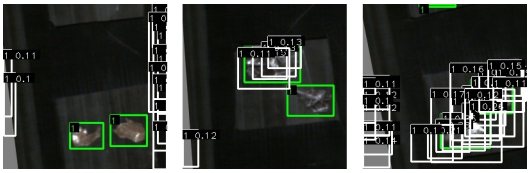

Fig. 25. Experiment 4 results

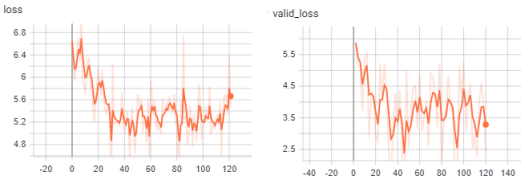

Fig. 26. Training and Validation loss curves of experiment 4

A.2.5 Experiment 5

In Figures 27 and 28 from experiment 5, the results with augmentation but without the rotation show that these gave a decent outcome in a way that there are far less boxes drawn around the targets. Therefore, this can be seen as the most optimal solution so far.

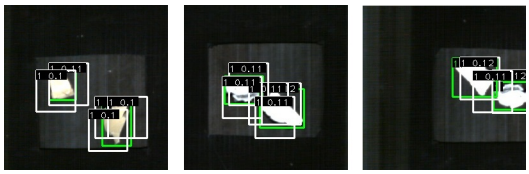

Fig. 27. Experiment 5 results

Fig. 28. Training and Validation loss curves of experiment 5

A.2.6 Experiment 6

To further improve on these results more experiments were done by making little changes to the hyperparameters. For experiment 6 the patience was halved to 5 to see if further reducing this parameter would improve the results. As can be seen in Figures 29 and 30 these results did not improve by much and seem to have an opposite effect as it appears that there are more boxes drawn than before.

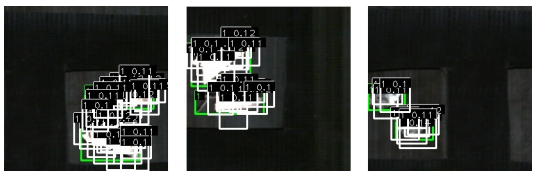

Fig. 29. Experiment 6 results

Fig. 30. Training and Validation loss curves of experiment 6

A.2.7 Experiment 7

For experiment 7 the learning rate was lowered since increasing it gave a “NaN or Inf found in input tensor” warning where the loss for both the training and validation would be NaN as can be seen in figure 31. So instead the learning rate was reduced to a tenth of the previous value. The results are shown in Figures 32 and 33 which don’t seem to have improved from the previous experiment.

A.2.8 Experiment 8

As for experiment 8 the patience was increased to 500 to see how the model would respond to this. The results are shown in Figure 34 and 35 which seem to only get worse.

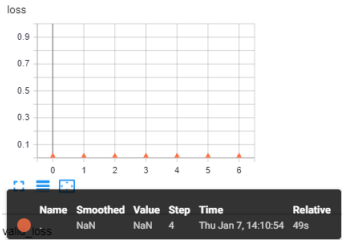

Fig. 31. Empty loss graph with NaN

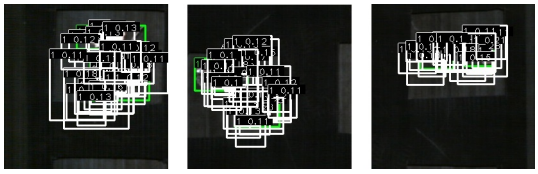

Fig. 32. Experiment 7 results

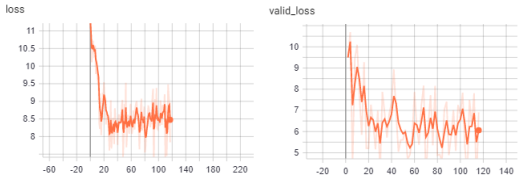

Fig. 33. Training and Validation loss curves of experiment 7

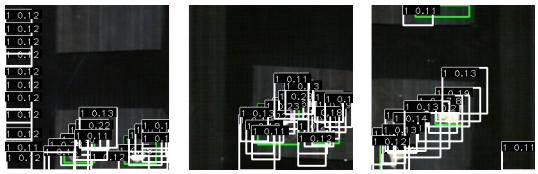

Fig. 34. Experiment 8 results

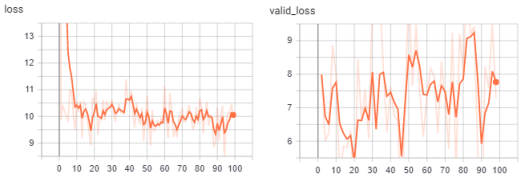

Fig. 35. Training and Validation loss curves of experiment 8

A.3 Conclusion

As can be seen from the results section, the results from experiment 5 seem to be the most optimal. Further experiments that where done by changing hyperparameters did not make any improvements on the results so far and thus can be concluded that the Faster-RCNN model performs poorly on the flakes dataset in it's current state. The results would possibly improve if more data is added to the dataset. Extra experiments have been done to investigate this further and the results of these experiments are shown in the next section.

A.4 Extra Experiments

With the current result no longer improving further this is where optimization for Faster-RCNN ends for this research. However other experiments have been done with these optimal settings that gave some interesting results as well. The next experiment, experiment number 9, is done to check if a bigger dataset would perhaps improve the results. This is done by training a model on a smaller dataset than the dataset used in the previous experiments to see the difference. In experiment 10 a training is done on all 6 classes were previous experiments were done only on 1 class. This is to see if Faster-RCNN is able to make sense of the data when there are more different classes. In experiment 11 and 12 the same experiment is run as with experiment 9, only now this is done with hyperspectral images to see if this would have the same result as with RGB. As a last experiment, experiment number 13, the same parameters were set as the previous experiments but now with all classes enabled to see if Faster-RCNN can classify and detect the flakes in an acceptable way.

A.4.1 Experiment 9

In experiment 9 we tested to see if a larger dataset would perhaps improve the results further. Here we trained a model with only three images compared to the 30 images that where in the dataset that the previous experiments were trained with. This experiment was done on the same values as experiment 5 which gave the best result so far. As can be seen from figure 36 the model does produce results with the small dataset but once again there are too many boxes drawn around the targets. By comparing the resulting images of Figure 36 with the images of Figure 27 it can be concluded that on RGB a bigger dataset does indeed improve the results of Faster-RCNN. The loss curves from experiment 9 can be seen in Figure 37.

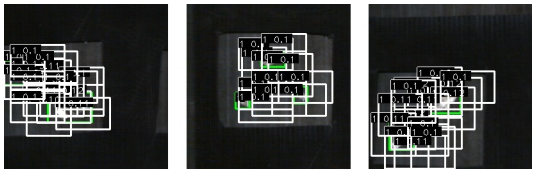

Fig. 36. Experiment 9 results

Fig. 37. Training and Validation loss curves of experiment 9

A.4.2 Experiment 10

Since experiment 5 is the optimal result so far we decided that for experiment 10 it is time to see what the result would be if the model has to classify more classes with these parameters. So instead of doing only object detection on one class this experiment was done on all six classes that are in the dataset. As can be seen from the Figures 38 and 39 below, the model does detect the objects, only classifying them is still a problem. This result is as expected since this is the very reason why the aim of this project is to try object detection and classification with hyperspectral images.

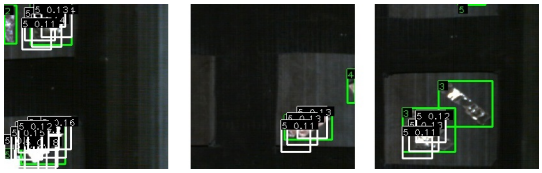

Fig. 38. Experiment 10 results

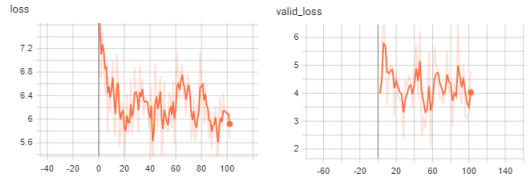

Fig. 39. Training and Validation loss curves of experiment 10

A.4.3 Experiment 11 and 12

For experiment 11 and 12 a training was done on a small and big dataset of HSI images with the most optimal parameters so far. This experiment is done on one class to check if the results matched the results from the previous experiments that were done on RGB. From the two leftmost images in Figure 40 that both originate from experiment 11 it can be seen that nothing is detected. This is the result of the small dataset with HSI images. The result from the rightmost image coming from experiment 12 seems a lot better which is the result of the bigger dataset of HSI images. From this it can be concluded that for HSI a bigger dataset will give better results as well. The loss curves from both experiments can be seen in Figures 41 and 42.

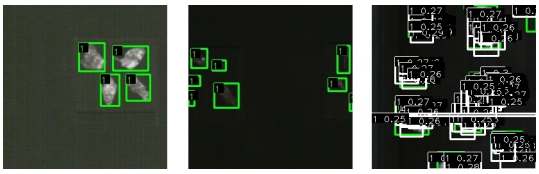

Fig. 40. Experiment 11 and 12 results

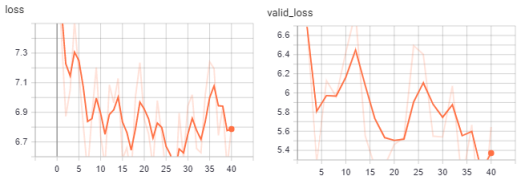

Fig. 41. Training and Validation loss curves of experiment 11

Fig. 42. Training and Validation loss curves of experiment 12

A.4.4 Experiment 13

At last for experiment 13 a training was done on the same large dataset as the previous experiment but now the model is trained to classify all six classes. The result can be seen below in Figure 43.
As can be seen in these images the results are decent enough that it is detecting the flakes and classifying them as class 5 which is correct. However, this is mostly for this class only and the other flakes are ignored most of the time. In the odd case that other flakes are detected they are classified as class 5 as well, as can be seen in Figure 44. The loss curves from this experiment is shown in Figure 45.

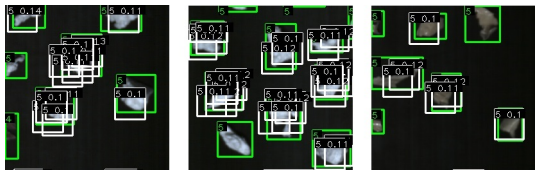

Fig. 43. Experiment 13 results

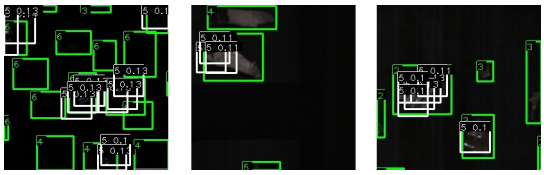

Fig. 44. Experiment 13 extra results

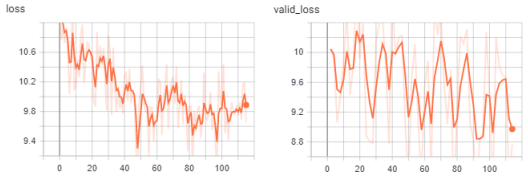

Fig. 45. Training and Validation loss curves of experiment 13

To get a full scope of the final results of the last test a complete reconstructed image is given below in Figure 46. In Table 26 and 27, the metrics are shown in two graphs. In the final result all the tiles are stitched back together to form the complete images. Here can be seen that all the boxes are in the right regions and none of the boxes are too far away from the targets. The results could be improved further by adding more images to the dataset since it became clear that this still seems to be the main problem.

Here in the table in Table 26 it can be seen that all the drawn boxes are of the fifth class as stated before. This is the polymer PS. Overall most of the boxes are spread across the other polymers as well but 28 boxes seem to be correctly drawn.

In the table in Table 27 the metrics can be seen which are all 0 as a result. This may be due to the fact that the scores predicted by Faster-RCNN are usually around 10 percent, as can be seen from all the images above.

Table 26. Confusion matrix for the flakes detected with Faster-RCNN network in experiment 13, the left column is the ground truth

	PE	**PET**	**PLA**	**PP**	**PS**	**PVC**
PE	0	0	0	0	6	0
PET	0	0	0	0	26	0
PLA	0	0	0	0	4	0
PP	0	0	0	0	16	0
PS	0	0	0	0	28	0
PVC	0	0	0	0	6	0

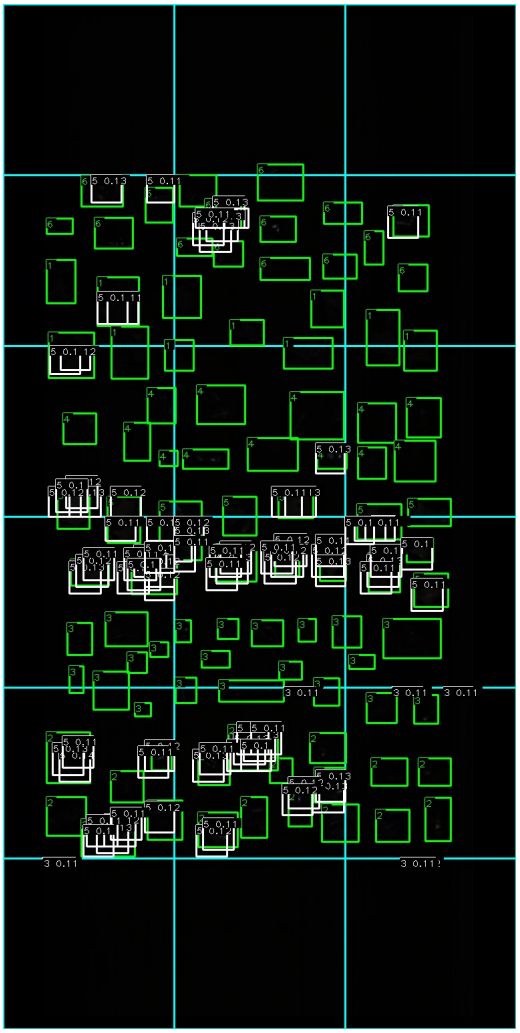

Fig. 46. Full reconstructed image of experiment 13

Table 27. Overall metrics from experiment 13

Precision	0
Recall	0
F1score	0
mAP	0
FDR	0