EXPLORATION OF DEEP LEARNING-BASED COMPUTER VISION FOR THE DETECTION OF FLOATING PLASTIC DEBRIS IN WATERWAYS

AJ VALLENDAR (4971752)

SUPERVISORS:

RICCARDO TAORMINA	(DELFT UNIVERSITY OF TECHNOLOGY)
STEF LHERMITTE	(DELFT UNIVERSITY OF TECHNOLOGY)
ZORAN KAPELAN	(DELFT UNIVERSITY OF TECHNOLOGY)
RINZE DE VRIES	(NORIA)

TU Delft

NORIA SUSTAINABLE INNOVATORS

Exploration of Deep Learning-based Computer Vision for the detection of floating plastic debris in waterways

By

AJ Vallendar (4971752)

in partial fulfilment of the requirements for the degree of

Master of Science

in Civil Engineering

at the Delft University of Technology,

to be defended publicly on Friday July 9, 2021 at 09:00 AM.

Supervisor:	Dr. ir. R. Taormina,	TU Delft
Thesis committee:	Dr. ir. S. Hermitte,	TU Delft
	Prof. dr. ir. Z. Kapelan,	TU Delft
	Ir. R. de Vries,	Noria Sustainable Innovators

This thesis is confidential and cannot be made public until July 1, 2021.

TU Delft

Acknowledgments

Before you lies my Master thesis ‘Exploration of Deep Learning-based Computer Vision for the detection of floating plastic debris in waterways’ concluding the Master study programme Civil Engineering, with a specialization in Urban Water Engineering. This thesis has been written to fulfil the graduation requirements from Delf University of Technology.

Throughout the writing of this thesis, I have received a great deal of support, guidance, and encouragement. I would first like to thank my two main supervisors: Riccardo Taormina and Rinze De Vries. It is difficult to put in words, but your incredible passion, enthusiasm and pro-active nature is absolutely contagious. The synergy you two created really inspired me and pushed my motivation to another level. I want to thank both of you for giving me the opportunity to conduct research on this amazing topic.

Riccardo, thank you for guiding me in my Deep Learning journey and giving me the freedom to explore, rather than restricting me to something specific. I have never had such an approachable and enthusiastic supervisor before. It is amazing how you can still provide such valuable and thorough supervision although you are completely overloaded with work (as I know you this will never change).

Rinze, I thank you for providing me with some of your Zen Buddha kind of calmness during my research. I am still awaiting the day that you openly express that something agitates you (but I guess this day will never come). You epitomize a real entrepreneur with your ambitions and mindset. Especially your practical nature was so valuable to carry out the experiments. This would not have been possible with you. One sentence I heard a lot from Rinze during this time and which I will always remember is:

“Alles is mogelijk, niks is onmogelijk”

This is the mantra Rinze lives by, and it rubs off on everyone surrounding him.

Stef and Zoran, thank you for your valuable feedback and great discussions. You really helped me to improve my thesis and see aspects that I had not seen before.

Hereby I also want to thank the Noria family Arnoud, Sophie, Fokke and Parshva. Thank you for the great talks and always creating such a homely atmosphere.

Tianlong, thank you for being my plastic tossing and gathering comrade during the experiments. Thank you for the great discussions and lunch meals.

Rich, thanks for being the greatest roomie someone could ask for. Your lively nature and positivity are an inspiration for everyone. Thank you for providing me also with some positive distractions to get my mind off the research.

I would also like to thank my dear friends. The people reading this will know who is meant.

Lastly, I would like to thank my family Albrecht, Manuka, and Saskia. You mean the world to me, and I am privileged to call you, my family. Thank you for your wisdom, support, and love. Nothing would be possible with you.

Summary

Plastic pollution is one of the most challenging global environmental problems. Currently, more than 1000 rivers transport approximately 80% of the plastic influx into the oceans. Naturally, more and more companies are interested in tackling this problem. One of them is Noria Sustainable Innovators, a company based in Delft (Netherlands). It is focussed on the detection, removal, and reuse of plastic from Dutch waterways. The company has the ambition to automate the detection of plastic for a wide range of applications. The quantification of plastic and understanding its spatiotemporal variability are crucial for the mitigation of plastic pollution. Current monitoring methods (e.g., visual counting) are tedious, time-consuming, and labour-intensive. Furthermore, the detection of different plastic debris objects could provide more insight about the source of plastic pollution.

This thesis explores the feasibility of automating plastic detection in waterways using modern deep learning (DL) algorithms named convolutional neural networks (CNNs) with image classification and object detection techniques. To train these models, a large dataset is required. Due to the unavailability of data, images were gathered in a controlled environment with two GoPros and a Huawei P30. The data was aggregated during sunny and cloudy conditions, different camera heights (2.7m and 4.0m) and angles (0 and 45 degrees).

For the simplest case (2.7m/0 degrees), a maximum accuracy of 87.6% was obtained for the multiclass classification of plastic debris in images, using the DenseNet121 model. By applying a majority vote for the three best performing models (DenseNet121, ResNet50 and InceptionV3), the accuracy could be increased to 91%. A qualitative and quantitative analysis found that the following factors influence the model performance negatively: presence of organic material, wind, transparent objects, submerged objects, small objects, overlapping and occluding plastic debris, sun glint and reflection of other objects on the water surface. Sunny conditions yielded a lower accuracy (79%) than cloudy conditions (90%), explained by the presence of sun glint.

By applying object detection, the error sources influencing the model performance could be reduced. For training and testing data from 2.7m/0 degrees on one class (‘plastic debris’), the YOLOv4 model yielded an accuracy of 95.61% (GoPro). For four classes (e.g., plastic bottles, other plastic, paper, metal tins) an average accuracy of 66.04% was found, indicating that the model experienced difficulties distinguishing different floating debris in water. Furthermore, it was also shown, that the use of a different image source (Huawei P30), does not have a negative effect on the accuracy (96.63%) compared to the original image source (GoPro).

Furthermore, due to height differences, discrepancy in object sizes and different camera settings, the trained model had large difficulties generalizing to a dataset from Indonesia (12.23%). On the other hand, training on the dataset from Indonesia and testing on the dataset from 2.7m/0 degrees achieved an accuracy of 63.51%. Although the error sources could be reduced, the model was still negatively impacted by small, transparent objects, submerged objects and the presence of sun glint.

This study clearly showed that Deep Learning-based computer vision can detect floating plastic debris with a high accuracy and have the potential to automate the process of plastic detection in the future. Future work would comprise the following aspects: sensor improvements (polarising filter for sun glint and multispectral sensor for continuous monitoring), data collection from the natural environment and different image sources, implementation of guidelines for Citizen Science platforms, addition of an object tracking module for monitoring (YOLOv4) and focussing on the detection of specific plastic debris objects after the removal from waterways.

Table of Contents

1 Introduction

The emergence of plastic pollution has become one of the most significant and challenging environmental problems of our times. From the 1950s onwards, where plastic had been invented for the use of sanitary and inexpensive material with near to limitless possibilities, it has gradually substituted paper and glass in the packaging of food and other goods, wood in furniture and metal in the productions of cars [1]. In this timeframe until 2018, the worldwide production of plastic has been largely expanded with a manufacturing increase from 1.7 to 359 million tons per year [2].

Furthermore, from the nine billion tons of plastic ever produced by mankind, only a mere fraction of 9% has been recycled [3], resulting in the fact that annually 4.8 to 12.7 million tons of plastic wind up in the world's oceans [4]. Plastics are resistant to biodegradation, however due to gradual wear off, macroplastics fragmentize into finer pieces named microplastics. Depending on the size of plastic, this can lead to either entanglement or ingestion by marine organisms, potentially leading to serious injuries and death [5]. Overall, it can be said that the presence of plastic litter in oceans and waterways is a global problem, associated with substantial impacts on the environment, public health, and economy.

Recently revealed by a study [6], it was found that plastic emissions are distributed over more rivers than previously assumed, with small urban rivers being amongst the main polluters. Through recent field observations and model calibration the study by [6] showed that more than 1000 rivers accouSnt for approximately 80% (Figure 1) of the global plastic flux with quantities ranging from 0.8 – 2.7 million metric tons per year.

Another current study by [7] concludes that a significant reduction of plastic debris in oceans can only be accomplished with collection systems at rivers or with a combination of river barriers and ocean cleanup devices. Therefore, it is crucial to focus on the removal of plastic in rivers now and in the future.

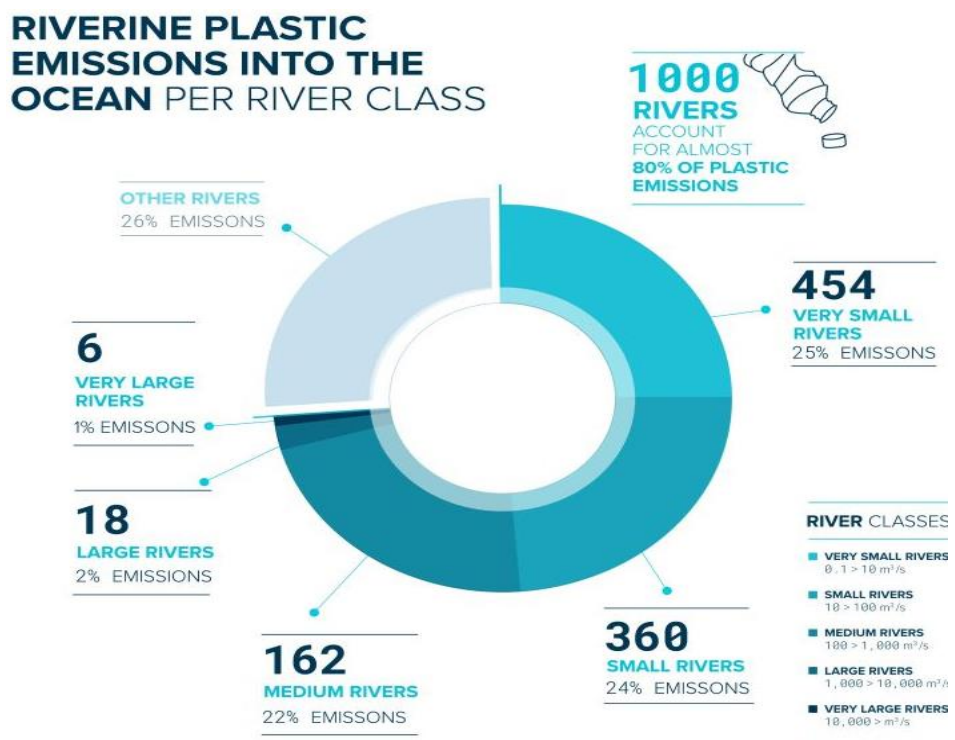


Figure 1 Distribution of riverine plastic emissions into the ocean per river class (image from [6])

Currently, most available research focusses on the quantities and detection of marine plastic instead of riverine plastics, resulting in the fact that riverine plastics are presently understudied [5]. Another key problem is that a knowledge gap exists regarding the distribution of plastic, due to its spatiotemporal variability [1] [8] [9]. It is therefore also crucial to focus on the detection of plastic debris to implement effective prevention strategies for the removal of plastic debris from waterways.

Numerous in situ and modelling approaches to conduct river plastic monitoring have been proposed in the recent past [10], including human visual counting [9] [11] [12], debris sampling by the use of drifting nets [13] [14] [15] and floating surface booms [16] [17]. Although the mentioned in situ methods provide site-specific data, these methods require additional equipment which includes cranes or boats and numerous people to perform measurements [5] [9]. Additionally, these techniques have a labour-extensive nature and are therefore inadequate for the application at different locations [5].

Alternatively, modelling approaches provide an option using secondary data, which is previously collected data that is accessible for researchers. Secondary data can include data on mismanaged plastic litter, population density, geography and hydrological properties to approximate the input of riverine plastic into oceans [2] [8] [10]. Nevertheless, although these models offer a first-order estimation of the local and global river plastic fluxes, they are highly dependent on approximations from a small amount of in situ measurements [18]. Therefore, model-based approximations are still prone to a large range of uncertainty, since the scarcity of field measurements is preventing the further progression of comprehending the dynamics of riverine plastic fluxes [4] [8] [11].

Video camera technology presents an alternative method, as these systems are frequently deployed as monitoring systems [19]. Especially products such as security cameras, drones and smartphones are easily accessible gadgets which can be used for monitoring purposes [10]. Although these image sources can provide sufficient temporal and spatial resolution for detection of riverine plastic, most of the existing methods currently developed are based on the visual detection and manual labelling of plastic fragments, which is a highly labour-intensive and time-consuming task [1]. It is also stated in a study by [10] that while the manual localization of plastic in images is viable that the automation of detecting plastic is a complex task.

In recent times modern artificial intelligence (AI) algorithms named convolutional neural networks (CNNs) have been widely used in object recognition and classification problems [1] [10]. CNNs fall into the category of deep learning and are a representation-learning method that supply the computer with raw data in form of images. This allows the automatic discovery of certain representations required for object recognition and classification [1] [10] [20]. Specifically, the emergence of CNNs have made computer vision (CV) techniques such as image classification, object detection and segmentation highly effective.

Due to the shortcomings of in situ methods and modelling approaches with respect to plastic monitoring this study explores an alternative method that is needed for the detection of plastic debris.

As this study was supported by Noria Sustainable Innovators, a company based in Delft which is focussed on the detection, removal, and reuse of plastic from Dutch waterways, it is evident that there is a practical need of an automatized system to detect plastic in water. This leads to the conclusion that parties with similar ambitions such as Rijkswaterstaat (RWS) would also profit from this research, which is one of my main intentions. Given these circumstances, the objective of this study becomes clear, namely, to develop and automated method for the detection of floating plastic debris.

Several previous studies that my work relates to suggested that the detection of floating plastic debris in a marine or riverine setting with deep learning-based computer vision could be applied effectively in the form of image classification [20] [21], object detection [10] [22] and segmentation [1]. This leads to the assumption that techniques of this nature have the potential to automate the process of plastic detection in water. Considering the

relevance of the issue and the acute nature of the underlying problem of plastic in waterways, the available studies do not seem sufficient to adequately cover this topic. Based on this, it can be assumed that work and validation is needed on multiple cases studies to prove and generalize findings. This leads to the following research question:

Does Computer Vision based on Deep Learning techniques have the potential to automate the detection of floating plastic debris in waterways?

However, for the application of deep learning algorithms for this study firstly a dataset in the form of images needed to be aggregated. The main problem is that that (open source) data on floating plastic debris is scarce. Gathering sufficient data for the application of deep learning algorithms is complex due to lacking monitoring networks. Additionally, a large majority of data from Citizen Science platforms such as CrowdWater or Litterati was unserviceable for the purpose of this study since a lot of images did not even contain plastic debris. To ensure a high accuracy of CNN based models a large dataset for training with several thousand images is required to ensure a high accuracy [22] [23]. Therefore, an additional method had to be found for the aggregation of a data that represents floating plastic debris in waterways. The gathered images served as the foundation for the application of deep learning-based computer vision techniques.

For the given duration of this research data collection from the natural environment would have been too time-consuming due to the spatiotemporal variability of plastic objects in waterways. To tackle these time constraints the process of capturing useful images and videos for the application of deep learning algorithms was accelerated in a controlled environment. Therefore, experiments were carried out at the Green Village which is a testing facility for sustainable innovation at Delft University of Technology. One of the central requirements was to ensure that the representation of the data was not limited by this approach. Once the dataset was gathered it was important to select a method of data analysis appropriate to the subject matter. To address the question …

1. **How do different deep learning architectures compare in terms of model performance for the classification of plastic debris objects?**

… I compared different deep learning architectures since the accuracy of detecting plastic can vary depending on the used deep learning architecture.

To assess whether the training set is adequately representative for the detection of floating plastic debris, it is inevitable to utilize datasets that were gathered in different locations and settings [10]. As the method to be developed should be applicable to any location and setting the requirement is to make the method as generalizable as possible. Therefore, one leading question of this work must be:

2. **How is the generalization influenced across datasets obtained using different sources of imagery (e.g., GoPro and mobile phone) and locations?**

The main idea of the experiments was to record floating plastic debris in a water body in form of a video, Therefore, plastic was tossed into the water, which was then recorded by different devices. For the experiments, plastic debris was gathered with volunteers from canals in Alkmaar (Netherlands). It is suggested by [10] and [11] that the detection of plastic debris objects is dependent on several parameters such as the weather conditions, sun orientation, characteristics of litter (e.g., colour, size, shape and floatability) and camera characteristics (height and angle). Since no other comprehensive deep learning-based study exists that explores the influence of weather-

and camera specific parameters on the detection of plastic debris in waterways, images were recorded for different weather conditions, camera heights and camera angles. Therefore, following aspect needs to be considered:

3. **In what way do various parameters (weather conditions, camera height and angle) influence the detection of plastic debris?**

Inspired by the ambition of Noria - to make a distinction between different plastic objects - this question become one of the leading questions of my study. This could potentially give insight about the origin of certain objects. With this knowledge, the origin of plastic debris could be tracked down more efficiently. Parties or industries responsible for the discharge of these objects could then be held accountable. This leads to the last of the four subquestions that I will particularly focus on in the following research report:

4. **To what extent can a deep learning model distinguish between different floating objects of interest (e.g., plastic bottles, bags, metal tins, paper) on the waterway?**

1.1 Limitations

This study merely aims to explore the possibilities of deploying deep learning-based computer vision techniques for the automated detection of plastic debris in waterways. Therefore, the products developed in this study form a foundation for automating plastic detection in waterways. The computational resources for this study were limited to the Graphical Processing Units (GPUs), offered by Google Colaboratory. The resources could be utilized for 12 hours continuously. Due to this limitation, the models could not be trained until full convergence.

Furthermore, this research will only consider floating macroplastics, which are defined as particles, items and objects larger than 25mm [24] [25]. Due to the limitation of RGB cameras, only floating plastic debris were considered. Suspended plastic debris were consequently not considered.

1.2 Embedding of the research project

AidroLab was chosen as the first Artificial Intelligence laboratory hosted by the Civil Engineering & Geosciences faculty at Delft University of Technology. Their mission is to develop novel artificial intelligence techniques that can exploit the abundance of data to improve the resilience of the urban water systems. This Master thesis project was conducted as an internship in collaboration with Noria Sustainable Innovators. Furthermore, the thesis was part of a PhD research from AidroLab that focusses on the 'Development of Artificial Intelligence techniques for monitoring and plastic removal in urban waterways'. The PhD is carried out in collaboration with Noria and Rijkswaterstaat (RWS). The findings of this study can be used to complement the PhD research.

1.3 Report structure

The second chapter will provide the reader with the needed technical background to comprehend the content of this thesis. The third chapter ('Methodology') focusses on the description of the data collection, the data processing, the description of used model architectures and the evaluation of model performances by means of accuracy metrics. After discussing these aspects, the results (chapter 4) and discussion (chapter 5) will be presented. This will lastly be followed up by the conclusions from this research (chapter 6) and recommendations for future research and applications (chapter 7).

2 Theoretical background

This chapter will supply the reader with the necessary theoretical background to understand this report. Firstly, the term 'plastic debris' will be introduced, which will focus on the type and size of objects that will be considered in this study (section 2.1). The following section (section 2.2) will focus on the video camera technology in combination artificial intelligence (AI), with a specific focus on convolutional neural networks (CNN).

2.1 Definition of plastic debris

Figure 2 Real-life plastic debris hotspots in Leeuwaarden (1), Den Haag (2) and Alkmaar (3). Image (4) characterizes the sort of plastic debris that can be found in hotspots (gathered in Leeuwaarden)

Since plastic hotspots are a real-life representation, these will be considered to determine the type of objects encountered in surface water systems in the Netherlands. Figure 2 visualizes real-life plastic debris hotspots that were found during fieldwork activities with Noria Sustainable Innovators. These and other encounters of plastic debris hotspots in the environment lead to the following definition of a plastic hotspot:

'***A plastic debris hotspot is a (stagnating) accumulation of debris with varying sizes and shapes, which does not only include the presence of plastic (e.g., plastic bottles, bags and other plastics), but other kinds of objects such as paper, metal tins, glass bottles and organic material. These hotspots are encountered in areas of low flow velocities, dead ends in canals, infrastructure (e.g., bridges) and riverbanks.***'

This definition is in line with the perception Noria has of plastic debris hotspots. The range of plastic debris objects also corresponds to the litter items encountered in a two-hour canoeing session in the canals of Leiden (Netherlands) in a study from [26]. The definition of a plastic hotspot was needed to determine the type of objects that represent the variety of floating plastic debris that are found in waterways. It is important to mention, that the methods developed in this study do not refer to the detection of plastic hotspots but to the floating objects that accumulate in these hotspots.

One area of concern is the classification of plastic sizes throughout studies since consistent terming and dimensions for data comparison are lacking [5] [27], due to differences in sampling protocols [27]. A segregation of plastic by size is advisable for the determination of the plastic pollution source and the environmental impact. For plastic sampling the most widely used categorization classifies plastic debris into microplastics (1 – 5 mm), mesoplastics (5 – 25 mm) and macroplastics (> 25 mm). This convention has been adopted by the UN Environment Programme (UNEP), MSFD Technical Subgroup on Marine Litter, National Oceanic and Atmospheric Administration (NOAA) and studies by [24] and [25]. The categorization is visualized in Figure 3 and this study will only consider macroplastics (>25 mm).

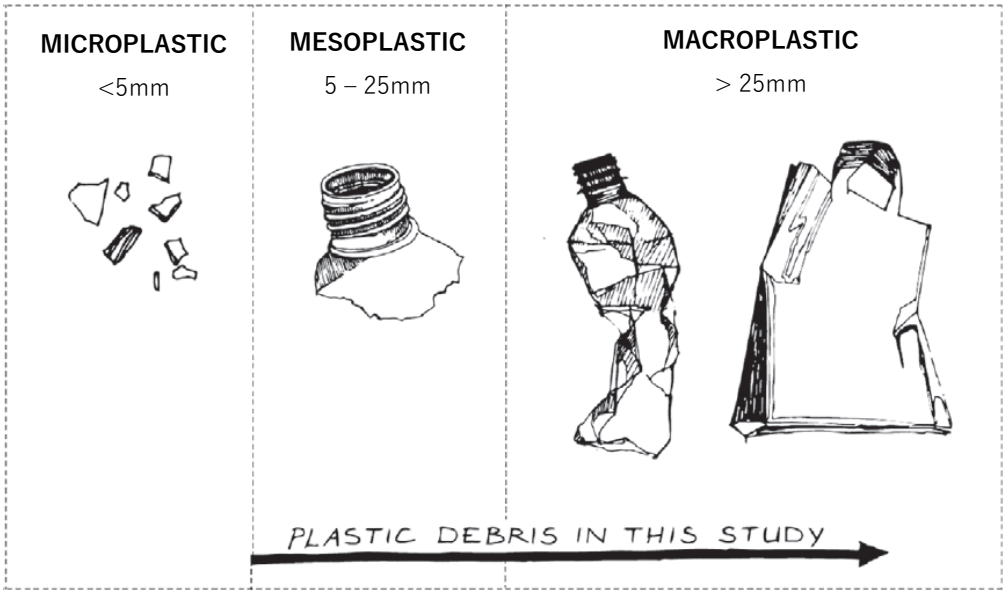


Figure 3 Most common categorization of plastic into categories of microplastic (<5 mm), mesoplastic (5-25 mm) and microplastic (>25 mm) (image taken and adapted from [5]

2.2 Combining Video Camera Technology and Artifical Intelligence (AI)

Video camera technology presents a method for an automated approach in the detection of floating plastic debris, as these systems are frequently deployed for monitoring purposes [19]. Especially security cameras, drones and smartphones are easily accessible gadgets which can be used for monitoring and offer three main benefits [10].

Firstly, the use of video cameras offers the opportunity to observe floating plastic in turbid rivers, which comprise the largest fraction of riverine plastic transport [10] [28]. According to a study by [10], it is possible to observe macroplastics for a height of 4-9 m, if a camera is installed perpendicular to the water surface.

Secondly, according to a research by [29] Unmanned Aerial Vehicles (UAVs) such as drones have established themselves as a low-cost and reliable image-capturing tool, allowing high accuracy monitoring of aquatic environments. Furthermore, through the deployment of UAVs for aerial surveys the monitoring of riverine plastic is less dependent on the local environment, as opposed to methodologies whereby existing infrastructure and accessible areas are required [30], such as static cameras.

Lastly, smartphones also offer a source for monitoring, especially in combination with Citizen Science data collection platforms. One example is a mobile application named 'CrowdWater' that includes a photographic documentation module for macroplastic in an urban or natural environment [26] [31].

Although video cameras, UAVs and mobile phones can identify and locate plastic debris in images most of the existing methods currently developed are based on the visual detection and manual labelling of plastic fragments, which is a highly labour-intensive and time-consuming task [1].

Within machine learning three primary paradigms of learning can be distinguished, namely: unsupervised, supervised and reinforcement learning. A task is called unsupervised learning, if only the input data is known and the algorithm is still capable of learning and discovering specific features on its own.

On the other hand, in supervised learning tasks the input and the output of the data are already known, but this technique also encompasses the mapping of an input to an output. To be able to predict the outcomes, data needs to be labelled beforehand, such that the algorithm can learn from these examples. The model is thus supervised or guided in the process of training itself on how to predict a certain output. For supervised learning a distinction can be made between a regression (continuous output) or a classification task (discrete output).

Reinforcement learning is a process whereby interaction takes place between an agent and its environment, resulting in a feedback or a signal that reflects the performance of the agent related to its task.

This study applies supervised learning for the detection of floating plastic debris, therefore unsupervised and reinforcement learning techniques will not be discussed further. For image classification the problem at hand becomes a classification task. If an object detection algorithm is considered, algorithms can either be distinguished as a classification or regression task. This is dependent on the chosen algorithm.

Modern artificial intelligence (AI) algorithms named convolutional neural networks (CNNs) have been widely used in classification problems and object recognitions [1] [10]. Connecting the automatization of plastic detection based on a deep learning approach offers added advantages, such as saving valuable search time, lower expedition costs and an increase in the accuracy of detecting larger fragments of plastic [20]. A study by [20] suggests, that the application of CNN algorithms can facilitate a faster and more accurate creation of tools for the detection and classification of afloat plastic debris. This suggests, that the application of CNNs for this study are a favourable method. But how do CNNs work and what advantages do they have for this study compared to other methods?

2.2.1 (Deep) Neural Networks

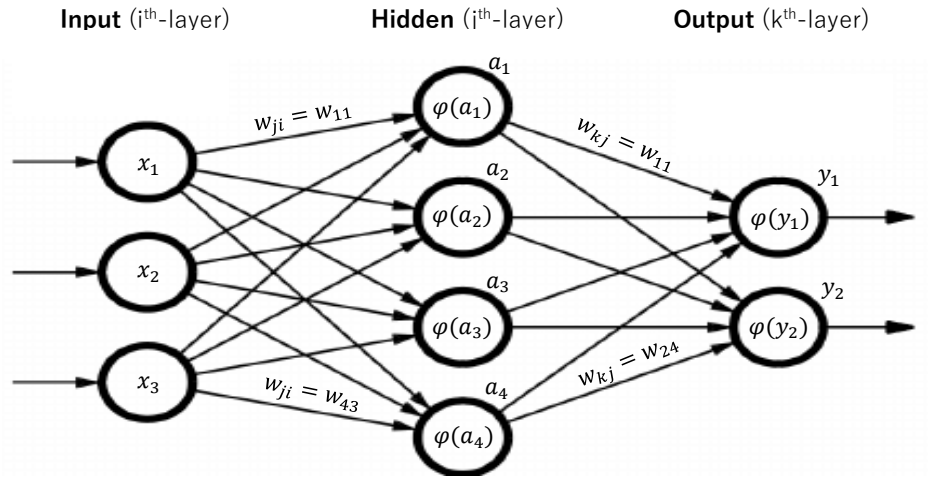


Figure 4 Single hidden neural network with input layer, one hidden layer and an output layer. The nodes represent the neurons and the arrows between the neurons visualize the connections between the individual neurons and the applied weights

Artificial Neural Networks (ANN) are a class of machine learning algorithms based on the functioning of the human brain. The first of such methods to have been proposed, is the single hidden layer neural network (Figure 4). In this model, three layers can be distinguished, namely: the input, hidden and output layer. The input layer performs no computational processing. However, it receives an input that needs to be mapped to the output variable by specified operations. If a 32x32 greyscale image is considered, the total number of neurons is equal to 1024, whereby each neuron contains the greyscale value of the corresponding pixel. A neuron is a node in the network, where impulses (or signals) from different sources assimilate. The number of neurons in the output layer is dependent on the desired output variables of the dataset. As an example: the CIFAR-10 dataset e.g., distinguishes between 10 different classes, therefore the size of the output layer would be equal to the number of classes.

Furthermore, in between the input and output layer, hidden layers with an arbitrary number of neurons can be found. Within hidden layers most of the processing takes place. The number of hidden layers and neurons in these layers are a design choice and are dependent on the problem at hand. If the number of hidden layers is equal or larger than two, the network becomes by definition a deep neural network (DNN).

A neural network is called fully-connected or dense, if all neurons in a layer (l) are connected to all neurons in the subsequent layer ($l+1$). The conveyed information from a neuron in layer l to a neuron in the following layer l+1 is multiplied by a weight w_{ji}, which is the weight of the connection between the i-th input neuron and the j-th hidden neuron. A bias term (b_j) of the j-th hidden neuron is added to the multiplied value of the input (x_i) and the weight w_{ji}. All the information gathered in layer l is then summarized in every individual neuron of layer $l+1$.

Until this point, all performed operations within the neural network were only a combination of linear functions. The addition of non-linearity is needed, such that the designed network can model more complex relationships between the input and output. Consequently, the value of each neuron passes through an activation function (φ), resulting in a non-linear input for the following layer. The mathematical function for a single neuron (a_j) and the output layer is represented in equation (1) and (2):

(1) $$a_j = \varphi\left(\sum_{i=1}^{m}(w_{ji}x_i) + b_j\right)$$

(2) $$y_k = \varphi\left(\sum_{j=1}^{N}(w_{kj}a_j) + b_k\right) = \varphi\left(\sum_{j=1}^{N}\left(w_{kj}\varphi\left(\sum_{i=1}^{m}(w_{ji}x_i) + b_j\right)\right) + b_k\right)$$

Equation (2) illustrates, that the non-linear output of the hidden layer is multiplied by a new set of weights (w_{kj}) in the subsequent layer, followed by the addition of a bias term (b_k) and the application of an activation function (φ). It becomes evident that the addition of more hidden layers implies, that the network will be capable of predicting more complex relationships, depending on the depth of the network.

Forward and Backward Propagation

To understand how an ANN or DNN can predict the result of the input based on equation (1) and (2), the processes of *forward and backward propagation* within a network needs to be explained. The computation of a NN with respect to the input is commonly known as *forward propagation*. To correctly predict the outcome, the network needs to search and find a specific combination of weights and biases, such that the input data can be mapped to the correct output. Therefore, the model firstly initializes a random set of weights to assess the accuracy on predicting the proper outcome. The error or loss, which is based on the difference between the predicted and true outcome, is calculated by a loss function. For simplicity it is assumed that β is a vector containing all weights and biases across all layers of the neural network, J is the loss function, y the observed values and $\hat{y}$ the predictions which were obtained during forward propagation. Since y are observations which do not change their value, following equation can be defined:

(3) $$J = f(y, \hat{y}) = f(\beta) = f(\{\beta_i, i = 1 \dots n\})$$

A process called 'gradient descent' updates every model parameter based on the derivative of the loss function, aiming to find the optimal combination of parameters by minimizing the loss function. This is represented by the following equation:

(4) $$\beta_i := \beta_i - \eta\frac{\partial J}{\partial \beta_i}$$

where η represents the learning rate, which governs the magnitude of the update. The choice of the learning rate can have a major impact on the efficiency and success of gradient descent. This can cause drastic updates and divergence if the learning rate is too high and slow convergence and many updates if the learning rate is too low. Therefore, most commonly an adaptive reduction procedure is applied to the learning rate during the training.

This process is also called backpropagation. This process computes the partial derivatives of a loss function with respect to the parameters of an ANN. Depending on the structure of the ANN, this requires the derivation via chain rule for the parameters in the hidden layers. By knowing the partial derivative the parameters can be adjusted according to the error. The calculated loss is back propagated through the network by updating each weight and bias, yielding whether the loss decreases for the updated parameters.

This procedure is continued until a point is reached, where the model has stopped learning. This is indicated either by stagnating or decreasing accuracies or stagnating and increasing losses, or if the model has reached the desired performance. One complete pass of data through the network by forward- and backward propagation is defined as one *epoch*. Especially with deep learning algorithms, the quantity of data is often too large for a computationally efficient pass through a network before the weights and biases are updated. Therefore, the dataset is split into smaller sets, also known as *mini-batches*. After every pass of a *mini-batch* through the network, the model weights and biases will be updated. To what extent these parameters are updated, is dependent on the used optimizer algorithm.

2.2.2 Convolutional Neural Networks (CNNs)

The principles of NNs also apply to CNNs, however CNNs are much more efficient if large-sized input data, such as images are used. If a fully-connected ANN is considered, whereby the input is a commonly used image size (224x224), the size of the input vector for a greyscale image would be an input vector with 50.176 features. Considering RGB images, whereby the three colour dimensions 'R' (Red), 'G' (Green) and 'B' (Blue) need to be considered, the input vector contains 224x224x3 = 150.528 features. If the NN with four neurons in the hidden layer in Figure 4 is taken as example, 602.112 weights and biases would need to be updated in the first hidden layer, if a 224x224x3 RGB image is considered as input, This example illustrates, that NNs are computationally inefficient for larger sized data. For what reason CNNs are much more efficient than ANNs for larger sized input, will be explained in the following paragraphs.

A CNN architecture typically includes alternating convolutional and pooling layers (Figure 5), which is followed by one or more fully-connected layers. In addition to several activation functions, different regularization components such as batch normalization and dropout can be integrated to optimize the performance of the CNN [32]. Common CNN architectures are commonly divided into two sections, namely: feature extraction and a classifier or regressor. The different CNN units are crucial in the assembly of a CNN architecture, since the arrangement of these components determine the performance of the model. This subparagraph will provide an explanation on the roles, functioning and mathematical background of the different CNN components.

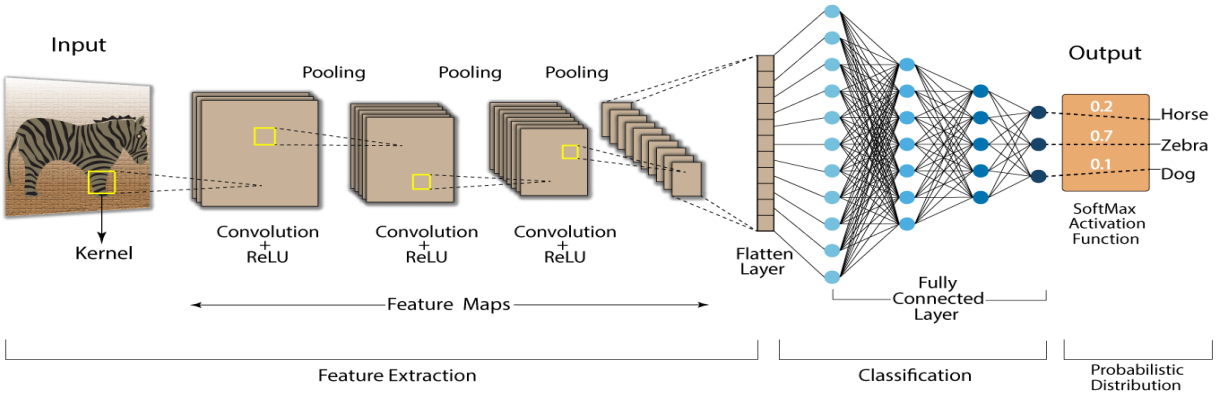


Figure 5 Classical CNN architecture with a segregation into feature extraction, classification and probabilistic distribution. These sections include an input image, kernels (filters), convolutional (section 2.2.2.2) + Rectified Linear Unit (ReLU) layer (section 2.2.2.3), pooling layers (section 2.2.2.4), a flatten layer, fully-connected layers (section 2.2.2.7) and an output vector with probabilities depending on the defined classes

2.2.2.1 Structure of digital input image

The input to a convolutional layer is represented by a $NxMxD$ image, whereby N is the height of the image, M is the width of the image and D the number of channels. Every pixel in an image represents a certain numerical value associated with the brightness of a specific pixel. The depth of an image is dependent on whether greyscale or RGB images are used as an input. RGB images possess three colour channels, consequently resulting in $NxMx3$ image corresponding to the respective colour channel, whereas for greyscale images the spatial dimension is $NxMx1$. RGB and greyscale images store numerical values from 0 to 255. This range represents a trade-off between the efficiency of storing data (256 values ≙ 1 byte) and the sensitivity of the human eye, which can only distinguish between a limited number of shades originating from the same colour [33].

2.2.2.2 Convolutional layers

Convolutional layer operations comprise the 'sliding' of a set of convolutional kernels (or 'filter') over the input image, which is a matrix of weights. Convolutional kernels operate by splitting the image into small sections, commonly also known as 'receptive fields' [32]. The kernel convolves with the input images, by using a set of weights and multiplying these with the corresponding elements of the receptive field [32]. One receptive field generates one node in the feature map. Mathematically, convolutional operations can be defined as follows:

$$f_l^k(p,q) = \sum\sum i_c(x,y) e_l^k(u,v) \tag{5}$$

where $i_c(x,y)$ expresses an element of the input image tensor I_c, which comprises an elementwise multiplication with $e_l^k(u,v)$ index of the k^{th} convolutional kernel k_l belonging to the l^{th} layer. The feature map of the k^{th} convolutional operation can consequently be expressed as follows:

$$F_l^k = \left[f_l^k(1,1), \dots, f_l^k(p,q), \dots, f_l^k(P,Q)\right] \tag{6}$$

For every time a convolutional operation is performed, the dimensionality of the image shrinks until the image vanishes. Furthermore, since the kernel convolves over the image, the impact of the pixels located on the boundary of the image is significantly smaller than those situated in the centre of the image. This process could lead to loss of information. To solve this issue, it can be chosen to pad the boundaries ('same' padding) of the image with an additional pixel border (Figure 6), such that either the input and output size of the are the same, or the output is larger than the input size. If no padding is applied, this refers to 'valid' padding, whereby the original image size is used for the convolutional operations, ultimately resulting in a shrunken output size.

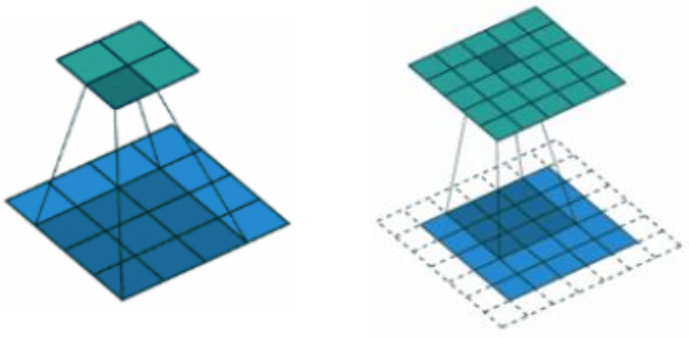


Figure 6 Visualization of 'Padding' in CNNs, with 'valid' padding (Left) and 'same' padding (Right) are represented with stride=1

Additionally, the size of the 'stride' or 'step length' of the filter needs to be considered in convolutional layers. The 'stride' can be increased if less overlap of the receptive field or a reduction of spatial dimensions is desired.

Lastly, a concept named convolution over volume is an important aspect that not only allows convolutional operations to be carried out on RGB images, but it also permits to apply a multitude of kernels on a single layer.

Commonly, a multitude of filters are used in convolutional layers to extract more information and features from the image, resulting in a number of feature maps that are equivalent to the number of filters used in a convolutional layer. It is a prerequisite that the applied filter and the input have the same number of dimensions. In three-dimensional space, the matrix multiplication between the filter and the image, is applied to all three colour channels. If it is desired to use multiple filters for the detection of features in an image, the convolutional operation for each filter is carried out separately, followed by summing the channels and adding a bias to the resulting feature map. Following equation can be applied to calculate the output dimension of a convolutional layer [33]:

$$[n, n, n_c] * [f, f, n_c] = \left[\left[\frac{n + 2p - f}{s} + 1\right], \left[\frac{n + 2p - f}{s} + 1\right], n_f\right] \tag{7}$$

whereby n represents the image size, f the filter size, n_c channel number present in the image, p the applied padding, s the used stride and n_f the number of filters.

One key advantage of CNNs compared to fully connected neural networks is parameter sharing [32]. Parameter sharing allows convolutional layers to exploit spatial inductive bias and grants translational equivariance. This means that an object of interest can be recognized regardless of its position in the image, whereas ANNs would have to learn to recognize the object in each position separately.

Next to parameter sharing, the concept of local connectivity, whereby each neuron is only linked to a subset of the input image, is a powerful attribute of CNNs. Convolutional layers exploit a 'spatial inductive bias' since correlation among pixels in images is usually local. Therefore, wasting resources by connecting the entire input would be redundant. Local connectivity and parameter sharing make CNNs more efficient compared to densely connected NNs, due to the reduction of parameters and consequently computation time.

Lastly, 'hierarchical feature extraction' works well for CNNs in image classification tasks. This entails that filters in the first layers of the network recognize edges and other basic features in the image. Deeper layers recognize more complex and abstract parts of the object or entire objects.

2.2.2.3 Activation function

Activation functions serve as a decision function and they help in learning complex patterns [32]. When transforming the feature maps of convolutional layers, the activation function can be written as follows:

$$T_l^k = g_a(F_l^k) \tag{8}$$

where F_l^k is the feature map of a convolution, which is fed to an activation function $g_a(.)$ that includes non-linearity and returns a transformed output T_l^k of the l^{th} layer [32]. The addition of non-linearity is needed because the neural network would otherwise only be a combination of linear functions, limiting the flexibility and creation of more complex functions during training [34]. The Rectified Non-Linear unit (ReLU) is commonly used as the activation function for DNNs since it prevents the vanishing gradient problem opposed to other activation

functions [35]. This problem refers to a large and rapid decrease of the gradients caused by backpropagation. Once the shallower layers of the CNN are reached, the update for the weights nearly vanishes [1].

2.2.2.4 Pooling layers

Different extracted features, which are the output of a convolutional operation, can occur at various locations of an input image. According to [32], the position of the features becomes less important, if the approximate position of the feature opposed to others is preserved. Pooling is a local operation, that outputs the dominant response within a region by simultaneously downsampling the respective feature map, thereby decreasing the number of parameters and consequently also the computational burden. Most commonly, two main types of pooling are applied, namely: maximum and average pooling. The procedure of these two operations is visualized in Figure 7.

The mathematical representation of the pooling operation is given by equation (9):

$$Z_l^k = g_p(F_l^k) \tag{9}$$

where Z_l^k symbolizes the pooled feature map of the l^{th} layer for the k^{th} input feature map F_l^k and $g_p(.)$ describes the type of pooling operation [32]. The most important advantage of applying pooling operations, is the addition of generalizability by making the model more robust against errors created by translational invariance [36] [37], by extracting features that are invariant to positional shifts and smaller distortions [32] [38]. The reduction of the feature map with respect to invariant features firstly controls the complexity of the model and secondly improves the generalization capability by the reduction of overfitting.

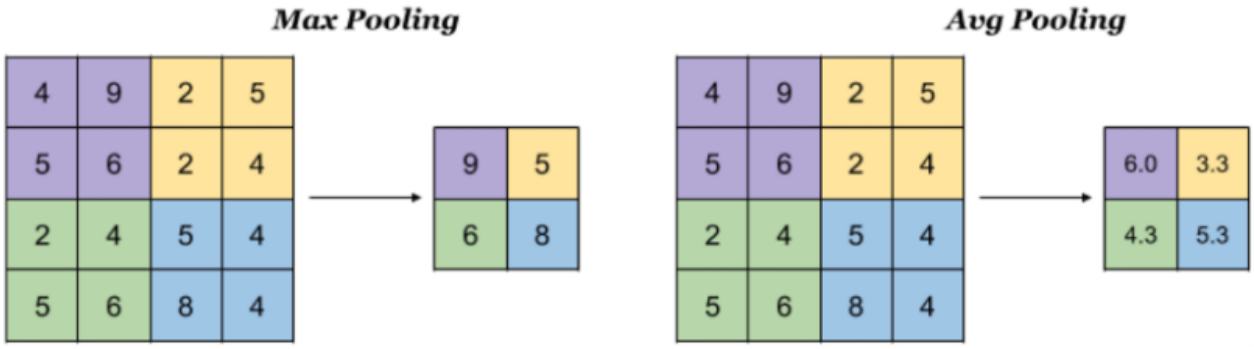


Figure 7 Example of maximum pooling (Left) and average pooling (Right) for a 4x4 matrix, whereby maximum pooling takes the maximum value of the respective color-coded section and average pooling the average value (image from [79])

2.2.2.5 Dropout

Overfitting usually occurs, if a model with large learning capability is trained on a small dataset and the model can memorize the used training data. This results in a minimal generalization power of the model by 'fitting to the noise' [1]. In other words: the model produces a good fit with the used training data, while this does not translate well to new and unseen data. With a scarcity of training data, the model learns patterns specifically associated to the training data, which are irrelevant for other data. One technique to diminish overfitting is *Dropout*. This is a technique that introduces regularization to the network, by randomly omitting nodes or connections in forward propagation with a predefined probability in the network, aiming to decrease overfitting and thus improving generalization of the network.

The intuition behind this procedure is that the input will be passed to a differently configured and smaller NN architecture due to the random omission of nodes during forward propagation, If one training example is considered, the neuron learns some specific feature of the example during gradient descent, which is also dependent on the input values. As the layers become more random in nature due to dropout, each neuron is forced to learn the entirety of the training example, rather than only learning one specific feature. The neuron cannot rely on any feature, because the inputs will randomly be eliminated and therefore the neuron is 'reluctant' to put too much weight on any input. Therefore, the neuron is required to spread out the weights more evenly.

2.2.2.6 Batch normalization (BN)

Batch normalization (BN) layers are used to address issues related to the internal covariate shift within feature maps [32]. Internal covariate shift is a phenomenon which refers to the change in distribution of the input to layers in the network. This slows down convergence by forcing the learning rate to take on a small value. Batch normalization standardizes the inputs to the layers and therefore stabilizes the learning process of the model. The mathematical formula for a feature map F_l^k is given by equation (10):

$$N_l^k = \frac{F_l^k - \mu_B}{\sqrt{\sigma_B^2 + \varepsilon}} \tag{10}$$

whereby N_l^krepresents the normalized feature map, F_l^k is the input feature map, μ_B is the mean of a feature map for a mini batch and σ_B the variance. To avoid a division by zero, a term for the numerical stability ε is added. Next to standardizing the input to every layer, BN also adds a slight regularization effect next to dropout by adding some noise to the values in the mini batch, also helping to improve the generalization capability of the network.

2.2.2.7 Fully connected layer (FCL)

For classification or regression purposes, the FCL is predominantly located at the end of a network. Unlike convolutional and pooling operations, the use of a fully connected layer is a global operation. In CNNs, FCL layers are used to build the final classifier or regressor that processes the feature extracted by the network to make a prediction. In the case of Figure 5, the output from the feature extraction stages is collapsed into a 1D feature vector, also called the *flatten layer*. The FCL can be a NN, which takes the flattened vector from the feature analysis as an input and applies weights to predict the proper labels. The mechanism of a FCL is identical to a NN in terms of computation, however in the last layer the activation functions can vary. In case of binary classification, a sigmoid function is used as the activation of the output neurons, a softmax function in case of a multiclass classification and a linear function for regression. Sigmoid and softmax functions output probabilities corresponding to the predefined classes, whereby the sum of the probabilities for all classes is equal to 1. The linear activation function outputs an unbounded numerical value.

2.2.3 Computer Vision (CV)

Deep learning algorithms have made computer vision (CV) tasks highly effective in today's world. Specifically, the arrival of CNNs have made CV practical for industrial applications. The field of computer vision aims to provide computers with the ability to identify, classify and categorize visual information encompassed in imagery or video data [39], such as humans with their eyes. Within computer vision, image analysis deals with the extraction of

important information from images, which should not be confused with image processing that focusses on the alteration of an image and the subsequent creation of a new image with improved characteristics [39]. Within image analysis, three categories can be distinguished, namely:

1. Image Classification and Localization (Figure 8 B)
2. Object Detection (Figure 8 C)
3. Image Segmentation (Figure 8 A and D)

In image classification the aim is to assign a class to an entire image. This could include for example classifying x-rays with cancer or not (binary classification) and the classification of a handwritten digit (multiclass classification). The image cannot be partitioned into segments and by processing the image all at once, regions in the image will be included that do not include any information. By segregating the image into segments, the important segments can be used for further processing. The idea of image classification can be extended by localization, which involves the assignment of a class to an image and the localization of an object with a bounding box within the image (Figure 8). Nevertheless, image classification and localization are restricted, because it can only detect a single object in an image.

Object detection builds upon the idea of image classification with localization, whereby multiple objects can be present in the image, which require localization and classification with bounding boxes. This is also depicted in Figure 8, whereby two classes receive different bounding boxes. Nevertheless, object detection does not convey any information about the shape as the bounding boxes have either rectangular or square shapes.

Instance segmentation combines the ideas of object detection and semantic segmentation. Semantic segmentation is comparable to image classification, however instead of classifying the entire image, every pixel in the image is assigned to predefined classes, without localizing the desired object. Instance segmentation touches upon the idea of classifying each pixel of a different class by also making a distinction between various objects of the same class. Therefore, a pixelwise mask is created for every object contained in an image.

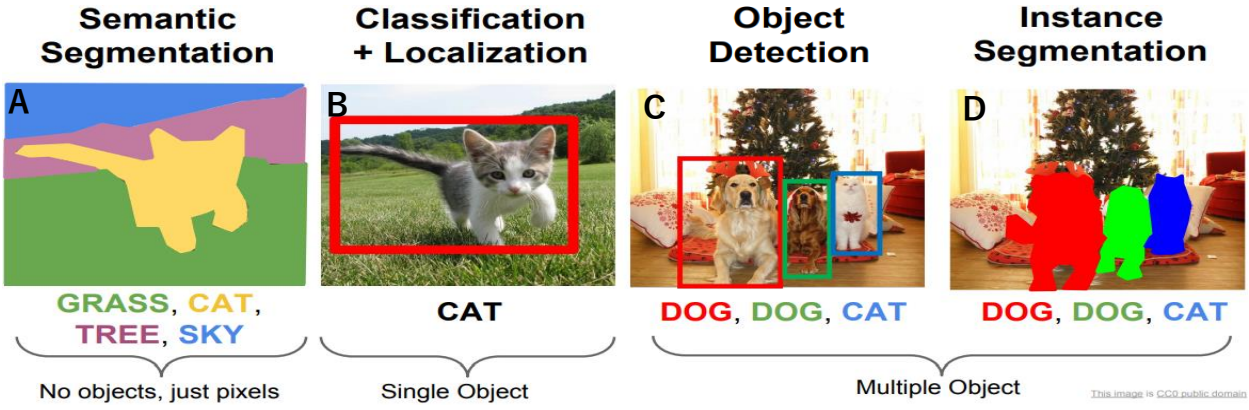


Figure 8 Computer vision techniques for image analysis with Semantic segmentation (A), Classification + Localization (B), Object detection (C) and Instance Segmentation (D) (image taken from [80])

3 Methodology

The main goal of this research was to investigate whether deep learning-based computer vision techniques can detect floating plastic debris in waterways. The applied methodology to achieve this objective will be described in the following paragraphs. This chapter consists of four main parts, namely: research guidelines and choices (section 3.1), data collection (section 3.2), image classification (section 3.3) and object detection (section 3.4). All the necessary code with respect to this study (pre-processing, image classification and object detection) can be found on https://github.com/ajv95/floatingplasticdebrisdetection.

3.1 Research guidelines and choices

Firstly, the simplest computer vision technique, namely image classification was applied, to assess whether plastic debris can be detected in images. For this purpose, the performances of a custom-made baseline model were compared against those of five state-of-the-art and widely applied models presented in the literature. Most commonly a technique named transfer learning is applied when using these models. This encompasses the idea of utilizing models, that were already pre-trained on larger benchmark datasets such as ImageNet, on a completely new classification task. Transfer learning can be applied due to the property of CNNs, whereby the first layers of a network learn standard features such edges, textures, or corners. Deeper layers learn more sophisticated and specific features which are dependent on the dataset and the task at hand [1]. By applying transfer learning the use of available task-specific data can be optimized since training can be done for deeper convolutional blocks or the classifier or regressor. This method is especially useful for the reduction of computational time and datasets with a limited number of samples.

To have a comparative benchmark and ascertain whether pre-trained models are indeed superior in terms of performance, these models were tested against a newly built model, but also tested against the same architectures trained from scratch. The best performing model was then selected for assessing the influence of instrumental and environmental factors.

Since image classification has its clear limitations due to its impractical nature of merely assessing the presence of plastic objects in an image, object detection was applied in a subsequent step.

The main disadvantage of object detection algorithms is however, that the labelling procedure is much more time-consuming compared to the labelling for image classification. In object detection labelling entails to annotate all objects of interest in an image by means of a rectangular box. For image classification, the images merely need to be sorted according to their class. To reduce the amount of time spent on labelling, practical monitoring aspects for the detection of plastic debris, such as weather conditions, camera height and camera angle were assessed by means of image classification. The evaluation of weather- and camera specific parameters is an important aspect since no other comprehensive study exists, that explores the influence of weather- and camera specific parameters on the detection of plastic debris in waterways.

For the quantification of plastic and the evaluation of the generalization ability for other locations and other image sources, YOLOv4 ('You Only Look Once') was used. YOLOv4 was chosen as an object detection model since it only requires one conventional GPU for operating in real-time. The problem with the most accurate CNNs nowadays is that these networks do not operate in real-time, requiring a large number of GPUs for training [40]. The

computational resources for this study were limited to the Graphical Processing Units (GPUs), offered by Google Colaboratory. This platform uses a 12GB NVIDIA Tesla K80, T4, P4 and P100 GPUs, that can be utilized for 12 hours continuously. GPUs are predominantly used for deep learning applications, as these units can simultaneously process several parallel processes due to the larger number of cores compared to conventional Computer Processing Units (CPUs). This allows more efficient and accelerated computation of parallel processes. YOLOv4 was therefore a suitable object detection algorithm for this study, since it only requires one conventional GPU, which is offered as free service by Google Colaboratory.

Furthermore, it was chosen not to utilize segmentation techniques such as semantic or instance segmentation since a study by [41] suggests, that some of the best methods require nearsupercomputer processing power for the training phase.

Globally, the research was divided into two stages (Figure 9), which summarize the previously discussed guidelines and choices for this study. The key aspects of the two flowcharts, related to the data collection, image classification and object detection will be described in the following sections.

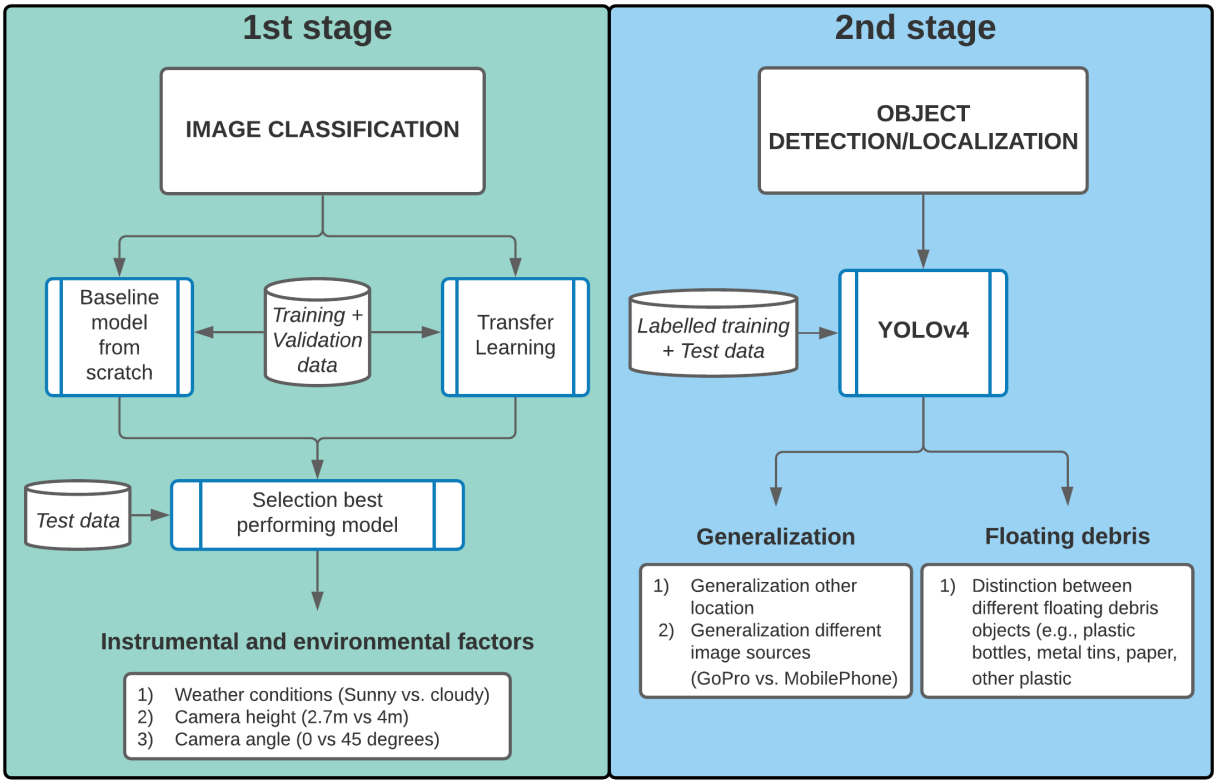


Figure 9 Research segregation into two stages, whereby the 1st stage focusses on image classification with the comparison a baseline model and pre-trained models followed by the assessment of instrumental and environmental factors (weather conditions, camera height and camera angle) with the best performing model. The 2nd stage focusses on object detection, which aims to evaluate the generalization for other locations and images sources and the quantification of plastic debris

3.2 Data collection

This section aims to describe the data collection for the controlled and natural environment. For data collection in the controlled environment experiments were carried out on the premises of Green Village, which is located on the campus of Delft University of Technology. For data from the natural environment a dataset by [10] using static cameras at bridges in Jakarta (Indonesia) was used.

3.2.1 Dataset controlled environment

Figure 10 Impression of plastic debris objects used for the experiments at Green Village(1: Plastic bottles, 2: Metal tins, 3: Various plastic objects, 4: Plastic bags, 5: Paper)

The main idea of the experiments was to record floating plastic debris in a water body with different image sources. Since the water body at the Green Village is stagnant with no plastic flux, floating objects were gathered with volunteers from canals in Alkmaar (Netherlands) to have a real-life representation of floating plastic debris from waterways. Furthermore, plastic from household waste was used to create more variability in the dataset. In total, 626 plastic debris objects were used, whereby 270 items were used for training data and 356 items for test data. An impression of the objects is given in Figure 10.

Two static cameras were used in the experiments, namely the GoPro HERO4 Silver and GoPro MAX 360 (Figure 11). The GoPro HERO4 was either mounted at location (2) or (4) and the GoPro MAX360 at location (1) and (3). The location was dependent on whether measurements were carried out for the centre or the right bank of the water body (Figure 11). Lastly, the mobile device Huawei P30 Pro was also used to gather images.

Since the water body is stagnant, wind was the main transport mechanism and therefore the floating barrier had to be placed based on the prevailing wind direction. Furthermore, the cameras and an external power supply were fixed to a wooden plank. All cameras used the following settings for taking videos (Table 1):

Resolution	Frames per second (fps)	Field of view (FOV)
1080p	24	Linear

Table 1 Camera settings for GoPro HERO4 silver and GoPro MAX 360

A resolution of 1080p = 1920x1080 (16:9) ratio was used. This resolution had to be used, since this was the highest available resolution for the GoPro HERO4 Silver. The frames per second (fps) were set to the lowest possible value to reduce the required storage capacity. Furthermore, a linear field of view (FOV) was used that applies corrections to the fisheye distortion caused by the lenses. By using this FOV, horizons and verticals get straightened by also narrowing the perspective. This is an advantage compared to the wide FOV, whereby the fisheye view creates curvature of straight lines, leading to the distortion of objects and shapes.

Overall, 11 days between February and April were chosen to conduct experiments. The selection of experimental days was dependent on the three following factors: precipitation and the direction of wind. Measurements could not be carried out during rainy conditions since the GoPro HERO4 Silver was not waterproof. Lastly, the prevalent wind direction had to either be from N/NW or S/SE direction for plastic transport to occur.

Ultimately, the influence of instrumental and environmental factors was assessed by evaluating different weather conditions (sunny vs. cloudy), camera heights (2.7m vs. 4.0m) and camera angles (0 degree vs. 45 degree) for which data was gathered accordingly. The mounting height of a camera in the Green Village was limited by the maximum height of the bridge (4.0 m). Next to mounting the static cameras perpendicular to the water surface, a camera angle of 45 degrees was also tested. This was done to assess whether different angles can be used for the detection of plastic debris, potentially providing more flexibility with respect to taking images and videos.

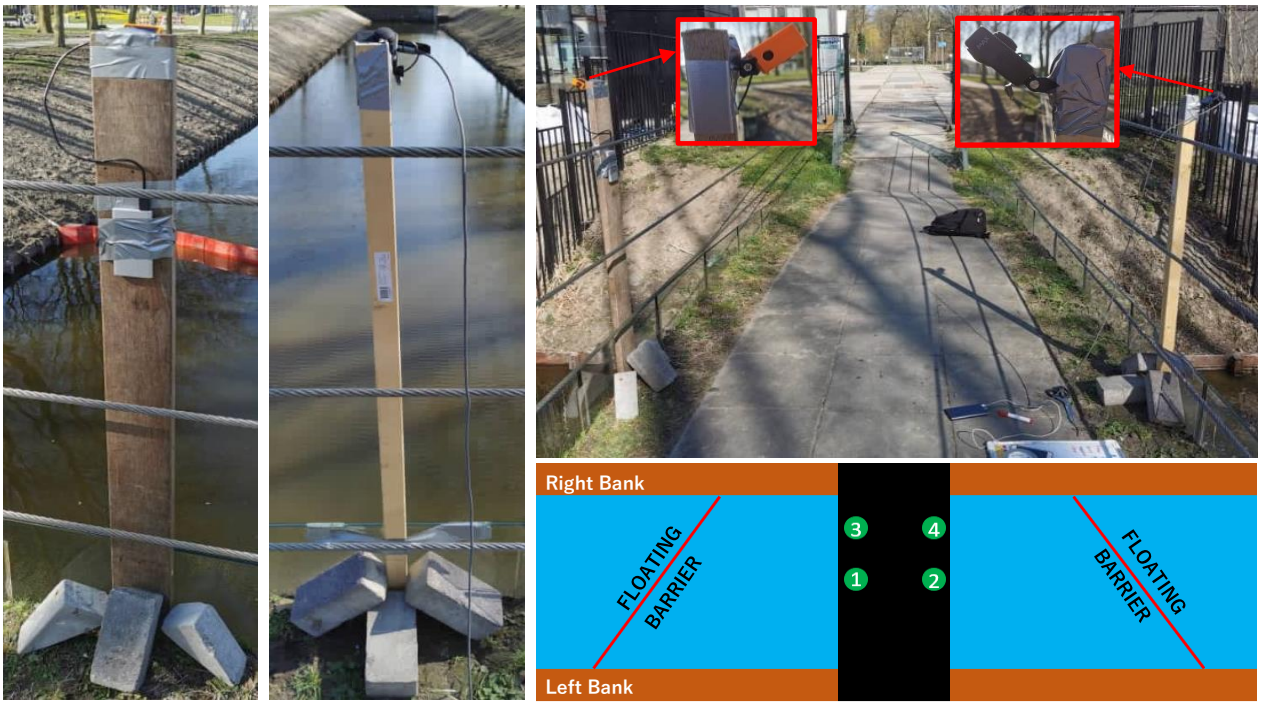


Figure 11 Experimental setup at Green Village. The GoPro Hero was mounted on the bridge on location (2) + (4) and the GoPro Max on location (1) + (3), depending on whether the measurements were carried out for middle of the water body or on the banks. The cameras and an external power supply were fixed to a wooden plank. The floating barriers were used to intercept the floating plastic. This example shows both cameras at a height of 4 meters and an angle of 45 degrees.

Lastly, no other study addressed the influence of weather conditions on the detection of floating plastic debris, therefore the detection ability during sunny and cloudy conditions was also evaluated.

Once the video material had been gathered, the data was pre-processed before it could be used in a model. The data pre-processing for image classification and object detection can be found in section 3.3.1 and 3.4.1 respectively. The next section will discuss the data collection from the natural environment.

3.2.2 Dataset natural environment

For the evaluation of generalization for other locations, a dataset from a study by [10] was used, whereby a static camera was mounted on five bridges at different waterways in Jakarta (Indonesia), Data collection took place from 30th April to 12th May 2018. At each location, the camera model *Dahua Easy4ip IPC - HDBW1435EP - W* recorded continuous video sequences of 1080p and 10 fps. A part of the dataset is available on [42]. The height of the measurements for this dataset was 4.5m. This refers to the monitoring location BKB-Grogol from the study by [10].

3.3 Image classification

This section describes all steps that were included in the image classification stage. This comprises the pre-processing of data, model construction from scratch, transfer learning and relevant accuracy metrics to evaluate model performances.

3.3.1 Data pre-processing

After the video material had been gathered at the Green Village, the audio files of these videos had to be extracted (Figure 12). This was done with *FFmpeg*, an open-source software that consists of a vast number of libraries specialized in processing video, audio and multimedia files. The extraction of the audio was necessary, because if the raw video was processed with the *VideoToImage* tool, only the first few frames were saved. This behaviour suggested, that when a certain frame is reached, that the algorithm switches to a different non-video stream track within the processed video file. After a certain frame is reached, the VideoToImage tool merely iterates over empty frames, explaining the fact that no images are saved.

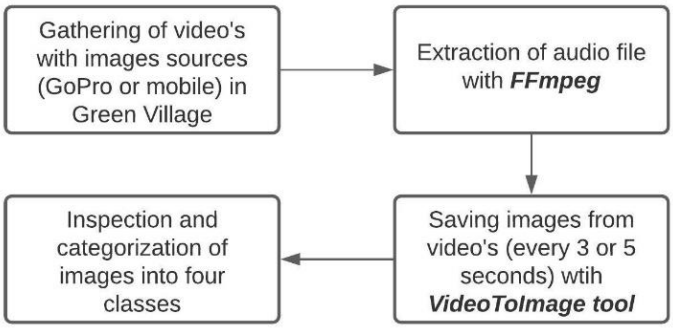


Figure 12 Workflow of data pre-processing for image classification

After extracting the audio file, the videos could be processed in the *VideoToImage* tool. With this tool, images could be saved at predefined framerates. All the videos were recorded with 24 fps and since images were saved either at 3 s or 5 s intervals, the frames per second were equivalent to 72 or 120 fps respectively. The chosen time interval was dependent on the velocity of plastic debris in the water, whereby 72 fps were chosen for higher velocities

(higher presence of wind) and 120 fps for lower velocities (lower presence of wind). The following calculations gives an example for the number of output images generated for a 20 min video and intervals of 72 fps:

$$Number\ of\ frames = \frac{24\ [fps] * 20\ [min] * 60\ \left[\frac{s}{min}\right]}{72\ [fps]} = 400$$

After generating the images per video, these images were inspected and manually categorized according to four different classes (Table 2). The categorization with four classes in this study, follows the same categorization as used by the Citizen Science platform CrowdWater [43]. Furthermore, this categorization was also used to evaluate whether image classification techniques, next to correctly predicting the presence of plastic debris in an image, can also give a rough indication about the number of plastic debris objects in an image.

Class	Description	Number of objects
0	No plastic	0
1	Little plastic	1-2
2	Moderate plastic	3-5
3	Lot of plastic	6-10

Table 2 Class, description and number of plastic debris objects for this study

After categorizing suitable images according to the predefined classes, the categorized images were re-assessed to guarantee label consistency. But what could be defined as suitable images for this research?

An article by [44] suggests, that one of the benchmark datasets for image classification ImageNet, has an estimated label error rate of 5.8%. This is equivalent to 812.000 wrongly classified images, if the entire dataset of 14.000.000 million images with 1000 classes is considered. Model performances using this flawed dataset therefore do not have any explanatory power or informative value, since these models were applied on a flawed dataset. Taking this as an example, it is therefore important to set guidelines regarding the labelling procedure. This will ensure transparency and consistency, such that in future work the labelling procedure can easily be reproduced Especially if multiple people conduct labelling on the same dataset, it is important that the labelling is consistent.

For this study, Figure 13 presents the image choices that had to be made for the aggregation of the dataset. Scenario (1) is the simplest inclusion of an image to the dataset, where no occlusion, overlap or cut-off from the image boundary are present. Images whereby multiple plastic debris objects overlapped, such as in scenario (2), were also included, since it can also occur in the natural environment that floating objects are intermingled. A choice had to be made regarding the degree of object cut-off at the boundary of an image. If only a small fraction of the object was visible (scenario (3)), the image was not added to the dataset. On the other hand, if the majority of the object crossed the boundary of the image, the image was included. Including this type of images has the potential advantage, that the model does not only learn the distinct shapes of the objects, but that new appearances of object shapes get created by a certain cut-off on the image boundary.

Lastly, scenario (5) and (6) represents two images from the same experiment, with a 30 s time difference between them. The images clearly show that a hardly discernible displacement of the objects has taken place. If the location of objects has not or hardly changed compared to preceding images these images, as depicted in scenario (6), will not be included in the dataset. An inclusion of these images would entail, that the model

technically trains on almost identical images, since the location of the objects barely changed. A representation of selected images per class for the Green Village data can be found in Appendix B.1.

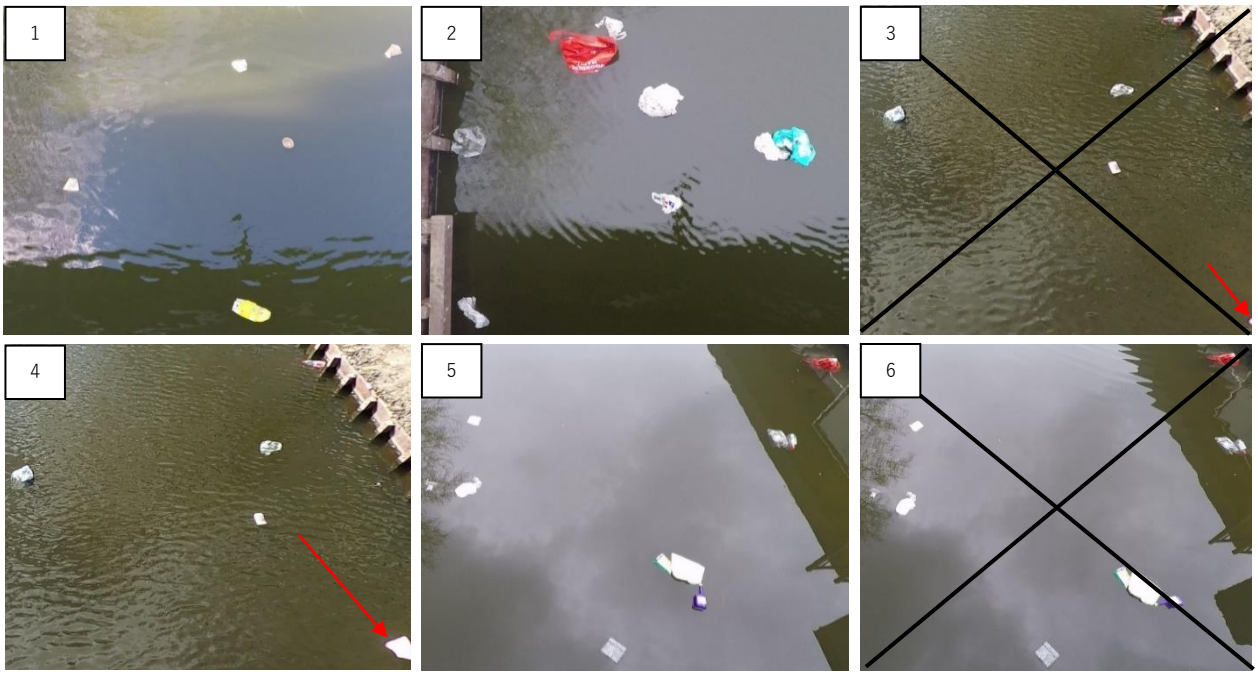


Figure 13 Visualization of image choices for dataset, whereby crossed out images ((3) and (6)) represent the type of images that were omitted from the dataset

3.3.2 Datasets

After establishing the guidelines with respect to the categorization of images, the assembled datasets will be briefly discussed. For the construction of the baseline model and the assessment of available state-of-the-art models, training and validation data from 2.7m/0 deg was used (*Train 1* and *Validation 1* in Table 3). This was done, because the most amount of data was gathered for this setting.

Dataset	Characteristics	Number of images
Train 1	2.7m/0 deg	4005
Train 2 (Data Augmentation)	2.7m/0 deg	7948
Train 3	All heights + angles	5915
Validation 1	2.7m/0 deg	801
Test - Cloudy	2.7m/0 deg	479
Test - Sunny	2.7m/0 deg	623
Test 1	2.7m/45 deg	689
Test 2	4.0m/0 deg	621
Test 3	4.0m/45 deg	345

Table 3 Training, validation and test data for the entire image classification stage

A validation or development set is most commonly used for the selection of the most suitable model and used for the tuning of hyperparameters. Furthermore, the effects of data augmentation were also assessed for the construction of the baseline model. Therefore, the 4005 images recorded from 2.7m/0 deg were augmented. This

resulted in another training dataset (*Train 2 - Data Augmentation*) with 7948 images. The performed data augmentation operations are described in section 3.3.4. Based on the highest validation accuracy, the best performing model was then chosen. This model was subsequently used to assess the influence of different instrumental and environmental factors. This refers to all the test datasets listed in Table 3. To also assess the influence of a larger dataset with different heights and angles on the test accuracies, the original dataset *(Train 1)* was expanded with 1910 images from 2.7m/45 deg, 4.0m/0 deg and 4.0m/45 deg. This resulted in a new training dataset with 5915 images from all heights and angles *(Train 3).*

3.3.3 Custom-made baseline model

To establish a minimum model performance, to which other available models can be compared to, a baseline model needed to be established. As a starting point, the architectural principles of the VGG16 model were used, which is a CNN with 138.423.208 million parameters developed by [45], that achieved a test accuracy of 92.7% on the ImageNet dataset.

The main principle of the VGG model architecture involves the stacking of convolutional layers with 3x3 kernel-sized filters, followed by a maximum pooling layer. Jointly, these layers can be considered as 'blocks' which can be repeated after each other, whereby the number of filters per block is increased (e.g., 32, 64, 128, 256 or 512) with increasing depth of the network. The major problem with VGG is the extremely long computation time due to the large number of parameters. Since the computational resources in this study were limited to one GPU, a baseline model had to be built that had significantly less parameters, but still achieves a satisfactory model performance. The following stepwise approach for building a model from scratch, taking the VGG model as a guideline in terms of architecture, was followed (Figure 14):

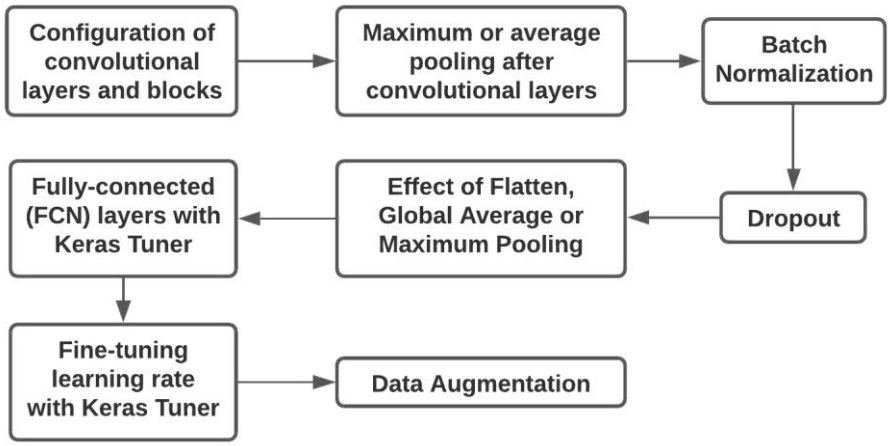


Figure 14 Flowchart describing the stepwise approach for custom-made baseline model in this study

An in-depth description with respect to the buildings blocks of a CNN can be found in section 2.2.2 'Convolutional Neural Networks (CNNs). The models were constructed in the Python 3 programming language by using the deep learning libraries TensorFlow and Keras.

Due to the limited available processing power, the original rectangular 1920x1080x3 images were resized to a commonly used square input of 224x224x3 for CNNs. A batch size of 32 was used, since smaller batch sizes are noisier compared to larger batch sizes and therefore offer more potential for regularization.

The Adaptive Moment Estimation (Adam) optimizer, which is a stochastic gradient descent method proposed by [46] was used. The Adam optimizer includes an adaptive learning rate, governed by two decay rates (β_1 and β_2). These parameters determine to what extent the gradients from the previous batch are used to update the weights after the current batch, by giving a timescale during which the learning rate decays. The authors of [46] suggest values of $\beta_1 = 0.9$ and $\beta_2 = 0.999$, which will also be used in this study. Higher beta parameters entail, that previously calculated gradients influence the weight update to a lesser extent. The initial learning rate was set to 0.001. Next to the Adam optimizer, the sparse categorical cross entropy was used as a loss function, which is commonly used for multilabel classification problems. The number of epochs was arbitrarily set to 30.

Before feeding the data to the model, the RGB images had to be pre-processed by scaling the input to values between 0 and 1 by dividing the pixel values of the images by a value of 255. This normalization is needed to ensure that each input has a similar distribution, which accelerates the process of convergence. After converting the images to a suitable input, different model configurations were evaluated.

A more precise description with values and steps for all operations with respect to the baseline model are listed in Appendix C.1. For every step, the best model was selected based on the losses, accuracies, degree of overfitting and the number of parameters, since the aim was to build a computationally efficient and accurate model.

Firstly, different configurations of convolutional blocks were assessed, followed by evaluating the influence of either a maximum or average pooling layer after the convolutional blocks. Therefore, five different configurations of convolutional layers and blocks were tested.

The implementation of batch normalization layers aimed to accelerate and stabilize the process of convergence. Next to the main regularization technique dropout, batch normalization can also have a regularizing effect. It is common practice to place batch normalization layers after every convolutional block to standardize the non-linear output. This practice was also applied in this study. The tuneable parameter in batch normalization layers is the momentum parameter, that determines how much the moving average of the previous batch influences the calculation of the current batch normalization. High momentum values encompass slow learning of the moving mean due to the higher lag. Values ranging from 0.5 – 0.9 were tested for the momentum. For the dropout layers, probabilities for omitting random nodes ranged from 0.2 – 0.7. Research by [47] shows, that model performance decreases if dropout layers are applied before batch normalization. A harmonic approach to combine both of these strategies, as suggested by the authors of [47], is to apply dropout after the last batch normalization block, which will also be adopted in this study.

Before the feature maps from the last convolutional layer are fed to the classification stage, commonly every single feature map is collapsed into a 1D vector, before being passed to the FCN layers. However, there are other techniques such as Global Average or Maximum Pooling, which take the average or maximum value for each single map respectively, without converting the entire feature map into a single column. Next to that, a hybrid global pooling layer by combining global average and maximum was tested, which is a simplified version of a global pooling method as suggested by [48].

The addition of more than one dense layer in the classifier can enhance the networks ability to classify features, that were extracted in the feature extraction stage. The number of hidden layers and number of neurons in these layers were determined with the Keras Tuner. This is a tool, which determines the optimal set of hyperparameter for a model. Within the tuner a Random Search algorithm was used, which randomly samples hyperparameter

combinations for testing the model performance. The search space was defined by combinations of 1 – 4 dense layers with varying number of neurons (16, 32, 64, 128, 256 or 512).

Lastly, the optimal learning rate was also evalulated with the Keras Tuner. Therefore, the search space was defined by four different learning rates, namely 0.1, 0.001, 0.0001 and. 0.00001.

3.3.4 Data Augmentation

Data augmentation is a technique that artificially expands the training data by random image transformations of the existing samples [1], which can be used to reduce overfitting and thus contributes to the improvement of generalization power of the model. For this study four different techniques were applied, namely: horizontal and vertical flipping, brightening, darkening and the addition of random salt and pepper noise. Techniques such as e.g., cropping, rotation or zooming were not assessed because this could lead to the omission of plastic debris objects in an image and consequently an undesirable label transformation [49], if the incorrect parameter values are chosen for these operations. To assess the effectiveness of data augmentation, up to two transformations were done for *Train 1*. This resulted in a total of 7948 images (Table 3). The image transformations were done via a machine learning platform named Roboflow. One image example of data augmentation is given in Figure 15 .

Figure 15 Implementation of data augmentation techniques (Left: vertical flipping, noise injection and slight brightness increase, Right: larger noise injection and larger brightness increase)

It is important to mention, that image transformations could contain the application of more than one technique. An augmented image could for example be vertically flipped, have a brightness adjustment of +15% and a salt and pepper noise injection of 5 pixels. The number of applied techniques per images were randomly generated.

Horizontal and vertically flipping was done to make the model more robust towards object orientation. This technique has proven to be successful on benchmark datasets, such as ImageNet or CIFAR-10. Furthermore, a higher accuracy could also be achieved in the study by [10] through horizontal and vertical flipping, which makes this technique also interesting for this research. A 50% probability existed for either horizontal or vertical flipping. Furthermore, it was chosen to apply a range of 0% to +35% for brightening and – 35% to 0% for darkening. Variation in brightness can help the model to be more resilient towards changes in lighting or camera settings.

Lastly, a study by [50] found, that blur and noise have the most adverse impacts on image classification tasks. Noise can be ignored by the human eyes, however slight pixel changes can have a detrimental influence on the networks prediction ability. The injection of noise could potentially help, to make the model more robust towards camera artifacts, which is an undesired modification of data caused for example by the compression of images. A study by [49] also suggests, that the addition of noise can help CNNs to learn more robust features. It

was chosen only to assess the implementation salt and pepper noise by applying salt and pepper noise injection between 0 to 10% of the pixels.

3.3.5 Transfer Learning

For the application of pre-trained models with transfer learning, three main strategies can be distinguished:

1) **Training with pre-trained weights**
   For this strategy, the model is instantiated with pre-trained weights acquired by training on another dataset (e.g., ImageNet). To ensure that the model parameters are not updated during training the layers are frozen, meaning that no model parameters are trainable. The old classifier of the model is substituted with a new classifier, that relates to the new dataset and the problem at hand. This is also the case for the following two strategies.
2) **Fine-tuning**
   This method consists of unfreezing the deeper layers of the model, whilst keeping the remaining model layers frozen. The intention behind this method is that earlier layers learn more generic and simple features which can be found in most of the datasets. Deeper layers learn more complex patterns, which are specifically related to the dataset that is used for training.
3) **Training from scratch**
   This approach implies the re-training of the entire model, by making every layer in the network trainable. Consequently, the model does not make use of pre-trained weights.

For this study, it was chosen to train with pre-trained weights and training from scratch. The process of fine-tuning is the most time-consuming strategy, since it comprises a trial-and-error process of unfreezing a number of layers, connected to frequently running a model to assess the performance.

The selection procedure of five state-of-the-art models was based on the number of parameters these models contained. Overall, two smaller, one medium-sized and two larger models were selected to compare the performance of existing deep learning architectures with the baseline model. The models including their size, top 1-accuracy, top 5-accuracy and number of parameters are given in Table 4. The models were originally proposed for the ImageNet dataset and the listed top 1-accuracy and top 5-accuracy were achieved by training on ImageNet.

Model	Size	Top-1 accuracy	Top-5 accuracy	Parameters
ResNet50	98 MB	0.749	0.921	25,636,712
InceptionV3	92 MB	0.779	0.937	23,851,784
DenseNet121	33 MB	0.750	0.923	8,062,504
MobileNetV2	14 MB	0.713	0.901	3,538,984
SqueezeNet	0.5 MB	0.572	0.803	1,235,496

Table 4 Characteristics (size, top-1 accuracy, top-5 accuracy and parameters) for state-of-the-art models used in this study

The top-1 and top-5 accuracy metrics indicate whether the model prediction with the highest probability exactly corresponds to the ground truth label, or whether the five highest probabilities match the true label. It can be seen that there is a trade-off between the computational speed (size and parameters) and the accuracy of the model, if e.g., SqueezeNet or MobileNetV2 are compared to the other three models. It is noticeable, that DenseNet121

outperforms ResNet50 in terms of accuracy, although the model is a third of the size of ResNet50. A more precise explanation with respect to all five model architectures can be found in Appendix C.2 – C.7. All models, except for SqueezeNet, were available in Keras alongside the pre-trained weights. For the implementation of SqueezeNet with its pre-trained weights, the code from [51] was used and adapted.

3.3.6 Dealing with class imbalances

Class imbalances within the datasets had to be addressed, which was problem for the data in this study. Imbalanced datasets can cause problems in classification tasks, since balanced datasets exceed imbalanced datasets in terms of model performance, as suggested by studies from [52] and [53]. This research applied a cost sensitive technique, that addresses the learning process itself by introducing class weights, entailing a higher penalty to the loss function if an underrepresented class is wrongly classified. This forces the model to improve on learning these classes [52]. The following formula was applied to assign class weights to all four classes:

$$Class\ weight_i = \frac{\left(\frac{Total\ number\ of\ images}{Number\ of\ images\ per\ class}\right)}{4} \tag{11}$$

whereby i represents the respective class for which the class weight is calculated. Other methods for addressing class imbalance, such as sampling techniques or one-class learning [52] [53], will not be discussed in this report. The class weights were applied for every image classification dataset in this study.

3.3.7 Accuracy metrics

Model performance for classification problems is commonly based on the confusion matrix and the different metrics that can be derived from it. For each image, the four following scenarios can occur, if it is assumed that the processed image belongs to class '0' – No plastic:

I.	Actual = Class 1, 2 or 3	Predicted = Class 1, 2 or 3	**TRUE NEGATIVE**	**(TN)**
II.	Actual = Class 1, 2 or 3	Predicted = Class 0	**FALSE POSITIVE**	**(FP)**
III.	Actual = Class 0	Predicted = Class 1, 2 or 3	**FALSE NEGATIVE**	**(FN)**
IV.	Actual = Class 0	Predicted = Class 0	**TRUE POSITIVE**	**(TP)**

The confusion matrix is attained by adding occurrences of TN, FP, FN and TP and placing them in a single matrix. The rows correspond to the ground truth classes (e.g., the actual outcomes) and the columns represent the predicted classes. By combining the confusion matrix cells, more sophisticated metrics can be obtained that evaluate different features of the classification performance. The standard metrics to evaluate the classification performance are the accuracy, precision, recall and the F1-score.

The accuracy (defined in equation (12)) measures all the correctly identified images belonging to a certain class, whether it is a TN or a TP. Nevertheless, it can be a misleading metric if an imbalanced dataset is considered. If for example a dataset is considered whereby 5 out of 100 images contain plastic debris, a naive model predicting no plastic for all 100 images, will yield an accuracy of 95%. Therefore, precision and recall are used, which divides the accuracy metric into two different components.

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

The precision (defined in equation (13)) represents the ability of a model to detect only the relevant classes. This metric gives the percentage of the correct predictions per class.

(13) $$Precision = \frac{TP}{TP + FP} = \frac{TP}{All\ detections}$$

Recall on the other hand, corresponds to the ability of the model to identify all the relevant true labels per class (defined in equation (14)). It gives the percentage of TP detected amongst all relevant true labels. In an ideal model, the precision and the recall would be equal to 1, indicating that the model predicts every input correctly.

(14) $$Recall = \frac{TP}{TP + FN} = \frac{TP}{All\ true\ labels}$$

The F1 score (defined in equation (15)) gives the weighted average of the recall and the precision. This metric is especially useful for imbalanced data [1] [54]. If the dataset is highly imbalanced, as in the previously considered example, the classifier can reach a low misclassification rate if it chooses the majority class for every prediction. Considering the previous dataset and a model that is constructed for only predicting plastic debris objects in an image, the precision would be equal to 0.05 (TP = 5, FP = 95) but the recall would be equal to 1 (TP = 5, FN = 0). Consequently, the F1-score would be equal to 0.095, balancing the high discrepancy between precision and recall. This also emphasizes the fact that not just one metric should be considered.

(15) $$F1 = \frac{2\ x\ Recall\ x\ Precision}{Recall\ x\ Precision}$$

3.4 Object detection and localization

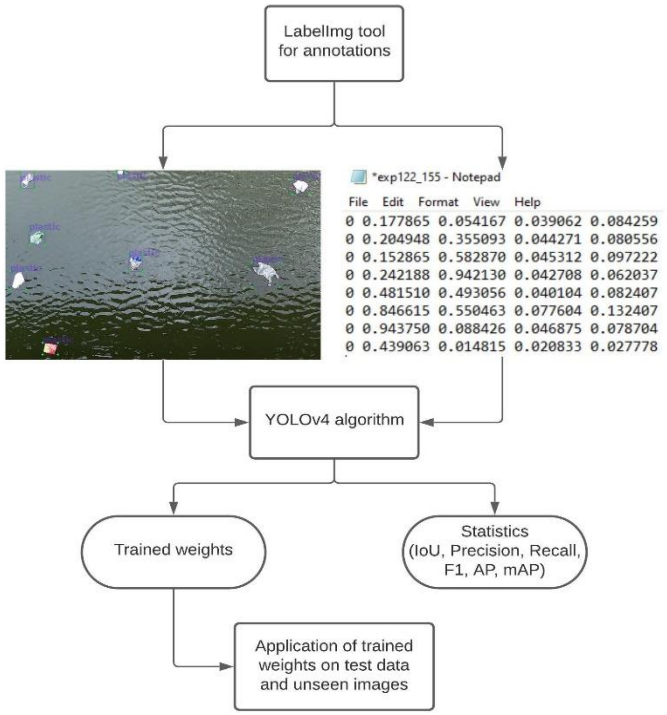


Figure 16 Workflow for object detection stage depicting the labelling procedure for annotating images which are used as input for the YOLOv4 model. This generates trained weights and statistics that are used for the assessment of the model performance. The trained weights were then used on test images for the detection of plastic debris objects.

The entire workflow for the object detection stage is depicted in Figure 16. Due to the constraints of image classification, object detection and localization was applied, to make the detection of plastic debris objects viable for practical applications. This section describes the methodology used for the implementation and evaluation of the YOLOv4 model on different datasets. Firstly, the data pre-processing will be described, followed by the application and model structure of YOLOv4 and the accuracy metrics to assess the model performance. The technical background for object detection algorithms in general and specifically the YOLOv4 model can be found in Appendix D.1 and D.2

3.4.1 Data pre-processing

This section describes the data processing of the Green Village data (section 3.4.1.1) as well as the data from the study by [10] (section 3.4.1.2), which was used to assess the generalization ability of the YOLOv4 model.

3.4.1.1 Green Village data

The labelling procedure for object detection comprised the selection of readily categorized images from the image classification (section 3.3.1) stage. Images from the moderate (class '2') and lot (class '3') of plastic class were selected to conduct annotations. To train the YOLOv4 algorithm on recognizing and localizing plastic debris, these objects had to be labelled per image. This was done by a free and open-source graphical image annotation tool written in Python, named *Labellmg* by [55]. With this tool rectangular bounding boxes can be drawn to localize the desired objects in an image. For the YOLO labelling format, a .txt file with an identical name corresponding to each

image file was created. Each .txt file contains the necessary information about the annotations made for every image, given in the following format:

<object class> <x center> <y center> <width> <height>

The file contains the object class, representing the object present in the bounding box, the coordinates (x and y center) for the centre of the rectangle and the width and height of the bounding box (Figure 16). A YOLO label file can contain multiple object labels, for which each new line represents a new object. The annotations are normalized and lie within a range of 0 – 1, which is relative to the width and the height of the image.

Furthermore, labelling guidelines had to be established for the Green Village data, to guarantee consistent labelling. The guidelines for the dataset from this study are listed here below:

1) **Labelling the entirety of a plastic debris object**

   It is important do draw tight bounding boxes for every plastic debris object in an image. However, a small buffer should be left around the object, since this guarantees that the entire object is covered by the bounding box. Cutting of objects or not annotating objects at all, could lead to more false negatives in the model. Examples of a good and bad annotation is given by the green and red bounding boxes, respectively.

2) **Labelling of occluded and overlapping objects**

   In the natural environment it can also occur, that multiple plastic debris objects are occluded or overlap such as in a plastic hotspot. Therefore, it is also important to label multiple occluded objects, as if they are fully visible. This trains the model to understand the true boundaries of an object. Examples of a good and bad annotation for occlusion cases are given by the green and red bounding boxes, respectively.

3) **Labelling of image boundary objects**

   Partial objects, that were cut off due to the boundary of the image, were also labelled. However, this was dependent to what extent these objects were still visible in the frame. If only a small fraction of the object was visible in the image, the image was not included in the dataset. An exemplification of boundary cases that were included and that were not included in the dataset, are depicted below.

4) **Labelling images with presence of organic material**

For the model to distinguish between plastic debris objects and other floating materials such e.g., organic material in the form of wood or leaves, images with the presence of all these objects also had to be labelled. This also helps the model to become more robust with respect to inhomogeneous conditions, where plastic debris are not the only objects present in the water.

It is important to mention, that only images from 2.7m (0 degrees) were annotated, because the assessment of model performance with respect to the height was carried out with image classification. The annotations were firstly made with the following four classes: plastic bottles (0), other plastic (1), paper (2) and metal tins (3). The distribution of annotations amongst all classes for the training and testing data is presented in Appendix C.1. It can be seen that the data is largely unbalanced, especially for the test data, The annotations for four classes could subsequently be easily converted to one class, if all the objects were considered as plastic debris. For the Green Village data with one class, following training dataset and test datasets were used (Table 5):

Dataset	Number of images	Annotations	Average (annot/image)
Training GV GoPro	1327	7076	5.4
Test GV GoPro	183	1104	6.0
Test GV Huawei	238	985	4.1

Table 5 Datasets for Green Village with number of images, annotations and average annot/image

3.4.1.2 Dataset 'Automated River Plastic Monitoring Using Deep Learning and Cameras'

The A-RPM data was used for an object detection algorithm in the research by [10], therefore the annotations were already available. Overall, 472 images were available for training data and 53 images for test data. This split was predefined by [42] and used for this study. The floating plastic throughput per image was much larger for this dataset, compared to the Green Village data (Table 6).

Dataset	Number of images	Annotations	Average (annot/image)
Training A-RPM	472	9823	20.8
Test A-RPM	53	1208	22.8

Table 6 Dataset A-RPM (Automated River Plastic Monitoring) with number of images, annotation and average annot/image

3.4.2 Application of YOLOv4 model

The source code from Alexey Bochkovskiy [40], one of the creators of YOLOv4, was used and customized for the application of YOLOv4 on the detection of plastic debris. The model was run through 'Darknet', which refers to an open-source framework for neural networks. Additionally, another repository by [56] was utilized. The described configurations are based on the suggested guidelines by Alexey Bochkovskiy, which can also be found via [57].

The YOLOv4 model has readily been trained on the COCO dataset, which is a large-scale object detection dataset with 80 classes. These pre-trained weights (162 MB) were used to train the model on the detection of plastic debris objects. There are a few adjustable parameters based on the recommendations by Alexey Bochkovskiy, which depend on the number of classes that are used for object detection. These will be briefly explained in the following paragraphs.

Firstly, the image sizes were set to 416x416. This can be any value multiple of 32, however the minimum should be 416x416. Larger image sizes would entail prolonged computational time. The batch size was set to 64 with a subdivision of 16. The subdivisions stand for the number of mini-batches made from a batch. In this case 4 images per mini-batch are forwarded to the GPU for processing. This process will be repeated 16 times until the maximum batch size is reached, which will then be followed by a new iteration of 64 images. The number of maximum batches (or maximum number of iterations) was set to 6000 for one class and to 8000 for four classes. It is suggested, to calculate this by multiplying the number of classes by 2000, but not having less than 6000 and not more than 10000 iterations [57]. Furthermore, the steps were set to 80% and 90% of the maximum batch, respectively. These steps indicate a learning rate change, whereby the learning rate is multiplied by a predefined scaling factor of 0.1.

Furthermore, the number of filters in each of the three convolutional layers before each YOLO layer, are calculated by (classes+5) x 3. For one class the number of filters had to be set to 18 and for four classes consequently to 27. Lastly, the learning rate (0.001), momentum (0.949) and decay rate (0.0005) were predefined values. Once the training and test data was uploaded to Darknet and all the parameters were adjusted, the YOLOv4 model simulations could be carried out. Based on the highest test accuracy, the model weights were also saved. After completion of the computation, the model was evaluated based on the prediction of bounding boxes on test images and unseen images containing plastic debris. Furthermore, different accuracy metrics (Precision, Recall, IoU, F1-score, Average Precision (AP) and Mean Average Precision (MAP)) were used to assess the model performance (section 3.4.3).

3.4.3 Accuracy metrics

The Intersection over Union (IOU) metric is the area of overlap between the object and the detected box (intersection) divided by the area of the union of the object and the detected box (Figure 17 C). In this case a TP is the correct detection of a plastic debris object, whereby the detection with its associated IoU value needs to exceed a certain threshold. A FP is a wrong detection whereby the detection and its IoU value is below a certain threshold. This threshold value was set to 0.5. A FN refers to a ground truth label that was not detected. Another performance indicator in object detection is the average precision (AP), whereby the area under the curve (AUC) is calculated for the precision-recall curve. For every image with annotations, the model calculates precision and recall values for the precision-recall curve can be plotted, based on the collected detections for TP and FP. YOLOv4

uses a 11-point interpolation approach, which aims to describe the shape of the precision precision-recall curve by taking the average of the precision at eleven equally spaced recall values, ranging from 0 – 1 [58], given by the following formula:

$$AP = \frac{1}{11} \sum_{r \in \{0, 0.1, \ldots 1\}} \rho_{interpolation(r)}$$

whereby $\rho_{interpolation(r)}$ is defined as the maximum precision whose recall value is greater than r [58]. Consequently, the mean average precision (mAP) takes the mean of the AP amongst all classes.

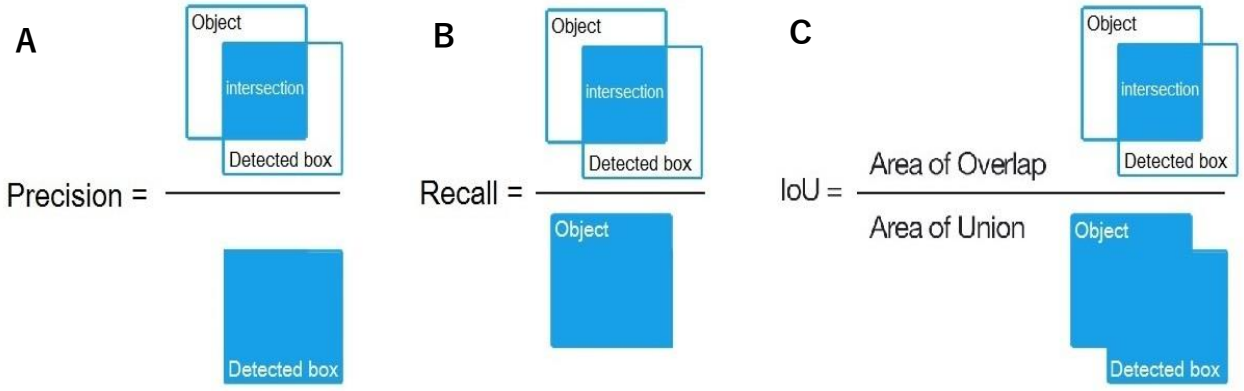


Figure 17 Accuracy metrics for object detection with precision, recall and Intersection over Union (IoU) (images taken from [58])

3.4.4 Generalization assessment

The generalization ability was assessed by the model performance for different dataset configurations. The combinations are listed in Table 7:

Abbreviation	Training set	Image source train	Test set	Image source test	Number of classes
GEN1	Green Village (2.7m/0 deg)	GoPro	Green Village (2.7m/0 deg)	GoPro	1
GEN2	Green Village (2.7m/0 deg)	GoPro	Green Village (2.7m/0 deg)	Mobile Phone	1
GEN3	Green Village (2.7m/0 deg)	GoPro	Green Village (2.7m/0 deg)	GoPro	4
GEN4	A-RPM	Dahua Easy4ip	Green Village (2.7m/0 deg)	GoPro	1
GEN5	Green Village (2.7m/0 deg)	GoPro	A-RPM	Dahua Easy4ip	1

Table 7 Different dataset combinations for the assessment of the generalization ability with characteristics (training set, image source training set, test set, image source test set and the number of classes)

4 Results

This section describes the attained results for image classification and object detection with YOLOv4.

4.1 Image classification

Firstly, the model results for the stepwise approach of building a baseline model (section 4.1.1) and training with pre-trained weights from ImageNet and from scratch for five different model architectures (section 4.1.2) will be presented. The abbreviations used for datasets in this section are listed in Table 3.

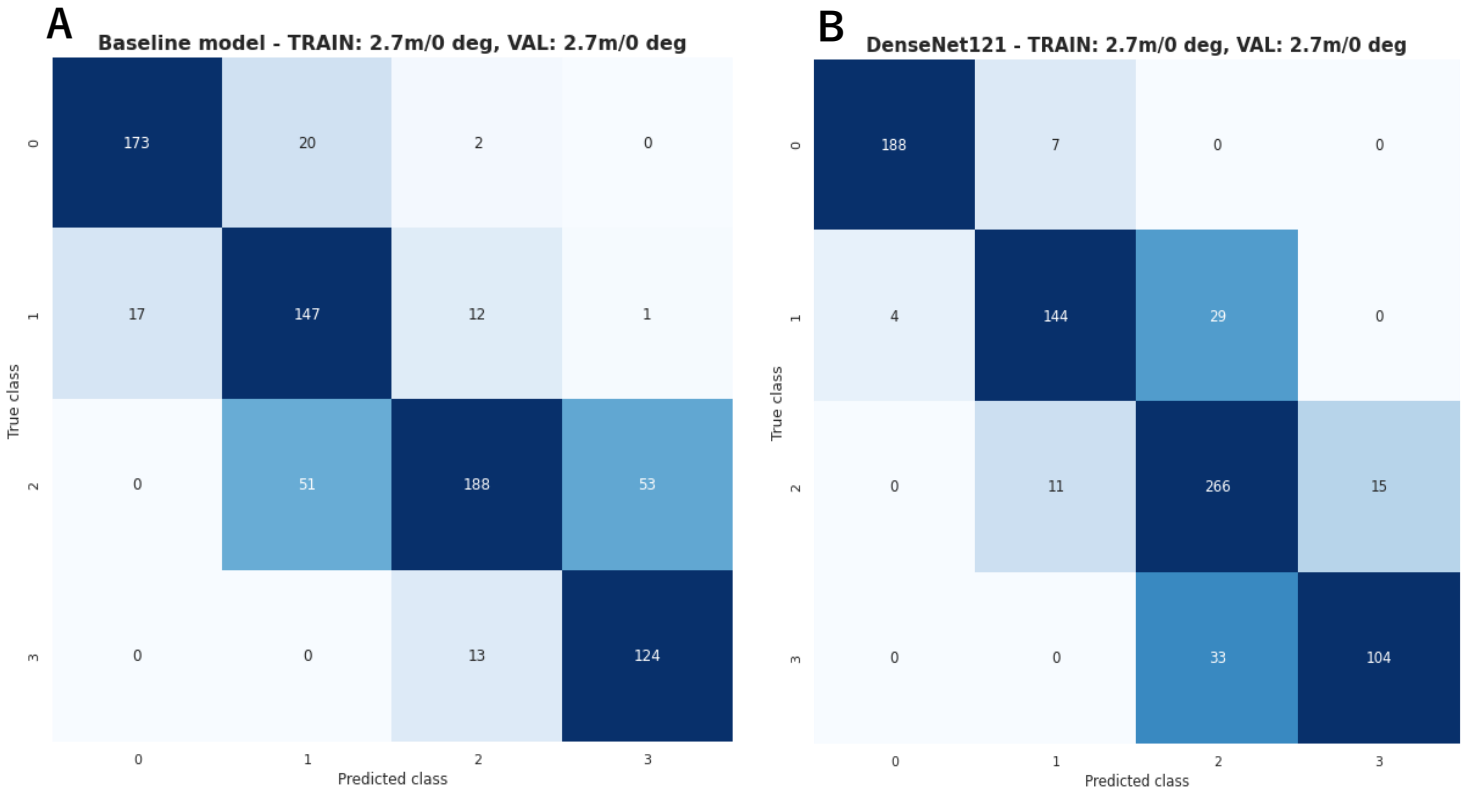


Figure 18 A: Confusion matrix for Baseline model (Train 1 and Validation 1), B: Confusion matrix for DenseNet121 (Train 1 and Validation 1)

	BASELINE MODEL				DENSENET121 MODEL		
	Precision	Recall	F1-score		Precision	Recall	F1-score
Class 0	0.91	0.89	0.90		0.98	0.96	0.97
Class 1	0.67	0.83	0.74		0.89	0.81	0.85
Class 2	0.87	0.64	0.74		0.81	0.91	0.86
Class 3	0.70	0.91	0.79		0.87	0.76	0.81
Accuracy			0.79				0.88
Macro Average	0.79	0.82	0.79		0.89	0.86	0.87

Table 8 Accuracy metrics (Precision, Recall, F1-score, Accuracy and Weighted Average) for Baseline model and Densenet121 model trained on Train 1 (2.7m/ 0 deg) and Validation 1 (2.7m/ 0 deg)

4.1.1 Baseline model

The results for the stepwise addition of the CNN building blocks and the final CNN model can be found in Appendix D.1. The confusion matrix of the baseline model is given in Figure 18 A and other relevant statistical metrics are listed in Table 8.

Firstly, the results show, that the stepwise addition of additional convolutional blocks increases the best validation accuracy from 33.66% to 71.18%. This indicates that the increase in number of filters through the addition of convolutional blocks, enables the model to learn more complex features related to the shape of plastic debris objects. VGG5 produced the best model results, by achieving the least amount of overfitting and losses and the highest validation accuracy with 71.18%. With the addition of pooling methods, maximum pooling gave the highest accuracy on the validation data with 75.9%.

Furthermore, due to overfitting regularization techniques such as Batch Normalization and Dropout had to be applied. The results show that the training process is slow for a momentum of 0.99, compared to the other evaluated momentum values. Batch Normalization did not produce the desired regularization to reduce overfitting, however the training process could be accelerated by setting the initial learning rate to 0.001. A momentum of 0.6 was chosen since this yielded the highest validation accuracy and lowest loss. With the implementation of Dropout, overfitting could significantly be reduced. Dropout values of 0.6 or 0.7 regularized too strongly since the training accuracy showcases an asymptotic behaviour from 15 epochs onwards. Therefore, a Dropout value of 0.5 was chosen since this also yielded the highest validation accuracy with 75.8%.

A Global Average Pooling layer was added before the FCN layer, because it generated the highest validation accuracy with 77.6%. In literature it is commonly suggested that global pooling operations should replace FCN layers and be fed directly into the output layer. However, this did not yield favourable results and therefore the Global Average Pooling was added before the FCN layers. Another reason for adding this pooling layer instead of a flatten layer is that the number of parameters could be reduced from 4.409.748 to 1.214.868.

Lastly, the configuration of the FCN layers was tested with the Keras Tuner. Up to this point, a FCN layer with 128 neurons was used. The search space for fine-tuning was defined by combinations of 1 – 4 FCN layers with varying number of neurons (16, 32, 64, 128, 256 or 512). The highest validation accuracy (78.9%) was attained with two FCN layers with 64 and 256 neurons respectively. After this step the construction and optimizing process of the baseline model was finalized.

The baseline model achieved an accuracy of 79%. The confusion matrix and statistical metrics show, that the largest amount of FN could be found for class 2 (Recall = 0.64) with 104 images being wrongly classified as class 1 or class 3. In 71 images, the model falsely predicted class 1 as the correct class (Precision = 0.67). Overall, class 0 performed the best with a F1-score of 0.90. Class 1 and 2 performed the worst, with a F1-score of 0.74.

4.1.2 Data Augmentation

To further improve the model performance, it was assessed whether data operations by means of data augmentation could further increase the accuracy. Results for data augmentation showed, that the validation accuracy was reduced by approximately 9% when trained on *Train 2*, compared to the baseline model trained on *Train 1*. With data augmentation, the model only performed better for class 1 (F1-score = 0.76), compared to training with the original dataset. All the other classes performed worse with respect to the F1-score.

4.1.3 Transfer Learning

The graphs with respect to the training and validation accuracies for each model can be found in Appendix D.2. The classifier for the transfer learning models was substituted with the classifier from the baseline model. The preceding architecture remained unaltered. Four different learning rates were tested (0.1, 0.01, 0.001 and 0.0001). The learning rates yielding the highest validation accuracy are listed in Table 9.

	PRE-TRAINED WEIGHTS			TRAINING FROM SCRATCH		
Model	**LR**	**Training time**	**Validation accuracy**	**LR**	**Training time**	**Validation accuracy**
ResNet50	0.0001	15 min	45.44%	0.0001	102 min	82.40%
InceptionV3	0.0001	12 min	54.56%	0.001	87 min	81.52%
DenseNet121	0.0001	15 min	62.67%	0.0001	112 min	87.60%
MobileNetV2	0.0001	6 min	60.67%	0.0001	63 min	75.03%
SqueezeNet	0.0001	3 min	66.70%	0.0001	19 min	73.91%

Table 9 Transfer learning results for training with pre-trained weights from ImageNet and training from scratch with learning rate, training time (Tesla K80) and validation accuracies

The results for the five selected models showed, that a large discrepancy exists between the validation accuracy for training with pre-trained weights or training from scratch. In every case, training from scratch yielded an improvement for the validation accuracy. The least improvement was found for SqueezeNet (+7.21%), whereas the largest improvement was found for ResNet50 (+36.96%). Overall, it can also be seen, that the larger models are superior in terms of validation accuracy, compared to the smaller models, if the model was trained from scratch. Nevertheless, the main disadvantage of training from scratch are longer computation times.

Lastly, it can be seen that all the models experienced severe overfitting (Appendix D.2). Since these are state-of-the-art models which also apply regularization techniques to prevent overfitting, the overfitting could potentially be attributed to an insufficient amount of training data.

The best validation accuracy (87.60%) was achieved with DenseNet121. This is an improvement of 8.70% compared to the validation accuracy of the baseline model. Overall, 702 from 801 images were classified correctly (Figure 18 B). Amongst all classes, class 3 has the lowest F1-score with 0.81 (Table 8). Images that did not contain plastic debris performed the best with a F1-score of 0.97. The true positives (TPs) for class 2 increased from 188 to 266 compared to the original dataset. Consequently, the recall greatly improved by 0.27 compared to the baseline model. However, the precision decreased marginally for class 2, as the false positives (FPs) increased. For class 3, the recall increased but the precision decreased. Compared to the baseline model, the DenseNet121 model performs better for every category, if the F1-score is considered.

4.1.4 Ensemble Prediction

Furthermore, a majority vote for the best three performing models (ResNet50, InceptionV3 and DenseNet121) was applied. For this procedure, the class predictions for the validation set from every model were compared. In a case, whereby two models agreed, and the third model disagreed with respect to the predictions, the vote was given to the majority prediction. If no majority was present the vote was given to the prediction of DenseNet121 since this model performed the best. This procedure ultimately yielded an increase of 4% for the validation accuracy (91%).

4.1.5 Misclassified examples

This section addresses potential error sources in the form of misclassified examples (Figure 19) from the validation set. Hereby, the misclassifications from DenseNet121 model were considered since it was the best performing model amongst all evaluated models.

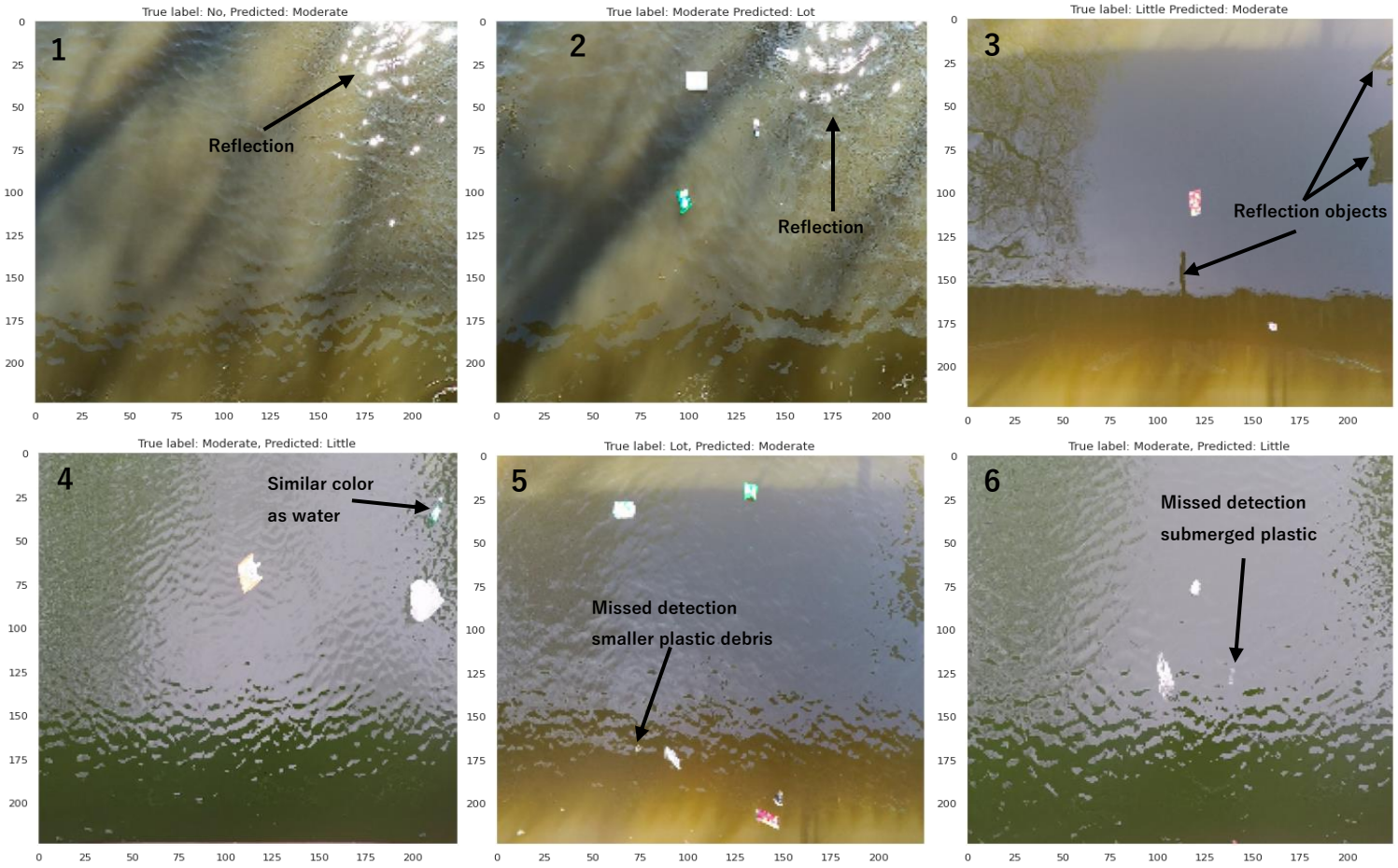


Figure 19 Misclassified examples for the dataset Validation 1 with its true and predicted label. The examples show that reflection due to sun glint, similar color of plastic debris as water surface, smaller objects and submerged objects are potential error sources.

The misclassifications by the model are predominantly due to the characteristics of the natural environment and the plastic debris itself. The most apparent source of error is caused by the presence of sun glint (images (1) and (2) from Figure 19). Due to reflection, the reflective regions in the water can be mistaken for plastic debris by the model during sunny days. The segregation of these objects from its surroundings becomes less dominant through the effect of sun glint. Furthermore, the reflection of objects could also be a source of error (image (3) Figure 19). If non-windy conditions prevail there is barely any distortion of the water surface, therefore making the water surface act similar to a mirror of the surrounding area. The reflection of the bridge, the wooden plank of the camera, trees and the building at the Green Village can be seen on the water surface. The model predicted a moderate amount of litter, although the predicted amount was little litter (1-2 objects). No other plastic objects are visible in the image and therefore it could be assumed that the model falsely detects object reflections as plastic debris. Another important factor seems to be the characteristics of the plastic debris itself (images (3), (4) and (5) from

Figure 19). The model seems to have difficulties in predicting the correct class for objects similar to the colour of the water, smaller objects, transparent or submerged objects.

The potential error sources are merely an assumption made by inspection of misclassified examples. For this reason, it was important to get an insight about how the model extracts information from images once they are passed through convolutional blocks. Therefore, a small network was built whereby an image was passed through three convolutional layers followed by maximum pooling. The suggested procedure by [59] was followed. The kernel sizes were set to 4, 8 and 16. The output were the feature maps of every convolutional and maximum pooling layer. To showcase the operations within a CNN to detect potential error sources, images (2) – (6) from Figure 19 were selected. The feature maps for images (3) – (6) from Figure 19 can be found in Appendix D.3, whereas the feature map for image (2) is shown in Figure 20.

The results show that the activations under the influence of sun glint are in the same range for reflective regions on the water surface and plastic debris. This phenomenon is showcased in Figure 20 whereby the red encircled areas demonstrate that the model wrongfully identifies plastic debris in the reflective regions. The reflection of other objects also seems to have a negative impact on the detection ability of the model, since the reflection of the wooden plank seems to have a similar activation as the plastic debris objects.

Furthermore, an occurrence that was difficult to detect in the misclassified images is the influence of the wind in form ripples on the water surface. For image (5) the small plastic debris object and spaces in between the ripples seem to have similar activations, which is also caused by the reflection of light. Since the object and the distance between ripples is relatively small, the model cannot make a clear distinction and therefore loses its ability to identify the smaller object. The activations for the submerged object and the object which has a similar colour as the water surface are not as pronounced as the other floating plastics. This could potentially explain a missed detection since the activations of these areas get smoothened out with increasing network depth.

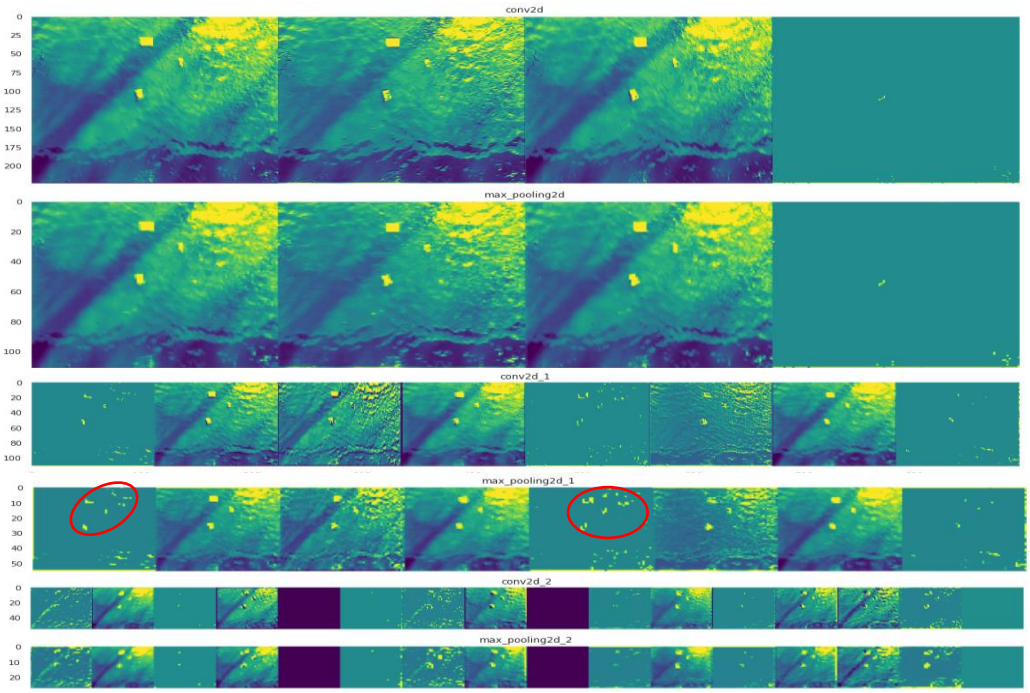


Figure 20 Visualization of feature maps with a convolutional and maximum pooling layer with increasing kernel size (4, 8 and 16) for an image with predicted class 3 and true class 2 (image (2) in Figure 20)

4.1.6 Impact of instrumental and environmental factors

The last step with respect to image classification was the assessment of relevant parameters for the detection of plastic debris in an image. Since the DenseNet121 model achieved the highest validation accuracy on the *Validation 1* dataset, this model was used for the assessment of the parameters. Therefore, firstly the model weights for *Train 1* and *Validation 1* were used to evaluate the influence of the height and angle of the camera. To validate whether the model performance could be improved with expansion of the dataset, the DenseNet 121 model was re-trained on *Train 3*. For the evaluation of weather conditions, the *Validation 1* was split into two test datasets, namely *Test – Sunny* and *Test - Cloudy* representing sunny and cloudy conditions, respectively. The confusion matrices and F1-scores for training on *Train 1* and *Train 3* can be found in Appendix D.4 and D.5, respectively. Overall, following accuracies were achieved (Table 10):

MODEL WEIGHTS (TRAIN 1 (4005 images): 2.7m/0 deg)		
Dataset	Characteristics	Test accuracy
Validation 1	2.7m/0 deg	**88%**
Test 1	2.7m/45 deg	**84%**
Test 2	4.0m/0 deg	**85%**
Test 3	4.0m/45 deg	**54%**
RE-TRAINING (TRAIN 3 (5915 images): All heights + angles)		
Dataset	Characteristics	Test accuracy
Validation 1	2.7m/0 deg	**89%**
Test 1	2.7m/45 deg	**87%**
Test 2	4.0m/0 deg	**88%**
Test 3	4.0m/45 deg	**59%**
RE-TRAINING (TRAIN 1 (4005 images): 2.7m/0 deg)		
Dataset	Characteristics	Test accuracy
Test - Sunny	2.7m/0 deg	**78%**
Test - Cloudy	2.7m/0 deg	**90%**

Table 10 Test accuracies for different datasets to assess the influence of instrumental and environmental factors

The results show that only a minor decrease in test accuracy was found for *Test 1* (84%) and *Test 2* (85%), if these values are compared to the accuracy achieved with *Validation 1* (88%). Compared to the other configurations, a considerable drop in accuracy (54%) could be observed for *Test 3* with the lowest F1-score of 0.32 for Class 3. In 98 cases, the model classified images belonging to class 3, as class 2. Due to the extended field of view caused by the tilt of the GoPro, the size appearance of plastic debris located further upstream seems to reach a threshold, where the model is incapable of detecting the object. This phenomenon is exemplified in Figure 21 A, whereby potentially undetectable objects are highlighted within the red encircled area.

By training on *Train 3* (5915 images) the overall accuracies for all configurations could be increased. Improvements of 1% (*Validation 1*), 3% (*Test 1*), 3% (*Test 2*) and 5% (*Test 3*) were found. It is important to mention, that images taken for *Test 2* had to be cropped, because close to half of the image captured parts of the bridge instead of the water body. By applying this procedure, the test accuracy could be improved from 46% to 85%.

Furthermore, a signifcant difference in test accuracy was found for sunny (78%) and cloudy (90%)

conditions. This confirms the adverse impacts due to sun glint. This assessment also yielded two further potential error sources for the dataset *Test-Cloudy*. The presence of organic material (Figure 21 B) and overlapping objects (Figure 21 C) can negatively influence the prediction ability of the model. Organic material can be seen as additional plastic debris, whereby the overlap of two or more objects can be seen as one object by the model.

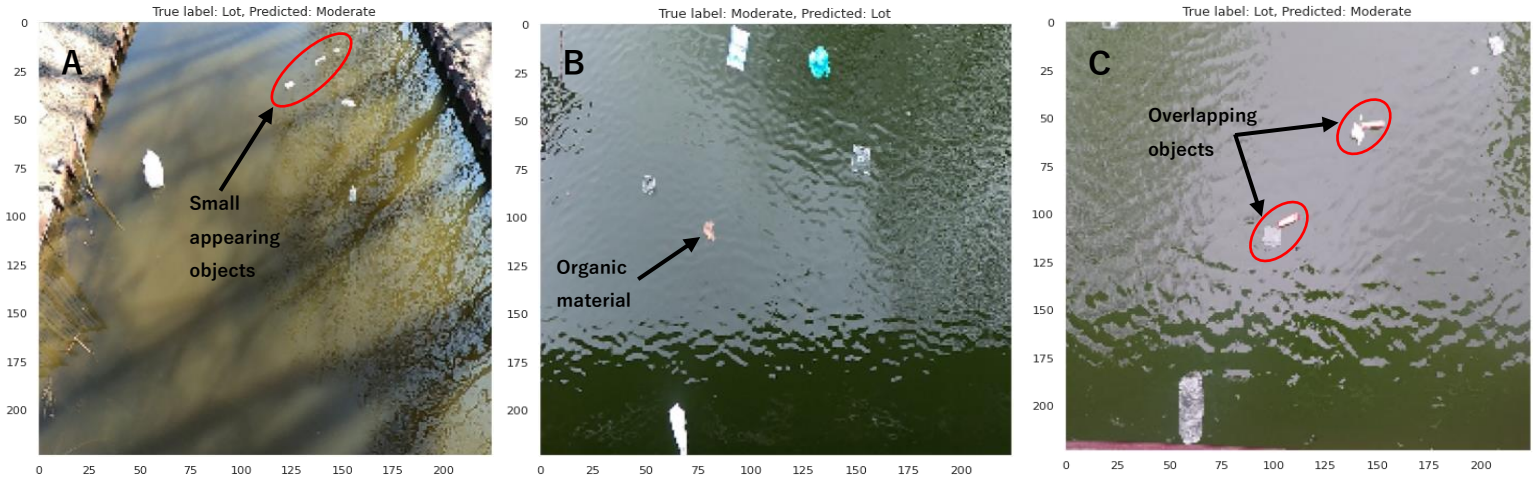


Figure 21 A: Misclassified example for Test 3 exemplifying the problem of detecting objects further upstream due to the change in appearance of size, B: Misclassified example for Test – Cloudy due to presence of organic material, C: Misclassified example due to overlapping objects

4.2 Object detection

Due to the limitations showcased by image classification, it was examined whether object detection could overcome these constraints. The training and test dataset combinations listed in Table 7 (section 3.4.4 'Generalization assessment') were used for the assessment.

4.2.1 Model results Green Village data with one class

The first evaluation focussed merely on the gathered data from Green Village. The results are listed in Table 11.

Dataset combination	TP	FP	FN	Precision	Recall	F1-score	Average IoU	mAP@ IoU=0.5
GEN1	1019	189	57	0.84	0.95	0.89	70.95%	95.61%
GEN2	948	217	37	0.81	0.96	0.88	69.95%	96.63%

Table 11 YOLOv4 model results for GEN1 (Test: GoPro data) and GEN2 (Test: Huawei P30) with true positives (TP), false positives (FP), false negatives (FN), precision, recall, F1-score, average intersection over union (IoU) and the mean average precision (mAP) for an IoU = 0.5

For the first simulation, the YOLOv4 model was trained and tested on the dataset combination *GEN1*. The model results yielded a mAP of 95.61% on the test data, if an IoU of 0.5 was considered. Overall, 1019 of 1076 plastic debris objects were correctly localized. In 57 cases, the model was not capable of detecting the annotated object. A larger number of 189 FPs was found, indicating the misclassification of non-plastic objects as plastic. For this reason, the precision score is considerably lower (0.84), than the recall (0.95). Overall, a F1-score of 0.89 was found. Lastly, the average IoU, thus the overlap of the detected and ground truth box, was 70.95%.

The model results for *GEN2* show, that only minimal deviations exist with respect to the statistical metrics, if the results are compared to the results from *GEN1*. A higher quantity of FPs suggest that the model had more difficulties detecting plastic debris for *GEN2*, hence resulting in a lower precision (0.81). The highest achieved mAP (96.63%) is 1.02% higher compared to the mAP for training and testing on GoPro data.

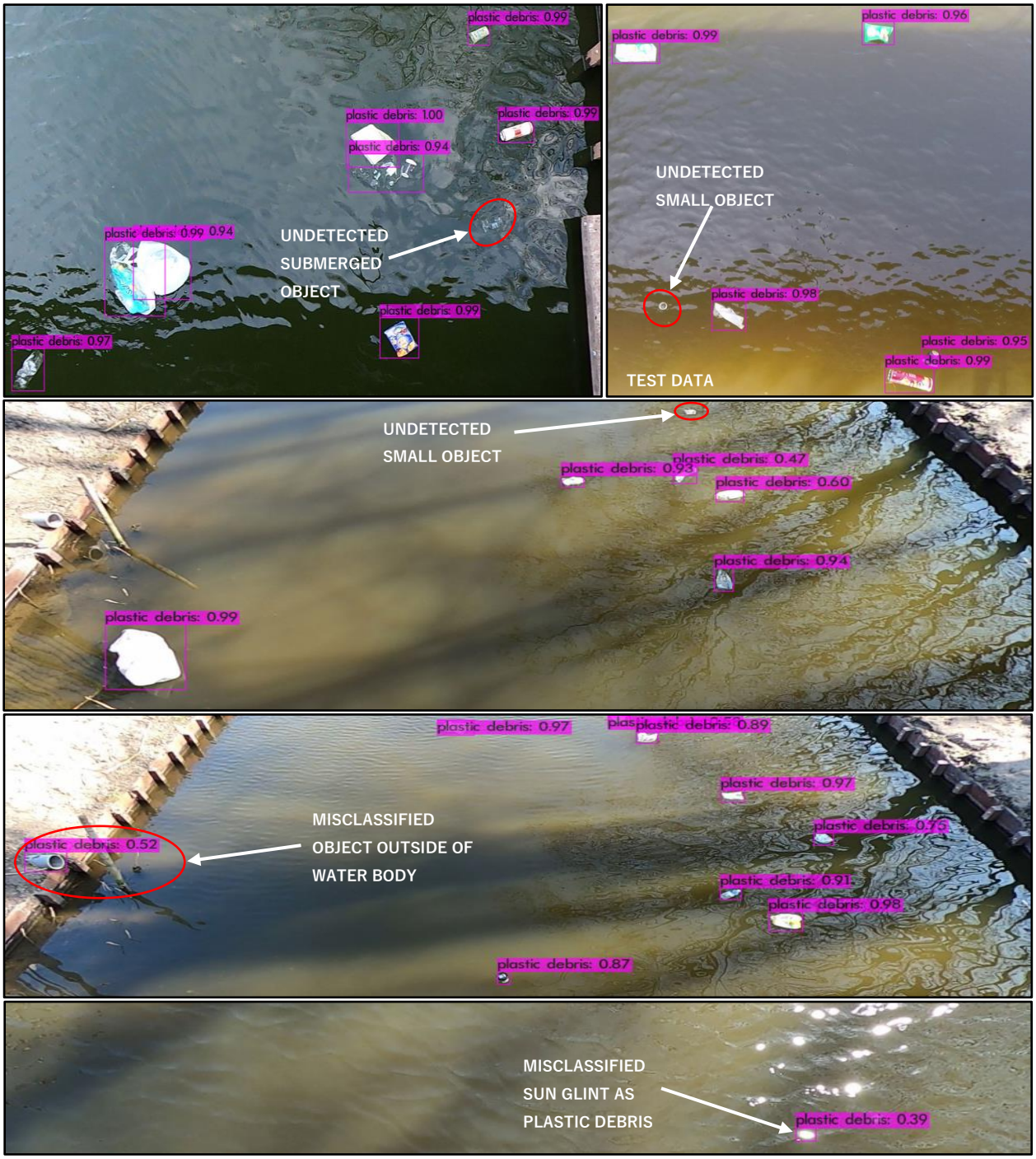


Figure 22 Examples of misclassified and undetected plastic debris objects based on the YOLOv4 model weights from GEN1. Each bounding box is linked to a confidence score that represents the certainty of a detected object being plastic debris calculated by the model.

To evaluate the ability of the model to accurately plastic debris in images, the model weights attained for the highest mAP accuracy were applied to images from the test data to get a visual representation of the predicted bounding boxes. In Figure 22 examples can be seen, whereby at least one plastic debris object was misclassified or undetected. Appendix E.6 represents cases whereby all present plastic debris object in an image were correctly classified. Images were chosen, whereby the DenseNet121 model deemed to have difficulties predicting the correct class due to a higher level of complexity in certain images. This refers to the following conditions: presence of organic material, transparent, submerged and small plastic debris, overlapping and occluding plastic debris, sun glint and reflection of other objects on the water surface.

The cases for which all plastic debris objects could be identified in an image clearly show, that the YOLOv4 model can make a distinction between floating organic material and plastic debris. Additionally, the model can make a clear distinction between occluded and overlapping plastic debris. The detection of submerged and transparent objects is highly dependent on the degree of submergence or transparency of the object. Corresponding images in Appendix E.6 show, that the model is experiencing difficulties detecting these objects with these characteristics, as the calculated confidence score is 0.72 for a transparent object and 0.56 for a submerged object. A missed detection for a transparent and submerged object (Figure 22) shows, that an increased level of transparency or submergence make it impossible for the model to identify the relevant object.

Furthermore, smaller objects or smaller appearing objects due to the field of view, are problematic for detection, as also experienced for image classification. An additional source of error is the surrounding environment, in form of the banks at Green Village. The algorithm detects a protruding PVC pipe directed towards the water body with a confidence score of 0.52 (Figure 22).

Sun glint still remains a potential source of error (Figure 22) for object detection. It is important to note, that the confidence score of the misdetected plastic debris due to sun glint is equal to 0.39. By setting the confidence threshold score to 0.5 as in the model iterations, the misdetection would have been omitted from the predictions.

4.2.2 Model results Green Village data with four classes

SCORES PER CLASS					
Class		**Description**	**TP**	**FP**	**Average precision (AP)**
0		Plastic bottles	115	55	79.99%
1		Other plastic	679	197	87.51%
2		Paper	28	33	35.63%
3		Metal tins	26	63	61.02%
OVERALL SCORES AMONGST ALL CLASSES					
TP	848	**Precision**	0.71	**Average IoU**	62.02%
FP	348	**Recall**	0.77	**mAP@IoU=0.5**	66.04%
FN	256	**F1-score**	0.74		

Table 12 YOLOv4 model results for GEN3 listing the number of true positives (TP), false positives (FP) and average precision (AP) per class. The overall scores include the TP, FP, FN, precision, recall, F1-score, average intersection over union (IoU) and the mean average precision (mAP)

Furthermore, it was also evaluated whether the model can make a distinction between different kinds of floating debris (e.g., plastic bottles, other plastic, paper, metal tins) by using the dataset configuration *GEN3*.The model results per class and overall scores are presented in Table 12. The AP for plastic bottles (79.99%) and other plastic (87.51%) were reasonably high. However, substantial declines in AP were found for metal tins (61.02%) and paper (35.63%). The overall mAP, by taking every class into consideration, was 66.04%. Overall, approximately a 30% decline in AP compared to the single class object detection for plastic debris was found. The lower average IoU for four classes (62.02%) as compared to one class (70.95%) also suggests, that the model experiences more difficulties detecting the ground truth bounding boxes.

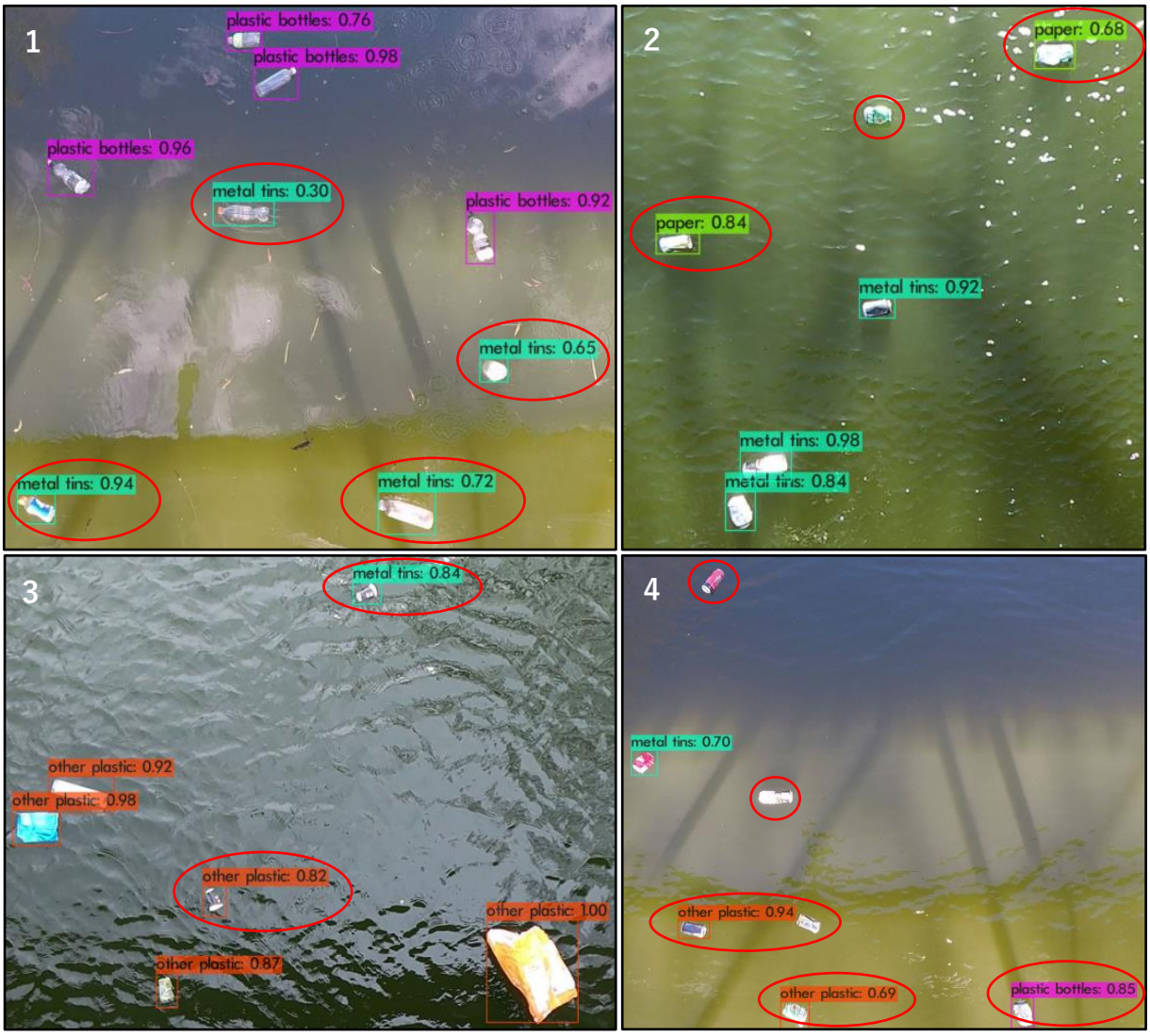


Figure 23 Examples of misclassifications from test data indicated by the red encircled areas. 1: Image with only plastic bottles (four FPs for metal tins), 2: Image with only metal tins (two FPs for paper + one FN for metal tins), 3: Image with all classes (one FP for metal tins and other plastic each), 4: Image with only metal tins (three FNs for metal tins, two FPs for other plastic and one FP for plastic bottles)

The misclassified examples for the test data (Figure 23) show, that some randomness exists in the prediction of the bounding boxes. Overall, 27 plastic debris objects from four different classes are present in the four images. From all these objects, 13 items were correctly and 12 items were wrongly classified and four items remained undetected. This also explains the lower precision (0.64) compared to the recall (0.73), due to a higher number of FPs. It seems that the model is not able to make a clear distinction between the different objects.

4.2.3 Generalization ability to another location and setting

Lastly, the generalization ability for other locations was evaluated. Since no data from the natural environment was available for the Netherlands a dataset from [10] was used, as previously explained. The results are presented below (Table 13):

Dataset combination	TP	FP	FN	Precision	Recall	F1-score	Average IoU	mAP@ IoU=0.5
GEN4	67	11	1141	0.86	0.06	0.10	59.37%	12.23%
GEN5	797	1089	279	0.42	0.74	0.54	31.82%	63.51%

Table 13 YOLOv4 model results for GEN4 and GEN5 training listing the number of true positives (TP), false positives (FP), false negatives (FN), precision, recall, F1-score, average intersection over union (IoU) and the mean average precision (mAP) for an IoU = 0.5

The model results for *GEN4* show that precision is reasonably high (0.86), whereas the recall is extremely low (0.06). The average IoU for the detection of the relevant objects is equal to 59.37%, also indicating a satisfactory performance once the model can predict the ground truth label. Nevertheless, if the 1208 annotations from the A-RPM test dataset are considered, the model is not able to detect these annotations in 94.5% (1141 FNs) of the cases. Overall, this results in a F1-score of 0.10 and a mAP of 12.23%.

In Figure 24 (A and B) two examples are presented, which showcase the large discrepancy with respect to the model predictions and the ground truth labels. The model can predict larger and clearly visible plastic debris. However, once the size of these objects reaches a certain threshold, the model is not able to detect these objects. This also holds for transparent objects or objects that are further submerged in the water.

By using *GEN5* an opposite behaviour could be observed. The model wrongfully detected regions in the water as plastic debris, although these areas were not annotated as such (1089 FPs), resulting in a low precision (0.42). For 27.8% (279 FNs) of the annotations, the model is not able to detect them, resulting in a recall of 0.74. Overall, the average IoU is significantly lower (31.82%), whereas the F1-score (0.54) and mAP (63.5%) are substantially higher.

Lastly, the example in Figure 24 C shows, that the model mistakes patches of sun glint for plastic debris. These patches are much smaller compared to objects used for the experiments in the Green Village. This also indicates that the labelling of small plastic debris in the A-RPM training dataset leads to difficulties for the model in the form of high FP rates. It could be concluded that there is a large discrepancy in object sizes between the datasets from the Green Village and A-RPM. This led to either high FP or FN rates.

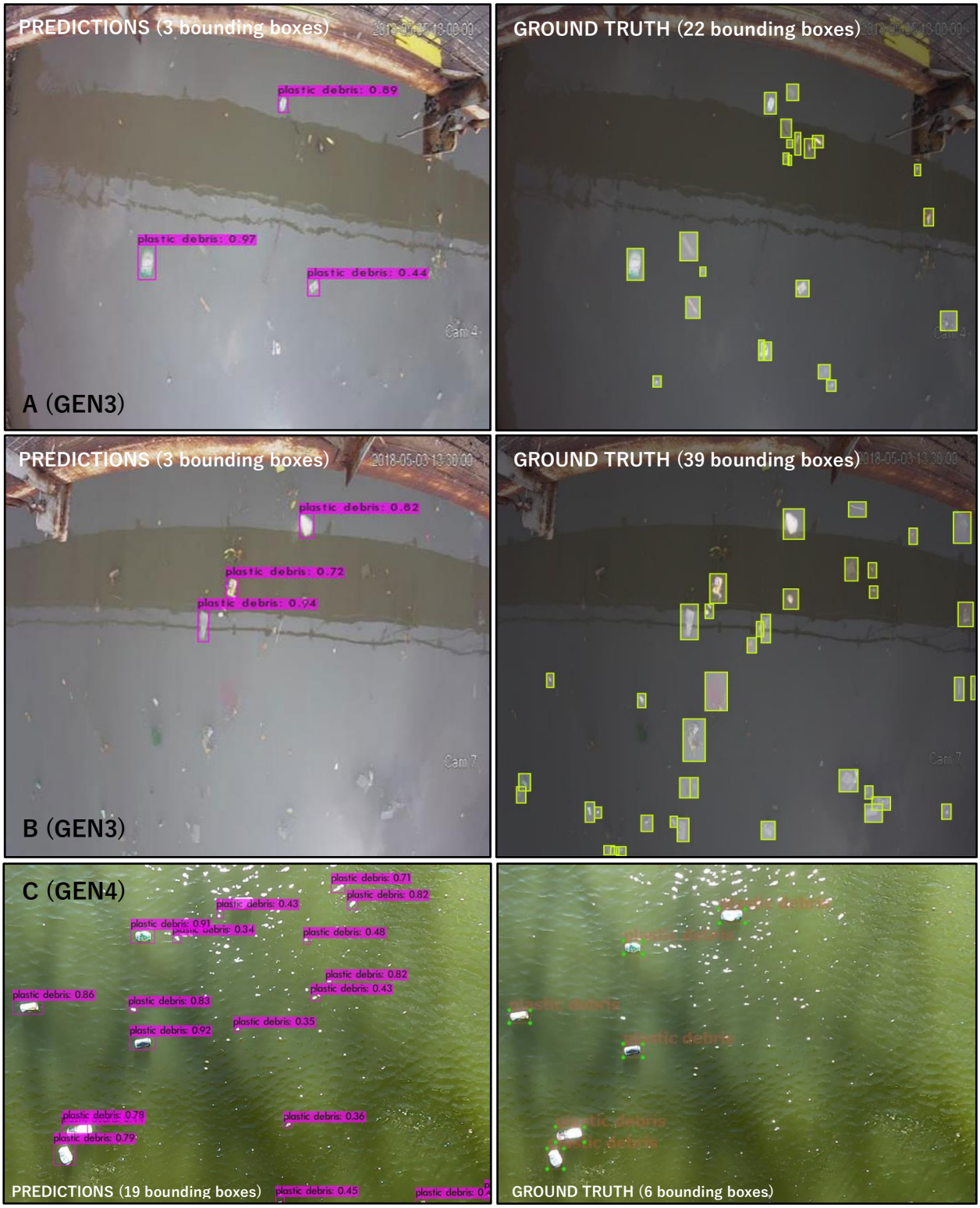


Figure 24 Corresponding images with predictions from YOLOv4 and ground truth labels. A: Model correctly predicts three objects but 19 objects remained undetected (GEN3), B: Model predicts three correct objects, but 36 objects were left undetected (GEN3), C: Model correctly predicts five objects, one object was left undetected and 13 objects were wrongly classified as plastic debris

5 Discussion

The discussion will be segregated into three different sections, namely: image classification, object detection and implication of results. The implication of results will place the work of this study into perspective.

5.1 Image classification

For the construction of the baseline model the loss, degree of overfitting and highest validation accuracy were used as guidelines for selecting the best configuration. This was also applied to evaluate the five selected state-of-the-art models. The results showed that large fluctuations (validation losses and accuracies) and overfitting were present until 30 epochs were reached.

Firstly, it needs to be mentioned that models with as many parameters as the ones at hand need to be fed with significantly more data, which could be an explanation for this behaviour. Furthermore, due to the present background noise in the images (e.g., differences in lighting or wind conditions) it could be the case that the algorithm potentially learns the pattern of the noise, rather than the actual sign in the form of plastic debris objects. This is also supported by a study of [60] who found that the neural network often has difficulties converging if the signal to noise ratio is low. A low signal to noise ratio is present if the background occupies more than 50% of the image [60]. This phenomenon was also showcased for cropping of data from 4.0m/0 deg, whereby the test accuracy could be drastically improved from 46% to 85%. This suggested the reduction of background noise leads to an increase of the model accuracy. Additionally, the variations can also be linked to potential difficulties of the model to correctly detect certain images in the validation set due to the detected error sources and differences in the training and test data.

Secondly, apart from the quantity of data and complexities in the images, the stochastic nature of the learning algorithm could also lead to these fluctuations [61]. For a deterministic algorithm, the same output is produced if the initial state or conditions remain unaltered, not introducing any form of randomness. However, the Adam optimizer is a stochastic algorithm, which possesses intrinsic randomness for the generation of parameter values, implying that the same set of parameters and initial conditions can lead to different outputs [61]. Therefore, the results could vary once the same model was run consecutive times.

Figure 25 Example of augmented image from validation set with noise injection and increased brightness (encircled areas highlight plastic debris)

Since regularization techniques such as Batch Normalization and Dropout could only limit overfitting to a certain extent, data augmentation was applied. Data augmentation led to a decrease in accuracy by approximately 9%. A study by [62] proves, that data augmentation does not provide the CNN with invariance, because the network learns to be invariant to transformations of images that are comparable to images in the training dataset. It was shown that merely one downward translation of a single pixel can

lead to a decrease in accuracy up to 30%. The distribution of images in the training dataset is highly biased, which can result in shortcomings with respect to invariance that do not follow this bias [62]. This bias could have also been present for this study, since the training and validation set were inherently different with respect to weather conditions and characteristics of the plastic debris. Therefore, it seems that the best solution to evaluate whether the degree of overfitting can be limited by gathering additional training data.

Another possible explanation for the adverse effects of data augmentation could be the chosen transformation techniques. An augmented image example with increased brightness and noise injection is presented in Figure 25. It can be seen, that through the introduction of these techniques, the plastic debris does not clearly segregate itself anymore from the surrounding water and consequently the objects become less recognizable. Furthermore, increased brightness potentially emulates similar effects as caused by sunny conditions, leading to a decrease in the validation accuracy. The random injection of noise possibly also makes the detection of smaller plastic debris object more challenging. Although the validation accuracy decreases with data augmentation, it can be argued, that the negative impact might have been caused by the choice of techniques. This however could not be assessed, since the transformation techniques were combined in images, rather than applying every method separately. A study by [20] found that the detection of marine plastic debris increased from 83.1% to 87.9% when the number of images increases from 1000 to 4000 by the generation of new images through data augmentation. Therefore, techniques such as zooming, shifting, flipping and rotation were used separately per image. This emphasizes the fact that the separate implementation of data augmentation techniques has the potential to boost the validation accuracy.

With respect to transfer learning, a study by [63] suggests that poor model performances with pre-trained weights, could be explained by a mismatch in the domain between the images the model was originally trained on and for what it was utilized. The distribution of images and classes from the ImageNet dataset are fundamentally different compared to the images and classes which were used in this study. Deeper layers in CNNs can recognize features belonging to the images the model was trained on. The more specific and complex features detected for the ImageNet dataset are seemingly not applicable for the detection of plastic debris. Therefore, re-training the model eliminates the problem of domain transferability, since the model is re-trained on one custom dataset, rather than using pre-trained weights from another dataset. Since training with pre-trained weights yielded unsatisfactory model results it can be argued, that training the last convolutional layers by fine-tuning could have overcome the problem of domain transferability by also decreasing the computational time compared to training from scratch.

The results with respect to pre-trained weights are contradicting compared to a study by [20]. In this research the VGG16 model pre-trained on the ImageNet dataset achieved an accuracy of 86% on the detection of marine plastic debris in the form bottles, buckets, or straws. However, the dataset required cropped images with a central alignment of the object. This could indicate that the mitigation of background noise by cropping elevates model performances. Another possible explanation could be the fact, that the study by [20] used 9600 images, whereas only 4005 images were used in this study for the assessment of pre-trained weights.

The better performance of larger models (Resnet50, Inception V3 and DenseNet121) can be explained by the fact that larger models can represent more complicated functions than smaller models with fewer parameters. The use of a small network can lead to problems with respect to identifying more complex features, explaining the lower accuracies for MobileNetV2 (75.03%) and SqueezeNet (73.91%).

Furthermore, it is noteworthy that the DenseNet121 model outperformed much deeper networks (ResNet50 and InceptionV3) with approximately a third of the total parameters. The main problem with ResNet50 is the fact, that the identity shortcut skips the residual blocks for the preservation of features, leading to limited representation ability of the model [64] [65]. DenseNet121 has an inherently different approach, as it does not depend on the representation power of an extremely deep network, since it reuses the learned features. This is done by dense concatenation of feature maps from previous layers that are used as inputs for subsequent layers [65] [66]. InceptionV3 also makes use of concatenation operations, however DenseNets are simpler and more efficient [66]. Nevertheless, the highest computation time was found for DenseNet121, which is caused by the more intense use of GPU due to concatenation operations and the use of small convolutions [65].

Additionally, it became evident, that image classification has its flaws with respect to the detection of plastic debris. Studies by [21] [67] [68] suggest that the training of neural networks with imagery including reflectance in the form of shadows or sun glint can be considered as problematic. This research supports these findings since the reflectance of objects in the form of shadows the presence of sun glint seemed to hamper the model performance. However more case studies will be needed to confirm these findings. It is suggested by [21] that data collection should take place during optimal meteorological conditions should take place whereby the reflection of objects and sun glint can be avoided.

Furthermore, since the entire image is used as an input, it became difficult to comprehend what the model perceives. The misclassified examples and the simplified model for the generation of feature maps gave an indication about the perception of the model. Nevertheless, it cannot be fully proven that the made observations and assumptions correspond entirely with reality. Therefore, more studies will be needed to confirm the adverse impact of the found error sources. Nevertheless, for this research, the use of object detection could also be justified since only the relevant objects were annotated, omitting most of the irrelevant background noise.

Lastly, with respect to the datasets it needs to be mentioned that only a limited number of plastic objects were used. In reality every floating plastic debris object is unique with respect to its characteristics. Furthermore, the dataset was artificially generated in a controlled environment, therefore the high validation accuracies might be inflated. To really evaluate the generalization ability of models with the given dataset from the Green Village, data also needs to be gathered from the natural environment. Only afterwards it can be assessed whether the use of image classification can be a meaningful tool for the detection of floating plastic debris.

5.2 Object detection

The results for the detection of floating plastic debris by means of object detection with YOLOv4 showed, that high test accuracies could be achieved for the Green Village data, if only one class was considered. By the introduction of four classes, the complexity was increased, which also resulted in a significant decrease in the test accuracy. This can be explained by the fact, that the object shapes amongst the classes were almost identical. Items might also not appear in their original shape due to deformation, leading to more impediments for the model to predict the correct class. This is also suggested in a study by [21] who found resemblance in shape and form for different pollution classes. These misclassifications included a mix up between plastic cups and plastic bottles whilst plastic bottles and Styrofoam were identified as polystyrene [21].

Furthermore, a study by [22] found a mAP of 81.5% for the detection of different pollutants in a marine environment (e.g., glass, paper, cardboard, plastic, metal, trash). This value is significantly higher than the mAP of 66.04% found for this study. It was a difficult task to gather images realistically with respect to representativeness, as they were gathered in a controlled environment. In view of the database in existing studies [20] [22], this also seems to have been challenging.

Another possible explanation for the large discrepancy in accuracy amongst all classes, is an imbalance in the annotations per class for the training and test dataset. Class 2 ('Paper') is the least represented class in the training data (11.9%), suggesting that the model does not have sufficient training examples to predict this class accurately. The recommended number of images according to [40] for each class should be equal to 2000. Overall, only 1327 training images and 183 testing images were used. This suggests that four times more training examples would have been needed for four classes as opposed to one class. However, it could be seen, that classes with the highest number of annotations, had the highest test accuracies, except for metal tins. The model seemingly has more difficulties predicting metal tins than plastic bottles, although less training examples were available for plastic bottles. This can potentially be explained by the scarcity of annotations metal tins in the test dataset.

Furthermore, the labelling procedure should be kept as simple as possible by classifying each plastic debris object as one class. With four classes, the model not only requires more training examples, but the labelling procedure also becomes more complex. In some scenarios, making a clear distinction between different plastic debris is an arduous task. Furthermore, it needs to be guaranteed that the labelling is consistent. With the introduction of more classes the probability increases of making erroneous annotations. The less complexity is introduced in the labelling procedure, the more consistency can be reached. Studies by [1] and [21] aimed to classify different kinds of plastic litter, using semantic segmentation and image classification, respectively. It is also suggested by [1], that the visual labelling of plastic debris is a tedious task involving a lot of errors. Therefore, the question arises whether it is necessary to make a distinction between different objects in the water, since this further adds complexity to the labelling procedure which can negatively influence the model performance.

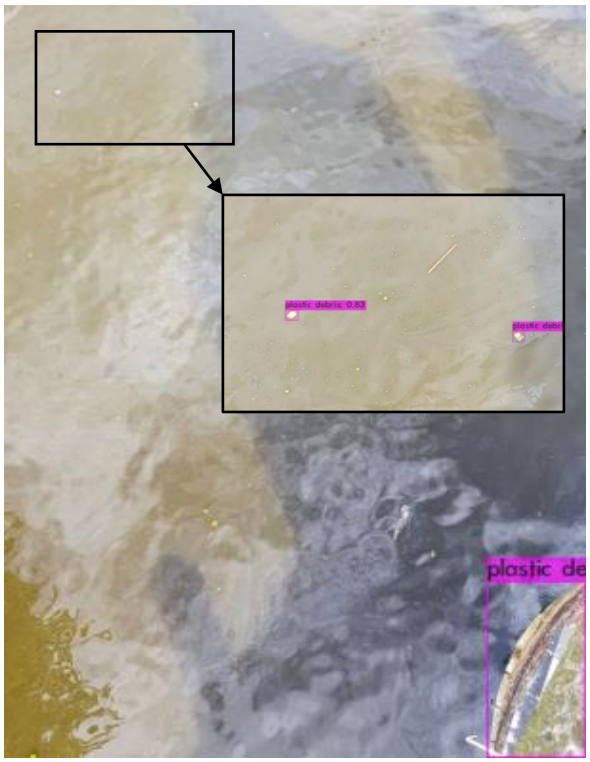


Figure 26 Example of an image taken by a mobile phone whereby originally only parts of a bridge are detected as plastic debris, but once zooming was applied, actual plastic debris could be detected

It also became evident, that the detection of floating plastic debris from the A-RPM dataset has a much higher complexity. Often the objects are so small, that it even becomes difficult for the human eye, to make a clear distinction between plastic debris and other objects such as organic material. The poor model performance for *GEN1* can be explained by the different settings, characteristics and the size of plastic debris compared to the A-RPM dataset. A glaring flaw of the Green Village dataset is that

only images from one height and angle (2.7m and 0 deg) were annotated. This made the model accuracy susceptible to smaller or smaller appearing objects. For this phenomenon two examples can be given. Firstly, one key disadvantage by labelling small objects could be observed with the presence of sun glint, whereby a large quantity of small sun glint patches was detected as plastic debris. Secondly, an example is given in Figure 26 whereby the model detected parts of the bridge. This could be remedied by specifically zooming to regions with plastic debris.

One of the two comparable studies for the detection of plastic debris with object detection in rivers is the study by [10]. In the research by [10] a baseline accuracy of 68.7% was found, whereas in this study an accuracy of 95.6% was found for *GEN1*. It is suggested by [69] deep learning algorithms tend to perform worse in non-Western countries and households with lower incomes, which would confirm these findings. Nevertheless, this comparison needs to be treated with caution, since the A-RPM dataset was recorded in another country with different settings.

With respect to the settings the A-RPM dataset was recorded from higher altitudes and the images were captured with a fisheye view that produces strong curvature and visual distortion. Fisheye view creates a wider panoramic view, however the curvature could lead to a deformation of the present objects in the water body.

Furthermore, the data in the study by [10] was aggregated for five locations in a natural environment with flowing water. The data from the Green Village was gathered in one location and a controlled environment with a stagnant water body. It is stated by [10] that training on data from one single location is insufficient to gain satisfactory generalization to other locations since large accuracy fluctuations could be found for the detection of plastic debris for five different locations in Jakarta. This indicates that generalization is already difficult for geographically similar locations. This is also suggested in a study by [69] which proves that these algorithms perform poorly on the recognition of these objects around the world, suggesting that generalization to different locations and situations is a difficult task. However, it also needs to be mentioned that the worst accuracy (20.5%) was achieved for the highest altitude (8.0 m) in the study by [10] indicating that a different setting could also have an adverse impact.

This emphasizes the fact that images from different locations in the natural environment in the Netherlands need to be gathered to evaluate the generalization ability within the Netherlands. The inclusion of data from other locations to the original training dataset also improved the generalization ability of the model in the study by [10]. To overcome the adverse impact of all these factors the YOLOv4 model could be trained on a combined dataset of the Green Village and A-RPM data. This may make the model robust towards height differences, smaller plastic debris and object characteristics, resulting in improved generalization.

With respect to pre-trained weights, it was observed that training with pre-trained weights for object detection yielded much better results as compared to image classification. This can potentially be explained by the fact that a lower margin of error is present for object detection as compared to image classification. Whereas image classification takes the entire image into account including the background noise, object detection specifically focusses on the object of interest by omitting unnecessary background noise.

Lastly, for practical applications this research sets a foundation, however additional data and research will be required. Monitoring with RGB cameras is limited to daytime hours. Studies by [70] and [71] investigated the detection of floating plastic litter and patches with multispectral sensors. This is also supported by a study of [21] which states that multispectral sensors could further improve plastic identification in the natural environment.

These sensor types would allow for continuous monitoring throughout day and night.

Lastly, the YOLOv4 model needs to be able to count the number of plastic debris objects passing through a frame for practical monitoring applications. Up to this point, the model is able to detect plastic debris, however for monitoring purposes, a real-time tracker would be needed to quantify the plastic debris.

5.3 Implication of results

Combining image classification and object detection this study provided an innovative approach that has not yet been used in this form in other studies. Existing methods were adapted for this research so that they could be applied with fewer resources. This can be seen as an achievement of this work.

Furthermore, this research effectively demonstrated that deep learning-based computer vision models can detect floating plastic debris in images with a high accuracy. The test accuracies for the baseline model and state-of-the-art models showed that the focus should not be placed on the development of new models, but rather utilizing existing state-of-the-art models. This is also supported by the results for object detection with the YOLOv4 model. This emphasizes the fact, that a paradigm shift should take place from a model-centric to a data-centric approach. There is no scarcity of well performing available models, but qualitatively meaningful data with respect to floating plastic debris is a rare commodity. To attain the required amounts of data for such a data centric approach help from Citizen Science Platforms such as CrowdWater is needed. Clear guidelines should be given by these platforms with respect to the collection of data, such that also deep learning algorithms can make use of the uploaded data.

Additionally, this study generated various datasets that can be used to further explore the detection of plastic debris in waterways. Especially for parties such as Noria and RWS these datasets can be valuable since this study generated the first available datasets that are representative of Dutch conditions. It was shown that the generalization to other countries and even other locations is difficult. Therefore, Dutch based parties that are interested in the detection of plastic debris in waterways with deep learning techniques should focus on the collection of data within the Netherlands. By aggregating the first datasets that are representative of Dutch conditions, the first steppingstone was set to build upon for the development of deep learning-based computer vision techniques in the Netherlands for the detection of plastic debris in waterways.

By investigating and proving potential error sources that influence the model performance, future research can include these aspects for setting up guidelines with respect to data collection. To make other research more comparable, the labelling procedures from this study can be adopted. This study also showed that variability in the dataset is an important aspect since the addition of data from different configurations (2.7m/45 deg. 4.0m/0 deg and 4.0m/45 deg) to the original dataset (2.7m/0 deg) could improve the test accuracies. This emphasizes the importance of variable data for the improvement of the model performance.

Overall, the products from this study yielded important insights with respect to monitoring guidelines for plastic detection in waterways and possibilities of deploying automated systems. Finally, it can be summarised that this study contributes to the research of an underrepresented but highly relevant problem that hits the current nerve of the time. It is desirable that, building on my study, methods and approaches can be further developed and improved in the future in relation to the underlying research interest. The results presented can have added value for solving the increasing plastic occurrence in global waters in practice. This will be addressed in more detail in chapter 7 after drawing conclusions of the key findings of this work in the following.

6 Conclusion

The purpose of this research was to explore how computer vision and deep learning techniques can be applied for the detection of floating plastic debris in Dutch waterways. This study clearly showed that computer vision and deep learning techniques can detect floating plastic debris with a reasonable accuracy. These techniques consequently have the potential to be utilized for various applications with respect to the detection of floating plastic debris in waterways, Below, the main conclusions with respect to the sub questions are listed.

(1) For image classification the model accuracy of a from scratch-built baseline model and state-of-the art models were compared. The final baseline model achieved an accuracy of 79% on the *Validation 1* dataset. Training with pre-trained weights from the ImageNet dataset for five chosen state-of-the-art models resulted in unsatisfactory model performances, with the highest model accuracy achieved by SqueezeNet (66.70%). By training from scratch, model accuracies could be greatly improved, with InceptionV3 (81.5%), ResNet50 (82.4%), DenseNet121 (87.6%) outperforming the baseline model. By applying a majority vote from the three best performing models, the accuracy could be boosted to 91%. With implementation of the object detection model YOLOv4 with pre-trained weights, plastic debris objects were detected with a mAP of 95.61%.

(2) The use of a mobile phone (Huawei P30) yielded a marginally higher accuracy (96.63%) compared to a GoPro (95.61%). It is therefore a promising result that no decrease in the model accuracy was observed if another image source was utilized. Furthermore, it seemed that the model trained on the GV data had large difficulties generalizing to the A-RPM data (12.23%), due to height differences, discrepancy in object sizes and different camera settings, Training on the A-RPM data and testing on the GV data, yielded a model accuracy of 63.51%. It appeared that better model performances can be achieved with the A-RPM data, due to better generalization across varying object sizes.

(3) By applying image classification, it was found that following factors influence the detection of plastic: presence of organic material, wind, transparent, submerged and small plastic debris, overlapping and occluding plastic debris, sun glint and reflection of other objects on the water surface. Sunny conditions deemed to be less favourable (78%) than cloudy conditions (90%), which can be attributed to the presence of sun glint. Furthermore, high model accuracies were found for following datasets: *Validation 1* (88%), *Test 1* (84%) and *Test 2* (85%). Additionally, it was identified that if the field of view is too large (*Test 3*) and the appearance of objects decrease with distance from the camera, that the model accuracy decreases substantially (54%). By the implementation of object detection, the influence of error sources was reduced. The results indicated, that the remaining negatively influencing factors were small, transparent and submerged objects and the presence of sun glint.

(4) For the distinction of four different plastic debris classes the YOLOv4 model performed well on the detection of plastic bottles (79.99%) and other plastic (87.51%). A decrease in AP was experienced for metal tins (61.02%) and paper (35.63%). The large discrepancy in accuracies may be attributed to the increased complexity of detecting different object shapes in water and the large class imbalances.

7 Recommendations for future research

From the results, conclusions and discussion recommendations for future research could be extracted. These will focus on five main aspects, namely: sensor improvements (A), limitation of error sources (B), data collection (C), application for monitoring (D) and application for detection of specific plastic debris object (E).

A. SENSOR IMPROVEMENTS

(1) Prevention and diminishing sun glint

The most straightforward and simple method to limit the adverse effects of sun glint is to avoid capturing images on sunny days. Another solution would be to filter the recorded images based on the presence of sun glint in specific images, since some images might not contain sun glint. However, this might be not the most favourable option since the model should be able to effectively detect floating plastic debris during sunny conditions. Polarizing filters can be used to reduce the influence of sun glint [30] [72]. Nevertheless, attention needs to be paid to potential information loss for rapidly changing weather conditions, caused by a too strong polarising filter. It is recommended, to explore the possibilities of implementing polarising filters.

(2) Sensor types for continuous monitoring

A clear limitation is that monitoring with RGB cameras is only possible during daytime. To evaluate whether continuous monitoring is possible throughout day and night, it is recommended to experiment with multispectral sensors. The images are consequently not limited to the lighting conditions, since multispectral images capture data based on the emitted wavelength of objects.

B. LIMITATION OF ERROR SOURCES

(1) Detection of small objects

To make the YOLOv4 model more accurate with respect to the detection of small objects, it is recommended to annotate images from different heights and angles for object detection. This increases the variability in the dataset, such that the model will also be able to detect smaller and smaller appearing objects in the future.

(2) Influence of wind

It is recommended, to further explore the influence of wind on the detection ability of the model. Image classification showed that through the influence of wind, the activations of smaller objects get lost in the activations of the ripples. However, this was done by means of a qualitative analysis rather than a quantitative analysis.

(3) Use of two stage-detector for object detection

Due to GPU constraints for this study and the ambition to implement the model in real-time, it was chosen to use the YOLOv4 model. Nevertheless, a two-stage detector for object detection (e.g., R-CNN or Faster R-CNN) could further improve the model results. Therefore, it is recommended, once more computation resources are available, to experiment with two-stage detector algorithms, as also used in the study by [10].

C. DATA COLLECTION

(1) Data natural environment

For generalization to other locations, it is recommended to gather data from the natural environment. Therefore, it is firstly important to gather data on small scale. This would refer to mounting a camera to a narrow water body with approximately the same width as the water body in the Green Village. Beforehand, locations should be selected, that experience continuous flow of plastic debris. To also assess whether there is a large decrease in validation accuracy, cameras should be mounted at heights exceeding 4.0m. It is recommended, to mount the camera horizontally or with a slight angle to the water surface and recording with a linear FOV.

If the model is able to accurately predict plastic debris in these locations, the size of the monitoring locations can be scaled up. This refers to broader canals, rivers or waterways, where multiple cameras can be mounted next to each other. Hereby it is important, that there is no visual overlap between the cameras, to reduce the probability of plastic debris objects being detected multiple times. Problems for continuously mounting a camera are available infrastructure, power supply and storage space for videos and images on the camera. The camera needs to be fixed at an easily accessible location that also ensures continuous power supply. To decrease the storage space demands, images can be taken at specified time interval instead of taking videos. A more sophisticated approach would be to connect the device(s) to a cloud system where the images or videos can be uploaded automatically.

(2) Data from Citizen Science

Citizen Science has the potential to be a significant contributor for valuable data that could be used for deep learning algorithms. However, platforms such as CrowdWater need to establish specific guidelines with respect to data collection. This study showed with the analysis of influential parameters, which factors are important for the image quality and the associated model performance. The decision process for people uploading their images to this platform should be kept as simple as possible. It is further recommended, to look into the implementation of an image classification step, before images are uploaded. It could be utilized as tool to sort images based on the presence of plastic debris in waterways.

(3) Data augmentation

Another improvement point for the future is the implementation of data augmentation. It is recommended to evaluate the influence of different data augmentation techniques separately. This should not be limited to techniques used in this study, but the scope should be extended to other techniques. This specifically refers to image classification, since data augmentation is implemented in the YOLOv4 model.

(4) Extension of image sources

For the future it is also recommended to extend the image sources (e.g., drones). This would give the opportunity to assess whether floating plastic debris can be detected from the air with deep learning algorithms. This would further allow Noria to carry out aerial surveys on waterways for the detection of floating plastic debris. Furthermore, research by [70] and [71] showed, that it is possible to detect floating plastic litter and patches with Sentinel-2 Imagery. It is therefore thus recommended to evaluate, whether satellite imagery can be used for the detection of floating plastic debris in waterways.

(5) Labelling procedure

For the future it is recommended that clear labelling guidelines are established, once more monitoring locations and data are available For this the guidelines from this study can be used. It is also important to ensure consistency in case more people are involved in the labelling of data. To measure consistency, independent labellers could label a sample of images, which could then be compared.

D. APPLICATION FOR MONITORING

(1) Object tracking

This study recommends adding an object tracking module to the YOLOv4 model, such that the detected objects can effectively be tracked and counted. To assess the accuracy of the model with respect to the quantification of objects, the counted number of objects by the model should be compared to human visual counting. This should be done for several videos from different locations and different conditions to ensure variability. Object tracking also offers the opportunity to associate the number of counted objects to weight quantification of floating plastic debris.

(2) Real-time detection

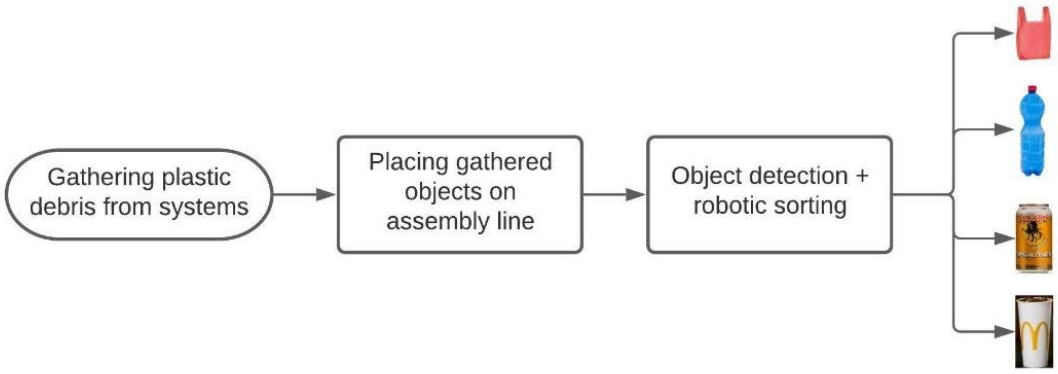
Once enough data has been gathered and labelled from the natural environment and the test accuracies are satisfactory, the feasibility of real-time detection can be evaluated. A similar approach can be taken as for the data collection in the natural environment. Once real-time detection works on small scale with one camera, the size of the monitoring location can be scaled up.

(3) Sensitivity to submerged plastic debris

This study clearly showed that the detection of plastic debris with RGB cameras is limited to the water surface. Plastic debris transported below the water surface are therefore not considered. A study by [73] showed, that echosounders can detect suspended plastics. For weight quantifications across the entire water column, a combination of echosounders and cameras could be considered. If merely the surface share is considered, echosounders could be also used to validate the quantity of detected floating plastic debris by the YOLOv4 model.

E. APPLICATION FOR DETECTION OF SPECIFIC PLASTIC DEBRIS OBJECTS

(1) Assembly line



Since Noria also has the ambition to make a distinction between the different kinds of plastic debris in waterways, another approach is recommended. Due to the complexity of making a separation between these objects in the water, as well as for a human (e.g., labelling) and the model, it is recommended to apply object detection once the objects were removed from the water. This can be done by means of an assembly line, whereby depending on the detected object, a robotic sorting system directs the item to an object specific container. An advantage of this approach is, that all the background noise from the natural environment gets eliminated. Furthermore, the objects can be captured with high resolution from a close distance.

8 Bibliography

[1] G. Jakovljevic, M. Govedarica and F. Alvarez-Taboada, "A Deep Learning Model for Automatic Plastic Mapping Using Unmanned Aerial Vehicle (UAV) Data," *Remote Sensing,* vol. 12, no. 9, p. 1515, 2020.

[2] R. Tramoy, J. Gasperi, R. Dris, L. Colasse, C. Fisson, S. Sananes, V. Rocher and B. Tassin, "Assessment of the Plastic Inputs From the Seine Basin to the Sea Using Statistical and Field Approaches," *Frontiers in Marine Science,* vol. 10, 2019.

[3] UN, "The State of Plastics - World Environment Day Outlook 2018," 2018. [Online]. Available: https://wedocs.unep.org/bitstream/handle/20.500.11822/25513/state_plastics_WED.pdf?isAllowed=y&sequence=1. [Accessed 2020 November 24].

[4] J. Jambeck, R. Geyer, C. Wilcox, T. Siegler, M. Perryman, A. Andrady and R. Narayan, "Plastic waste inputs from land into the ocean," *Science,* vol. 347, no. 6223, pp. 768-771, 2015.

[5] T. Van Emmerik and A. Schwarz, "Plastic debris in rivers," *Wiley Interdisciplinary Reviews: Water,* 2019.

[6] L. Meijer, T. Van Emmerik, R. Van der Ent, C. Schmidt and L. Lebreton, "More than 1000 rivers account for 80% of global riverine plastic emissions into the ocean," *Science Advances,* vol. 7, no. 18, 2021.

[7] S. Hohn, E. Acevedo-Trejos, J. Abrams, J. de Moura, R. Spranz and A. Merico, "The long-term legacy of plastic mass production," *Science of the Total Environment,* vol. 746, 2020.

[8] L. Lebreton, J. van der Zwet, J. Damsteeg, B. Slat, A. Andrady and J. Reisser, "River plastic emissions to the world's oceans," *Nature Communications,* vol. 8, no. 1, 2017.

[9] T. Van Emmerik, T. Kieu-Le, M. Loozen, K. Van Oeveren, E. Strady, X. Bui, M. Egger, J. Gasperi, L. Lebreton, P. Nguyen, A. Schwarz, B. Slat and B. Tassin, "A Methodology to Characterize Riverine Macroplastic Emission Into the Ocean," *Marine Pollution,* 17 October 2018.

[10] C. Van Lieshout, K. Van Oeveren, T. Van Emmerik and E. Postma, "Automated River Plastic Monitoring Using Deep Learning and Cameras," *Earth and Space Science,* 28 July 2020.

[11] D. González-Fernández and G. Hanke, "Toward a Harmonized Approach for Monitoring of Rivering Floating Macro Litter Inputs to the Marine Environment," *Frontiers in Marine Science,* 28 March 2017.

[12] C. Van Calcar and T. Van Emmerik, "Abundance of plastis debris across European and Asian rivers," *Environmental Research Letters,* vol. 14, 2019.

[13] A. Lechner, H. Keckeis, F. Lumesberger-Loisl, B. Zens, R. Krusch, M. Tritthart, M. Glas and E. Schludermann, "The Danube so colourful: A potpourri of plastic litter outnumbers fish larvae in Europe's second largest river," *Environmental Pollution,* vol. 188, pp. 177-181, 2014.

[14] S. Rech, V. Macaya-Caquilpán, J. Pantoja, M. Rivadeneira, D. Madariage and M. Thiel, "Rivers as a source of marine litter – A study from the SE Pacific," *Marine Pollution Bulletin,* vol. 82, pp. 66-75, 2014.

[15] D. Morritt, P. Stefanoudis, D. Pearce, O. Crimmen and P. Clark, "Plastic in the Thames: A river runs through it," *Marine Pollution Bulletin,* vol. 78, no. 1-2, pp. 196-200, 2014.

[16] R. Dris, J. Gasperi, V. Rocher and B. Tassin, "Synthetic and non-synthetic anthropogenic fibers in a river under the impact of Paris Megacity: Sampling methodological aspects and flux estimations," *Science of the Total Environment,* vol. 618, pp. 157-164, 2017.

[17] J. Gasperi, R. Dris, T. Bonin, V. Rocher and B. Tassin, "Assessment of floating plastic debris in surface water along the Seine River," *Environmental Pollution,* vol. 195, pp. 163-166, 2014.

[18] C. Schmidt, T. Krauth and S. Wagner, "Export of Plastic Debris by Rivers into the Sea," *Environmental Science and Technology,* vol. 51, no. 21, pp. 12246-12253, 2017.

[19] M. Monge-Ganuzas, J. Gainza, P. Liria, I. Epelde, A. Uriarte, R. Garnier, M. González, P. Nuñez, C. Jaramillo and R. Medina, "Morphodynamic evolution of Laida beach (Oka estuary, Urdaibai Biosphere Reserve, southeastern Bay of Biscay) in response to supratidal beach nourishment actions," *Journal of Sea Research,* vol. 130, pp. 85-95, 2017.

[20] K. Kylili, I. Kyriakides, A. Artusi and C. Hadjistassou, "Idenitfying floating plastic marine debris using a deep learning approach," *Environmental Science and Pollution Research,* pp. 17091-17099, 2019.

[21] M. Wolf, K. Van den Berg, S. Garaba, N. Gnann, K. Sattler, F. Stahl and O. Zielinski, "Machine learning for aquatic plastic litter detection, classification and quantification," *Environmental Research Letters,* vol. 15, no. 11, 2020.

[22] H. Panwar, P. Gupta, K. Siddiqui, R. Morales-Menendez, P. Bhardwaj, S. Sharma and I. Sarker, "AquaVision: Automating the detection of waste in water bodies using deeptransfer learning," *Case Studies in Chemical and Environmental Engineering,* vol. 2, 2020.

[23] C. Sun, A. Shrivastava, S. Singh and A. Gupta, "Revisiting Unreasonable Effectiveness of Data in Deep Learning Era," 4 August 2017. [Online]. Available: arXiv:1707.02968v2.

[24] J. Lee, J. Su Lee, Y. Jang and S. Hong, "Distribution and Size Relationships of Plastic Marine Debris on Beaches in South Korea," *Archives of Environmental Contamination and Toxicology,* vol. 69, no. 3, 2015.

[25] M. Blettler, M. Ulla, A. Rabuffetti and N. Garello, "Plastic pollution in freshwater ecosystems: macro-, meso-, and microplastic debris in a floodplain lake," *Environmentl Monitoring and Assessment,* vol. 189, no. 11, p. 581, 2017.

[26] P. Tasseron, H. Zinsmeister, L. Rambonnet, A. Hiemstra, D. Siepman and T. Van Emmerik, "Plastic Hotspot Mapping in Urban Water Systems," *Geosciences,* vol. 10, no. 9, p. 342, 2020.

[27] P. Ryan, C. Moore, J. Van Franeker and C. Moloney, "Monitoring the abundance of plastic debris in the marine environment," *Biological Sciences,* 2009.

[28] T. Van Emmerik, M. Loozen, K. Van Oeveren, F. Buschman and G. Prinsen, "Riverine plastic emission from Jakarta into the ocean," *Environmental Research Letters,* vol. 14, no. 8, 2019.

[29] S. Hong, Y. Han, S. Kim, A. Lee and G. Kim, "Application of Deep-Learning Methods to Bird Detection Using Unmanned Aerial Vehicle Imagery," *Sensors,* vol. 19, no. 7, p. 1651, 2019.

[30] M. Geraeds, T. Van Emmerik, R. De Vries and M. Razak, "Riverine Plastic Litter Monitoring Using Unmanned Aerial Verhicles (UAVs)," *Remote Sensing,* vol. 11, no. 17, p. 2045, 2019.

[31] T. Van Emmerik, J. Seibert, B. Strobl, S. Etter, T. Den Oudendammer, M. Rutten, M. Shahrizal bin Ab Razak and I. Van Meerveld, "Crowd-Based Observations of Riverine Macroplastic Pollution," *Frontiers in Earth Science,* vol. 8, p. 298, 2020.

[32] A. Khan, A. Sohail, U. Zahoora and A. Qureshi, "A Survey of the Recent Architectures of Deep Convolutional Neural Networks," *Artifical Intelligence Review,* vol. 53, pp. 5455-5516, 2020.

[33] P. Skalski, "Gentle Dive into Math Behind Convolutional Neural Networks," 12 April 2019. [Online]. Available: https://towardsdatascience.com/gentle-dive-into-math-behind-convolutional-neural-networks-79a07dd44cf9. [Accessed 15 March 2021].

[34] P. Skalski, "Deep Dive into Math Behind Deep Networks," 2018 August 17. [Online]. Available: https://towardsdatascience.com/https-medium-com-piotr-skalski92-deep-dive-into-deep-networks-math-17660bc376ba. [Accessed 16 March 2021].

[35] S. Hochreiter, "The Vanishing Gradient Problem during Learning Recurrent Neural Nets and Problem Solutions," *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems,* vol. 6, no. 2, pp. 107-116, 1998.

[36] K. Snel, "Predicting Buckling and Plasticity of Finite Element Models Using Machine Learning - An Application of Convolutional Neural Networks on the Ultimate Strength and Stress Distribution Prediction of Stiffened Panels," Delft, 2019.

[37] H. Wu and X. Gu, "Max-Pooling Dropout for Regularization of Convolutional Neural Networks," *Neural Networks,* 2015.

[38] M. Ranzato, F. Huang, Y. Lan Boureau and Y. LeCun, "Unsupervised learning of invariant feature hierarchies with applications to object recognition," *Computer Science,* pp. 1-8, 2007.

[39] dynam.AI, "What is Computer Vision? Why Deep Learning Changed It All," [Online]. Available: https://www.dynam.ai/what-is-computer-vision-technology/. [Accessed 21 April 2021].

[40] A. Bochkovskiy, C. Wang and H. Liao, "YOLOv4: Optimal Speed and Accuracy of Object Detection," *Computer Vision and Pattern Recognition,* 2020.

[41] Y. Guo, Y. Liu, T. Georgiou and M. Lew, "A review of semantic segmentation using deep neural networks," *International Journal of Multimedia Information Retrieval,* vol. 7, pp. 87-93, 2017.

[42] C. Van Lieshout, "GitHub Repository - Automated River Plastic Monitoring Using Deep Learning and Cameras," 18 June 2020. [Online]. Available: https://github.com/colinvanlieshout/riverplasticdetection/.

[43] Swiss National Science Foundation, "CrowdWater," [Online]. Available: https://crowdwater.ch/en/crowdwaterapp-en/. [Accessed 15 June 2021].

[44] K. Hao, "Error-riddled data sets are warping our sense of how good AI really is," 1 Apri 2021. [Online]. Available: https://www.technologyreview.com/2021/04/01/1021619/ai-data-errors-warp-machine-learning-progress/. [Accessed 3 May 2021].

[45] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks For Large-Scale Image Recognition," 4 Sep 2014. [Online]. Available: https://arxiv.org/abs/1409.1556v6. [Accessed 3 May 2021].

[46] D. Kingma and J. Lei Ba, "Adam: A Method for Stochastic Optimization," 22 December 2014. [Online]. Available: https://arxiv.org/abs/1412.6980. [Accessed 5 May 2021].

[47] X. Li, S. Chen, X. Hu and J. Yang, "Understanding the Disharmony between Dropout and Batch Normalization by Variance Shift," 16 January 2018. [Online]. Available: https://arxiv.org/abs/1801.05134. [Accessed 4 May 2021].

[48] Z. Boxue, Z. Qi, F. Wenquan and L. Shuchang, "AlphaMEX: A smarter global pooling method for convolutional neural networks," *Neurocomputing,* vol. 321, no. 10, pp. 36-48, 2018.

[49] C. Shorten and T. Khoshgoftaar, "A survey on Image Data Augmentation for Deep Learning," *Journal of Big Data,* vol. 6, pp. 6-60, 2019.

[50] S. Dodge and L. Karam, "Understanding How Image Quality Affects Deep Neural Networks," 14 April 2016. [Online]. Available: https://arxiv.org/abs/1604.04004. [Accessed 6 May 2021].

[51] R. Malli, "GitHub Repository - SqueezeNet Implementation Keras," 19 February 2019. [Online]. Available: https://github.com/rcmalli/keras-squeezenet.

[52] C. Chen, A. Liaw and L. Breiman, "Using Random Forest to Learn Imbalanced data," 1 July 2004. [Online]. Available: https://statistics.berkeley.edu/sites/default/files/tech-reports/666.pdf. [Accessed 5 May 2021].

[53] H. He and E. Garcia, "Learning from Imbalanced Data," *IEEE Transactions on Knowledge and Data Engineering,* vol. 21, no. 9, pp. 1263 - 1284, 2009.

[54] Fawcett, "An introduction to ROC analysis," *Pattern Recognition Letters,* vol. 27, no. 8, pp. 861-874, 2006.

[55] Tzutalin, "LabelImg - Github code," 2015. [Online]. Available: https://github.com/tzutalin/labelImg.

[56] The AI Guy, "YOLOv4 Cloud Tutorial," 31 December 2020. [Online]. Available: https://github.com/theAIGuysCode/YOLOv4-Cloud-Tutorial.

[57] A. Bochkovskiy, "GitHub Repository - Darknet," 2020. [Online]. Available: https://github.com/AlexeyAB/darknet.

[58] R. Padilla, W. Passos, T. Dias, S. Netto and E. Da Silva, "A Comparative Analysis of Object Detection Metrics with a Companion Open-Source Toolkit," *Electronics,* pp. 237-242, 2021.

[59] R. Khandelwal, "Convolutional Neural Network: Feature Map and Filter Visualization - Learn how Convolutional Neural Networks understand images," 18 May 2020. [Online]. Available: https://towardsdatascience.com/convolutional-neural-network-feature-map-and-filter-visualization-f75012a5a49c. [Accessed 17 June 2021].

[60] M. Rajnoha, R. Burget and L. Povoda, "Image Background Noise Impact on Convolutional Neural Network Training," *International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT),* 2018.

[61] P. Renard, A. Alcolea and D. Ginsbourger, "Stochastic versus Deterministic Approaches," *Environmental Modelling,* vol. 2, pp. 133-149, 2013.

[62] A. Azulay and Y. Weiss, "Why do deep convolutional networks generalize so poorly to small image transformations?," *Machine Learning Research,* vol. 20, pp. 1-25, 2019.

[63] N. Patricia and B. Caputo, "Learning to Learn, from Transfer Learning to Domain Adaptation: A Unifying Perspective," *IEEE Conference on Computer Vision and Pattern Recognition,* 2014.

[64] C. Zhang, F. Rameau, S. Lee and J. Kim, "Revisiting Residual Networks with Nonlinear Shortcuts," in *British Machine Vision Conference (BMVC)*, Cardiff, 2019.

[65] C. Zhang, P. Benz, D. Argaw, S. Lee, J. Kim, F. Rameau, J. Bazin and I. Kweon, "ResNet or DenseNet? Introducing Dense Shortcuts to ResNet," *Computer Vision and Pattern Recognition,* 2020.

[66] G. Huang, Z. Liu, L. Van der Maaten and K. Weinberger, "Densely Connected Convolutional Networks," *Computer Vision and Pattern Recognition,* 2018.

[67] C. Martin, S. Parkes, Q. Zhang, X. Zhang, M. Mccabe and C. Duarte, "Use of unmaned aerial vehicles for efficient beach litter monitoring," *Marine Pollution Bulletin,* vol. 131, pp. 662-673, 2018.

[68] L. Xue, S. Yang, Y. Li and J. Ma, "An automatic shadow detection method for high-resolution remote sensing imagery based on polynomial fitting," *International Journal of Remote Sensing,* vol. 40, pp. 2986-3007.

[69] T. De Vries, I. Misra, C. Wang and L. Van der Maaten, "Does Object Recognition Work for Everyone?," 18 June 2019. [Online]. Available: arXiv:1906.02659v2.

[70] L. Biermann, D. Clewley, V. Martinez-Vicente and K. Topouzelis, "Finding Plastic Patches in Coastal Waters using Optical Satellite Data," *Scientific Reports,* vol. 10, no. 1, 2020.

[71] K. Themistocleous, C. Papoutsa, S. Michaelides and D. Hadjimitsis, "Investigating Detection of Floating Plastic Litter from Space Using Sentinel-2 Imagery," *Remote Sensing,* vol. 12, no. 16, 2020.

[72] S. Tyler, O. Jensen, Z. Hogan, S. Chandra, L. Galland and J. Simmons, "Perspectives on the Application of Unmanned Aircraft for Freshwater Fisheries Census," *Fisheries,* vol. 43, no. 11, 2018.

[73] S. Broere, "The Sound of Plastic - A proof-of-concept for detecting suspended riverine macroplastics with echo sounding," Delft, 2020.

[74] C. Luo, X. He, J. Zhan, L. Wang, W. Gao and J. Dai, "Comparison and Benchmarking of AI Models and Frameworks on Mobile Devices," 7 May 2020. [Online]. Available: https://arxiv.org/abs/2005.05085. [Accessed 27 May 2021].

[75] N. Adaloglou, "Intuitive Explanation of Skip Connections in Deep Learning," 23 March 2020. [Online]. Available: https://theaisummer.com/skip-connections/. [Accessed 27 May 2021].

[76] Pröve and PL:, "MobileNetV2: Inverted Residuals and Linear Bottlenecks," 11 April 2018. [Online]. Available: https://towardsdatascience.com/mobilenetv2-inverted-residuals-and-linear-bottlenecks-8a4362f4ffd5.

[77] A. Ng, "C4W3L04 Convolutional Implementation Sliding Windows," 7 November 2017. [Online]. Available: https://www.youtube.com/watch?v=XdsmlBGOK-k&list=PL_IHmaMAvkVxdDOBRg2CbcJBq9SY7ZUvs&index=5&ab_channel=DeepLearningAI. [Accessed 29 May 2021].

[78] P. Rugery, "Explanation of YOLO V4 a one stage detector," 7 September 2020. [Online]. Available: https://becominghuman.ai/explaining-yolov4-a-one-stage-detector-cdac0826cbd7. [Accessed 29 May 2021].

[79] A. Choulwar, "The Art of Convolutional Neural Network," 12 April 2019. [Online]. Available: https://medium.com/@achoulwar901/the-art-of-convolutional-neural-network-abda56dba55c. [Accessed 20 April 2021].

[80] F. Li, J. Johnson and S. Yeung, "Lecture 11: Detection and Segmentation," 10 May 2017. [Online]. Available: http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture11.pdf. [Accessed 21 April 2021].

[81] K. He, S. Zhang, S. Ren and J. Sun, "Deep Residual Learning for Image Recognition," 10 December 2015. [Online]. Available: https://arxiv.org/abs/1512.03385. [Accessed 27 May 2021].

[82] Q. Guan, X. Wan, H. Lu, B. Ping, D. Li, L. Wang, Y. Zhu, Y. Wang and J. Xiang, "Deep convolutional neural network Inception-v3 model for differential diagnosing of lymph node in cytological images: a pilot study," *Annals of Translational Medicine,* vol. 7, no. 14, 2019.

Appendices

A – Datasets

A.1 – Impression of Green Village images per class

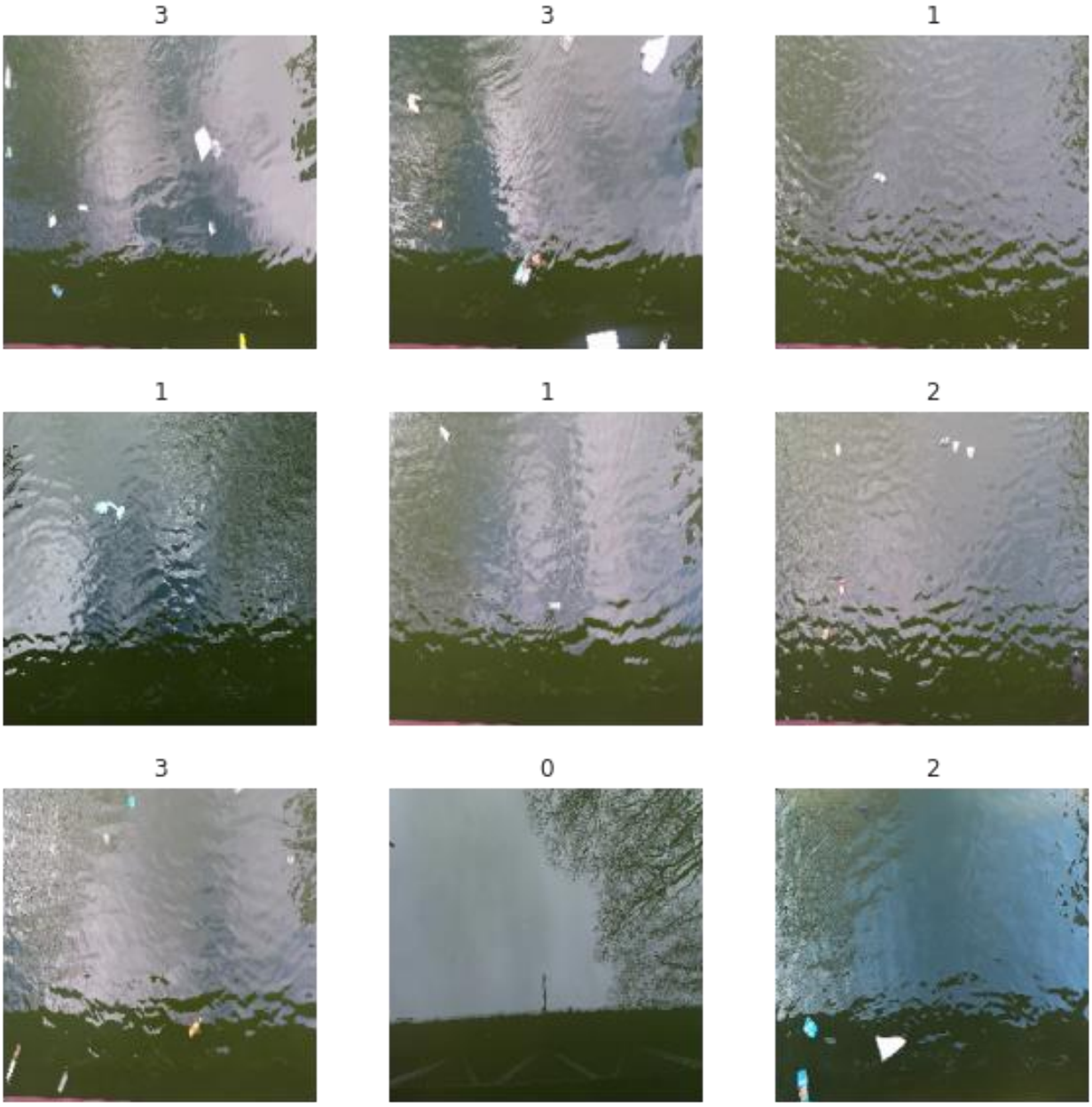


Figure A.1 Impression of Green Village data per class with resized images (224x224)

B – Model architectures

B.1 – Stepwise operations for baseline model construction

Operation	Methods
Configuration convolutional layers	Opt1: 16,16 - 32,32 Opt2: 16,16 - 32,32 - 64,64 Opt3: 16,16 - 32,32 - 64,64 - 128,128 Opt4: 16,16 - 32,32 - 64,64 - 128,128 - 256,256 Opt5: 16 - 32 - 64 - 128 - 256
Maximum or average pooling	Testing either maximum or average pooling after every convolutional block
Batch Normalization	Momentum values between 0.5 – 0.9
Dropout	Dropout values between 0.2 – 0.5
Flatten, Global Average or Maximum Pooling	Configurations with either Flatten, Global Average, Global Maximum or a Hybrid Global Pooling layer
Fully-connected layer	Keras tuner testing for 1-4 dense layers and predefined number of neurons per layer (16, 32, 64, 128, 256)
Data Augmentation	Influence of horizontal and vertical flipping, brightening, darkening and noise

B.2 – ResNet50

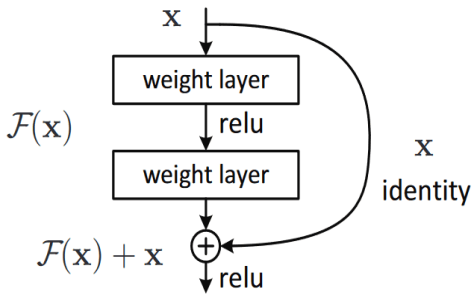


Figure C.3.1 Concept of skip connection ResNet (image from [81])

ResNet50 is the most popular CNN architecture amongst all ResNet architectures [74]. The novelty about ResNet is the introduction of a concept called 'skip connection'. With this procedure layers in the neural network are skipped and the output of a layer is fed as input to the following layers. This is done instead of only feeding this output to the next layer. The main idea, is that the process of backpropagation is run through the identity function by the use of vector addition [75]. Since the gradient will be multiplied by one, the value of the gradient in the earlier layers will be maintained. The use of identity mapping consequently mitigates the effects of a vanishing gradient. The main concept of Residual Networks (ResNets) is also depicted in Figure C.3.1. The residual blocks are stacked together and the identify function is used for the preservation of the gradient. Another reason for the use of skip connections is the transport of information from earlier layers. For a large majority of tasks it is desired, that information retrieved in earlier layers is also used for learning in deep layers. According to [75], the learned features in earlier layers hold semantic information to some extent. Without the use of skip connections, the generated abstraction level of this information would have been too high.

B.3 – InceptionV3

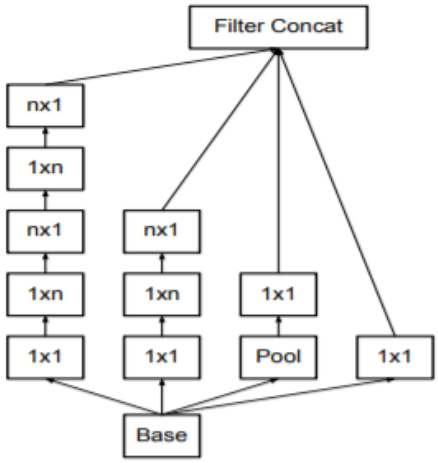


Figure C.4.1 Main principle of Inception module (images taken from [82])

The typical Inception module is presented in Figure C.4.1. Inception networks use filter sizes of 1x1, 3x3 and 5x5 for different branches to extract multiscale information [74]. The main design criteria of an Inception network are the computational efficiency and practicality [74]. Instead of using one large filter size, these networks separate larger filter sizes into smaller convolutions by using e.g., two stacked convolution filters (3x3) rather than a 5x5 filter [74]. Furthermore, Inception modules also use asymmetrical convolutions e.g using a 1x3 followed by 3x1 convolutional filer, instead of a 3x3 filter. These smaller convolutions are used to reduce the number of parameters. In principle, the Inception module concatenates feature maps produced by varying small filter sizes.

B.4 – DenseNet121

To ensure maximum flow of information between all the layers, DenseNet models heavily utilize the concatenation of feature maps. DenseNet connects all the layers with each other by means of concatenation [74]. Every layer concatenates all the previous layers and passes itself to all the following layers, as depicted in Figure C.5.1. It is suggested by [74], that DenseNet models mitigate the vanishing gradient problem and further boost the propagation of features. Due to the narrowness of each output layer, the number of parameters are greatly reduced compared to ResNet50 and InceptionV3. However, the dense blocks commonly have much larger inputs due to the concatenation of preceding layers. For the reduction of inputs and consequently the improvement of computational efficiency, a 1x1 convolution is used as bottleneck feature before every 3x3 convolution [74]. For the input passed into a 1x1 convolution the height and width of the input remain the same, however the third dimension or filter dimension can be altered to reduce the dimensionality. Overall, DenseNet models result in more compacted models whilst ensuring a high level of feature reusability.

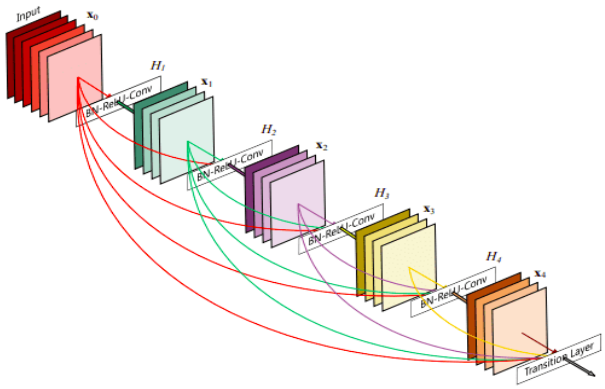


Figure C.5.1 DenseNet principle of feature map concatenation (image taken from [66]

B.5 – MobileNetV2

MobileNet was originally designed for mobile and embedded applications and is considered as a lightweight architecture due to its low number of parameters [74]. This architecture makes use of depthwise separable convolutions, which divides convolutional operations into a depth- and pointwise convolution. Depthwise convolution refers to the application of a 3x3 filter that extracts the spatial relation between features, whereas pointwise convolution uses a 1x1 convolution to capture the relation between channel features [74] (Figure C.6.1).

Furthermore, MobileNetV2 uses the following two optimized mechanisms: inverted residual structure and linear bottlenecks [74]. The inverted residual structure follows the same principal as the residual blocks for ResNet50, however MobiletNet architectures follow a narrow (1x1) - widen (3x3) – narrow (1x1) pattern. It can be seen in Figure C.6.1, that the inverted residual block connects the narrow layers with a skip connection. Due to the squeezing of layers where the skip connections are present, the performance of the network is negatively influenced. With a linear bottleneck, the non-linearities in the narrow layers are removed [74] and consequently the last convolutional layer of a residual block produces a linear output that is combined with the initial activations [76].

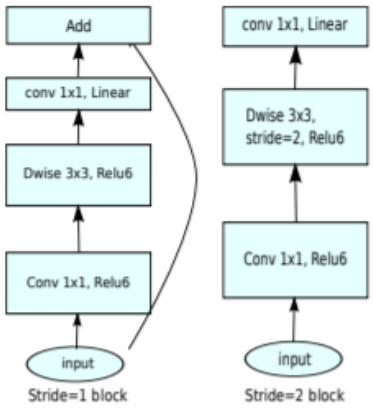


Figure C.6.1 Concept of MobileNetV2

B.6 – SqueezeNet

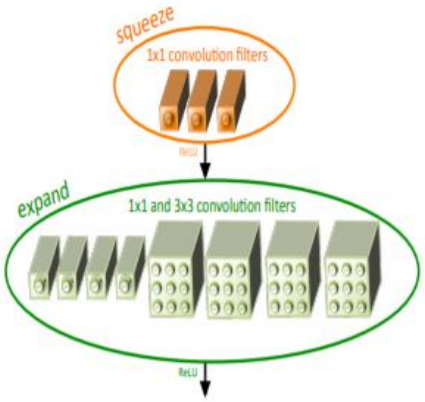
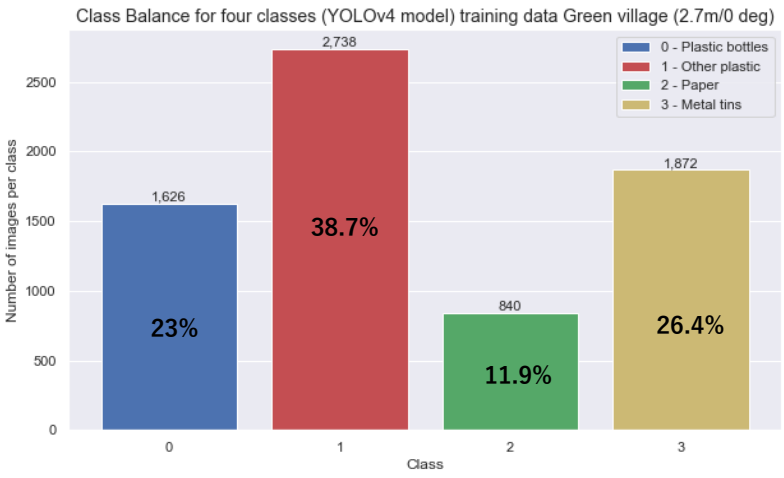


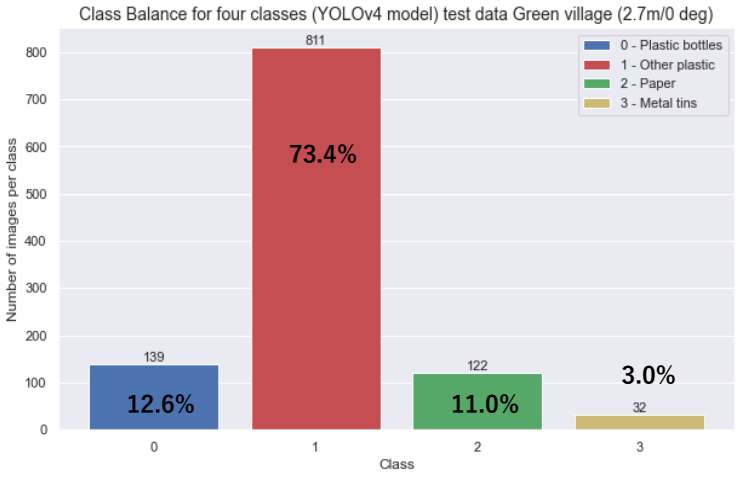
Figure C.7.1 Main principle of SqueezeNet architecture

The primary aim of SqueezeNet is to reduce the number of parameters while maintaining an acceptable accuracy [74]. The SqueezeNet architecture follows three main principles for the reduction of parameters and consequently computational time: replacement of 3x3 filter with 1x1 filters, decreasing the number of input channels and downsampling deep in the network such that convolution layers produce large feature maps [74]. This is achieved by a 'fire module' that uses 'squeeze' and 'expand' convolutions. In the squeeze module, the input is fed through a 1x1 convolution, which is followed by an 'expand' module that comprises an expansion with a combination of 1x1 and 3x3 convolutions. SqueezeNet contains 50x less parameters (1.25 million) than AlexNet.

C – Object Detection

C.1 – Class Balance for four classes (YOLOv4 model) for training and testing data





C.2 – Intuitive description of object detection for YOLO algorithm

Compared to an image classification task, where the softmax unit outputs the probabilities for every class if the input image is considered, object detection outputs additional values which represent the characteristics of the bounding box (x and y coordinate, width and height). All the parameter values can be given in with the following vector:

$$y = \begin{bmatrix} p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \end{bmatrix}$$

whereby y represents the output vector, p_c the probability that one of the classes is present (0 - 1), b_x the x coordinate for the centroid of the rectangle, b_y the y coordinate for the centroid of the rectangle, b_h the height rectangle, b_w the width of the rectangle, c_1 presence of class 1 ('pedestrians'), c_2 presence of class 2 ('car') and c_3 presence of class 3 ('motorcycle'). The values for each class is either 0 or 1, depending on the presence of the object in the image.

One approach to object detection algorithms is a sliding windows approach, whereby a small rectangular box slides over an image with a specified stride. These cropped image regions are then fed to a convolutional network (ConvNet), whereby the ConvNet determines whether an object is present in the cropped region. This process can then be repeated with larger sliding windows. One large disadvantage of this method is the large computational costs, due to the sliding windows over an image, extracting these regions and feeding them independently to a ConvNet. By choosing a larger stride, the computation cost can be reduced, however this could be associated with lower accuracies for detecting an object, caused by the larger granularity due to the increased stride.

The solution is a convolutional implementation of the sliding window approach. Instead of passing cropped regions sequentially to a model, the entire image can be implemented and all the predictions can be made convolutionally at the same time. If it is assumed that the input for the ConvNet are 14x14x3 images, the test image is 16x16x3 and stride is 2 pixels, the ConvNet is run four times to retrieve four labels corresponding to the image regions. The computations done by these ConvNets is highly redundant due to the high overlap between the sliding window

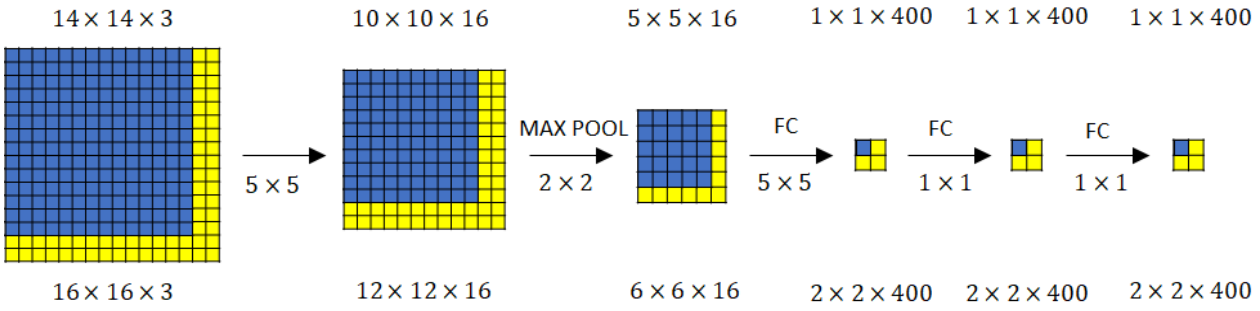


Figure D.2.1 Convolutional implementation of sliding window. The top row gives the dimension of the blue grid (normal sliding window approach) and the bottom row the dimension with inclusion of the yellow grid (convolutional sliding window)

regions. By the implementation of a convolutional sliding window approach, a lot of computation can be shared through these four forward passes.

This results in the fact that instead of having a 1x1x4 output volume for each sliding window region, a 2x2x4output volume is created with a convolutional implementation, if the last layer in Figure D.2.1 is run through a 1x1 filter. The blues subset (1x1x4) represents the results for the upper left corner (14x14 image) and the other three yellow squares (1x1x4 volume) give the results for the upper right, lower left and bottom right for a 14x14 image.

In conclusion, a convolutional implementaiton combines four forward passes into one compuation, whereby a lot of the computations are shared in the regions where the image is identical [77].

The convolutional implementation of sliding windows is computationally more efficient, however outputting the most accurate bounding box prediction still remains a challenge. If the image in Figure D.2.2 is considered, for each grid cell the output vector y is outputed. The total output for this image would correspond to a 3x3x8 volume. Consequently, the 100x100x3 input should be mapped to 3x3x8 output volume with a ConvNet. In this example, if an object is present, the p_c component should be equal to 1 for the middle left and right grid cell. Nevertheless, it needs to be mentioned, that this approach works well when only one object is present in a grid cell. Commonly in practice, a finer grid size is also used, to avoid the presence of multiple objects in a grid cell. For the object, the midpoint of target is considered, which is then assigned to the specific grid cell. The x and y coordinate, the height and the width are specified relative to the grid cell. Values of the x and y coordinate are between 0 – 1. The height and the width can have a larger value than 1, if the size of the bounding box exceeds the size of a single grid cell. Once the model calculates a bounding box for a certain object, the predicted bounding box can be compared to the ground truth bounding box. This is done by the Intersection over Union (IoU), which describes the degree of overlap between two bounding boxes (as described in section 3.4.3). Most commonly an IoU threshold of 0.5 is used.

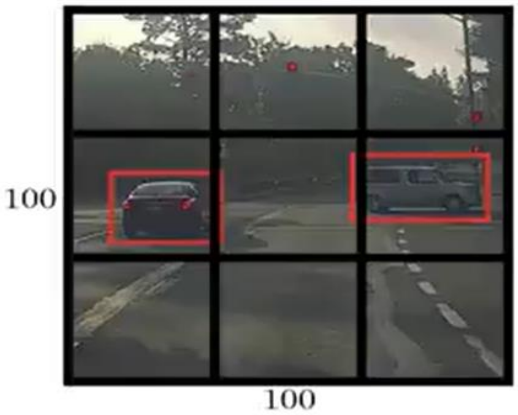


Figure D.2.2 Example for bounding box predictions with a 3x3 grid (image taken from [77])

Figure D.2.3 Example of multiple detection with different probabilities for the predicted bounding boxes (image taken from [77]))

One occuring problem is that the applied algorithm might find a multitude of detections for the same object (Figure D.2.3). A method named 'nonmax suppression' is a method which ensures, that the algorithm detects each obejct only once. If many grid cells are used in the computation, the probability of more than grid cell indicating the presence of the centroid of the object is higher. The model firstly outputs the probability of every bounding box prediction. The model then selects the most confident bounding box prediction.

Afterwards the 'nonmax suppression' method the evaluates all of the remaining rectangles. All bounding boxes that have a high IoU (>0.5, since bounding box predictions below that threshold will be omitted) with the most confident prediction will then be suppressed. In the case of Figure D.2.3, the bounding box predictions with a probability of 0.8 and 0.9 will be retained, whilst the remaining rectangles will be discarded.

The last remaining problem is the fact, that each object in an image is assigned to a specific grid cell, which contains the centroid of the object. To detect multiple objects within a grid cell, the idea of anchor boxes can be used (Figure D.2.4). Beforehand, predefined shapes (or anchor boxes) are defined by the model whereby two predictions can be associated with these two anchor boxes. Therefore, output vector y will not only contain 8 entries, but 16 entries per grid cell, since two anchor boxes fall into the same grid cell for this example. The parameters c_1, c_2, c_3 then decode to which class a certain anchor box belongs. In principle, the object is still assigned to a grid, however the object is also assigned to an anchor box for this specific grid cell with the highest IoU value. In practice, more than two anchor boxes are usually applied.

Figure D.2.4 Example of anchor boxes for two different objects in a grid cell

This section was intended to give the reader an intuitive explanation with respect to the YOLO algorithm. Since there are different versions of the YOLO algorithm, the following section will briefly introduce the YOLOv4 model architecture.

D.3 – Model architecture YOLOv4

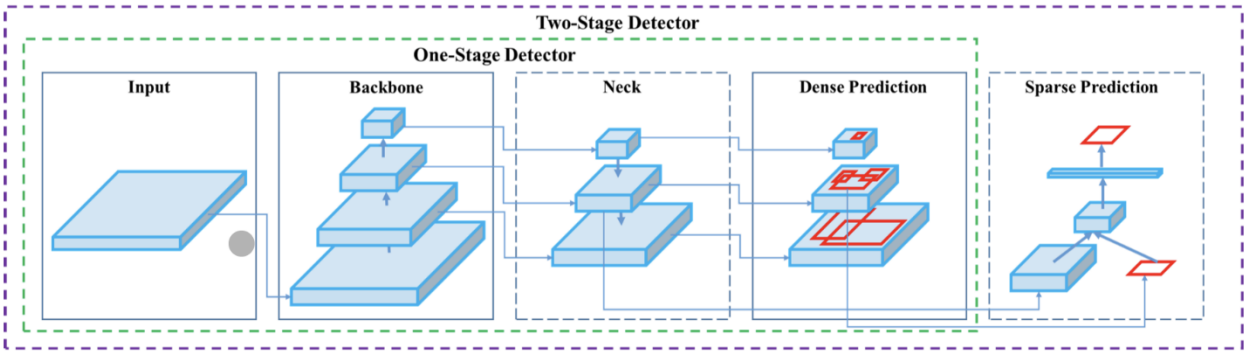


Figure D.3.1 Typical model architectures of object detection algorithms with one- and two stage detectors

Typically, there are two types of model architectures for object detection algorithms, namely one or two stage detectors. An one stage detector is capable of detecting an object without the need of a preliminary step (e.g., YOLO algorithms). Two stage detectors use a preliminary step to generate a region of interest that is then used for object detection and the prediction of bounding boxes (e.g., R-CNN or Faster R-CNN). This also suggests, that the use of an one-stage detector consumes less time. In the backbone of an object detection model, the essential features in the form of feature maps are extracted from an input image. In the case of the YOLOv4 model

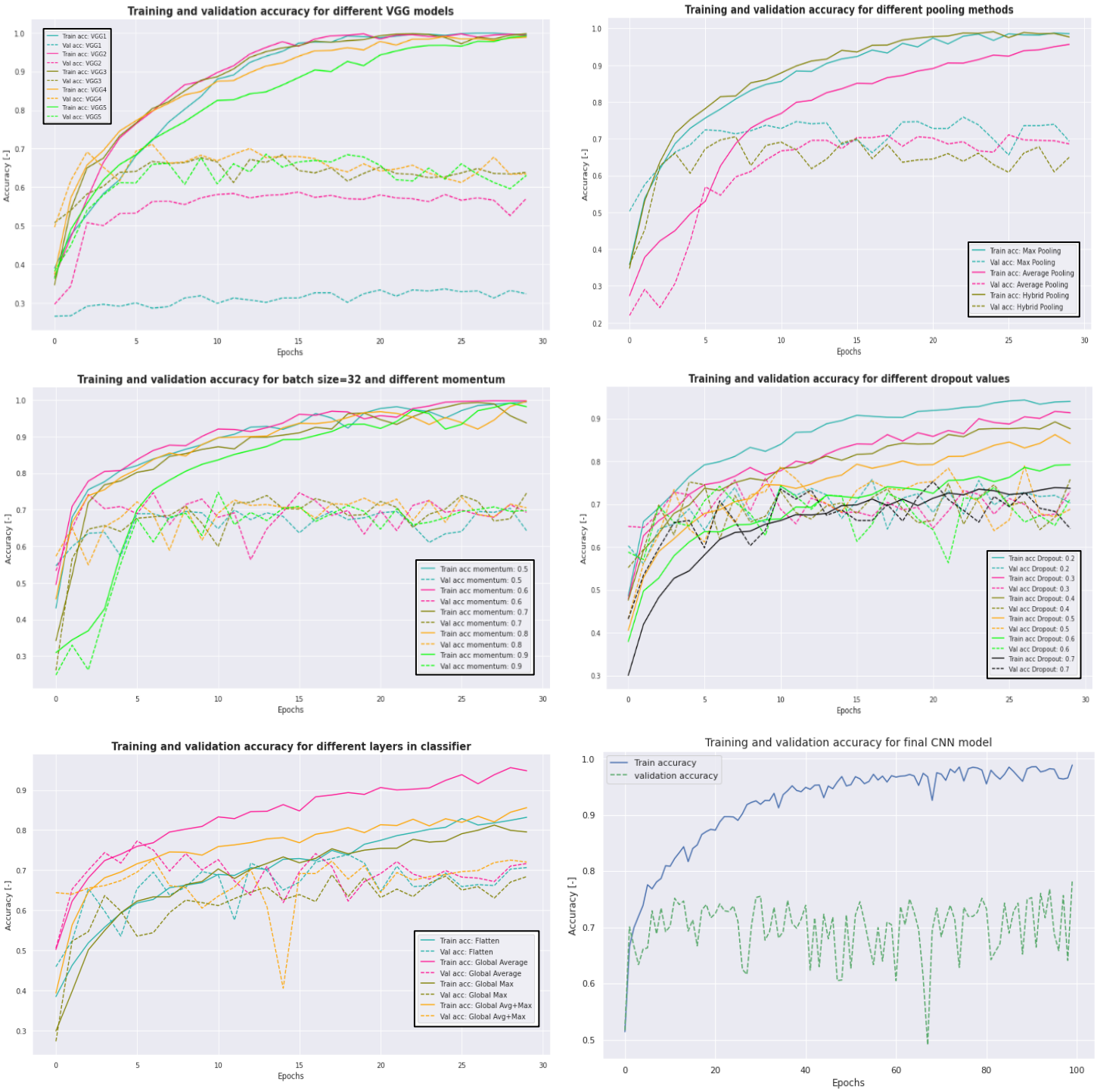
CSPDarknet53 is used. The neck and head ('dense prediction' in Figure D.3.1) are subsets of the backbone, which aim to improve the feature discriminability and robustness of the model. For YOLOv4, Spatial Pyramid Pooling (SPP) and a Path Aggregation Network (PAN) are used. SPP removes the constraint of the network to input a fixed image size and PAN allows for a better propagation and transport of layer information through the entire network [78]. The head (or the detector) finally outputs a vector prediction (y), as described in the previous section (Appendix D.2). For this component the head of YOLOv4 was used for the YOLOv4 model.

In the backbone and detector YOLOv4 makes use of 'bag of freebies' and 'bag of specials'. The bag of freebies refers to methods that only change the training strategy or only increase the training [40], while limiting the cost of inference. These methods comprise data augmentation, the use of focal loss due to semantic bias caused by class imbalance, label smoothing and improvements to the objective function of Bounding Box (BBox) regression [40]. The bag of specials increases the inference cost by a minor amount but in return the accuracy of object detection is greatly improved. These methods enlarge the receptive field, introduce an attention mechanism, strengthen feature integration and introduce a post-processing step to assess the model prediction results.

This section aimed to give a brief introduction to the YOLOv4 model. The entire model comprises an enormous amount of steps and computations. All technical details and subtleties are not within the scope of this research. For more in depth explanation the original YOLOv4 research paper [40] or [78] can be utilized.

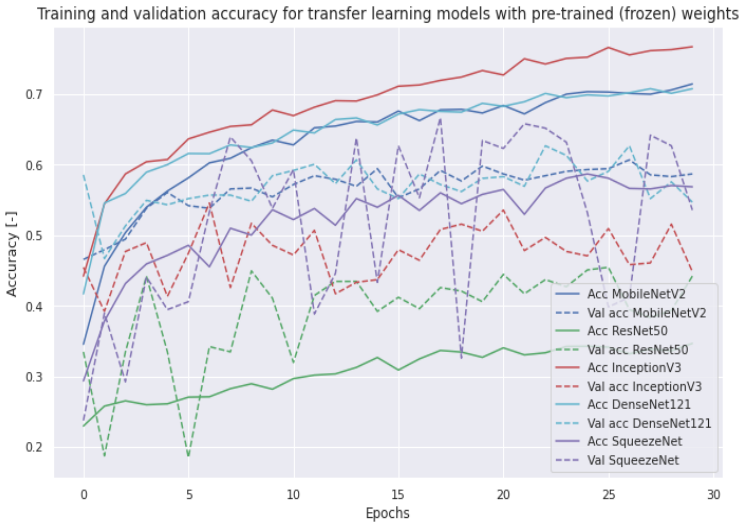
D – Model Results

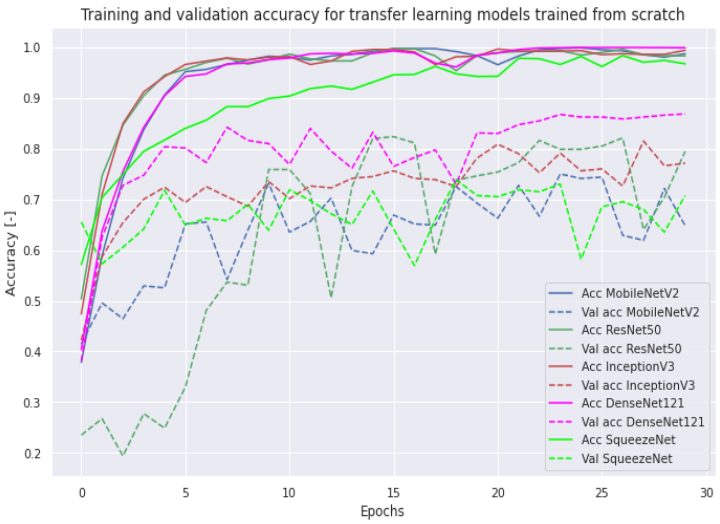
D.1 – Baseline model



Appendix E.1 Training and validation accuracies for A: Convolutional Blocks, B: Pooling Methods, C: Batch Normalization, D: Dropout, E: Classifiers, F: Final CNN model

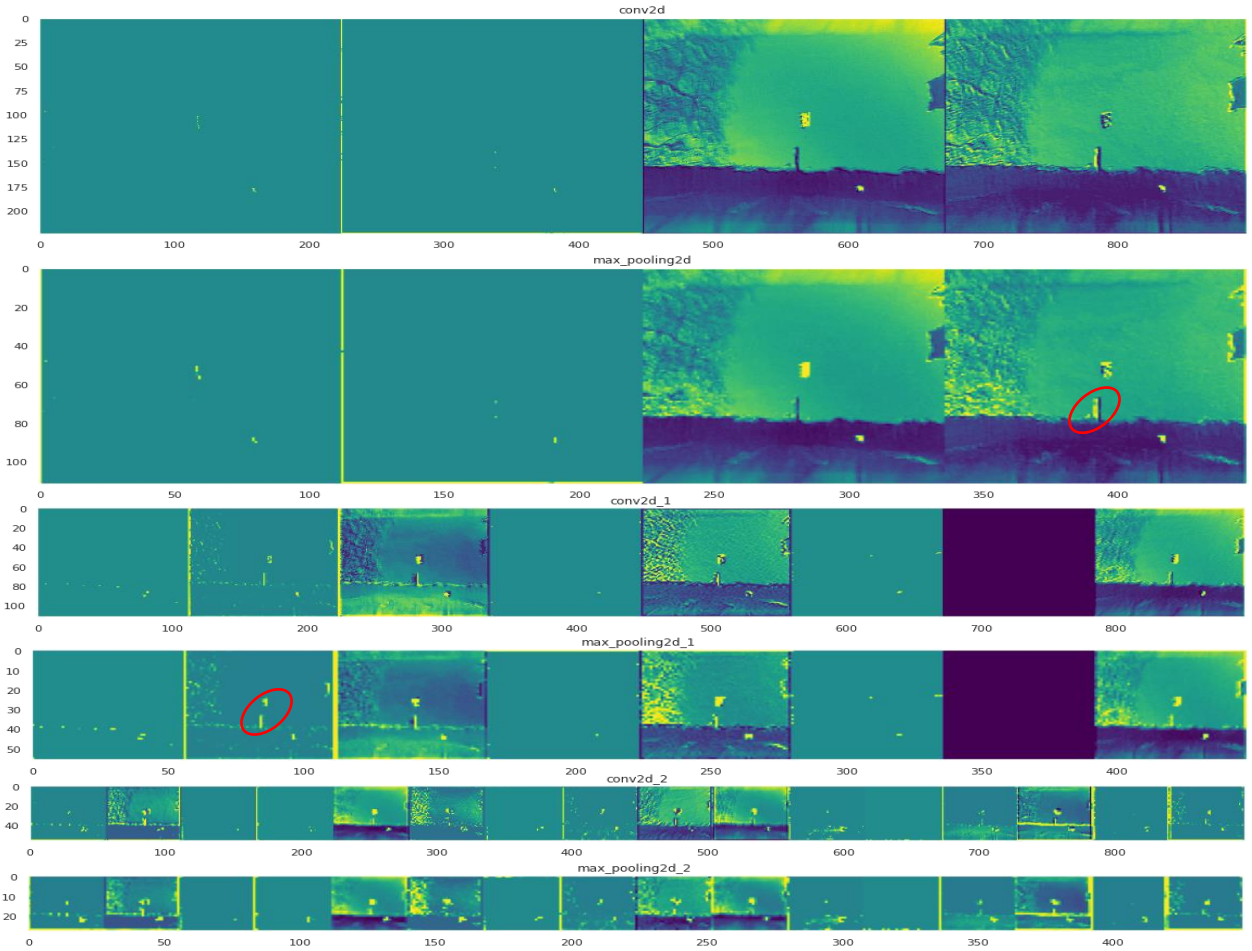
D.2 – Transfer Learning



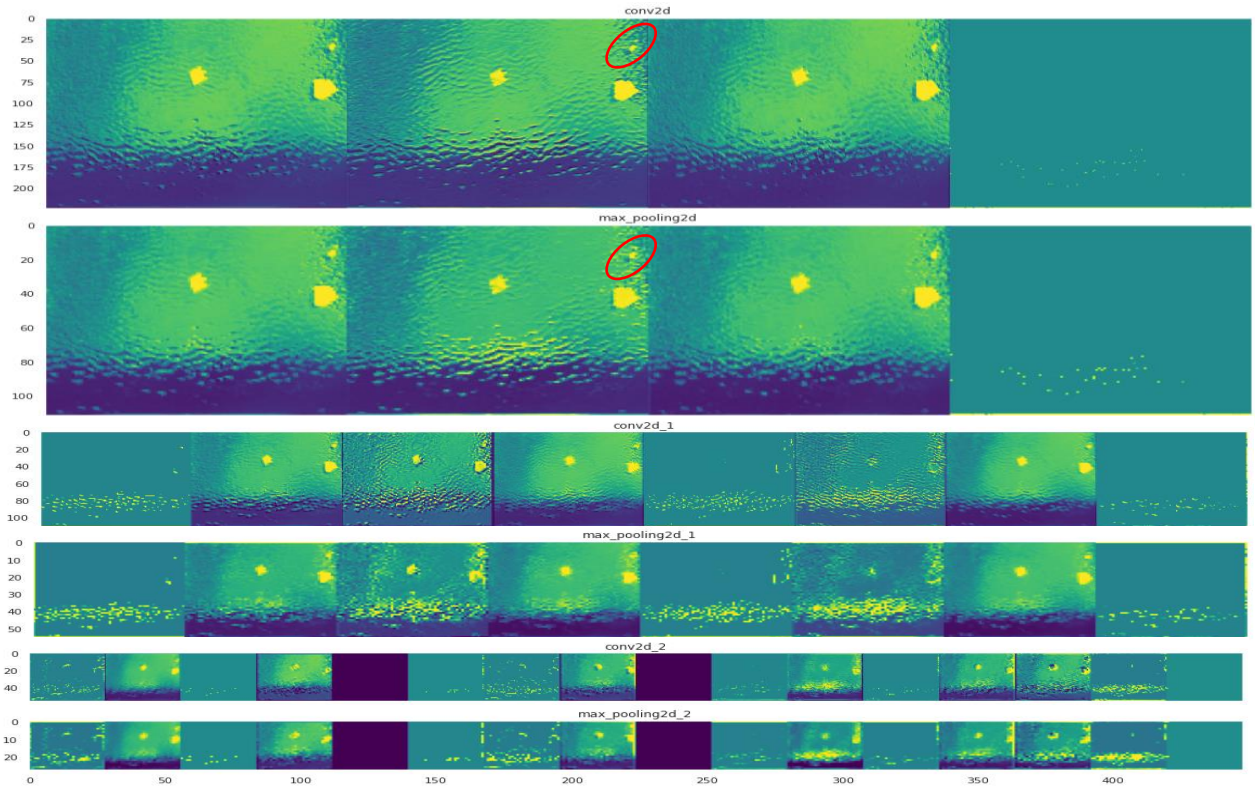


D.3– Visualisation of feature maps in convolutional operations

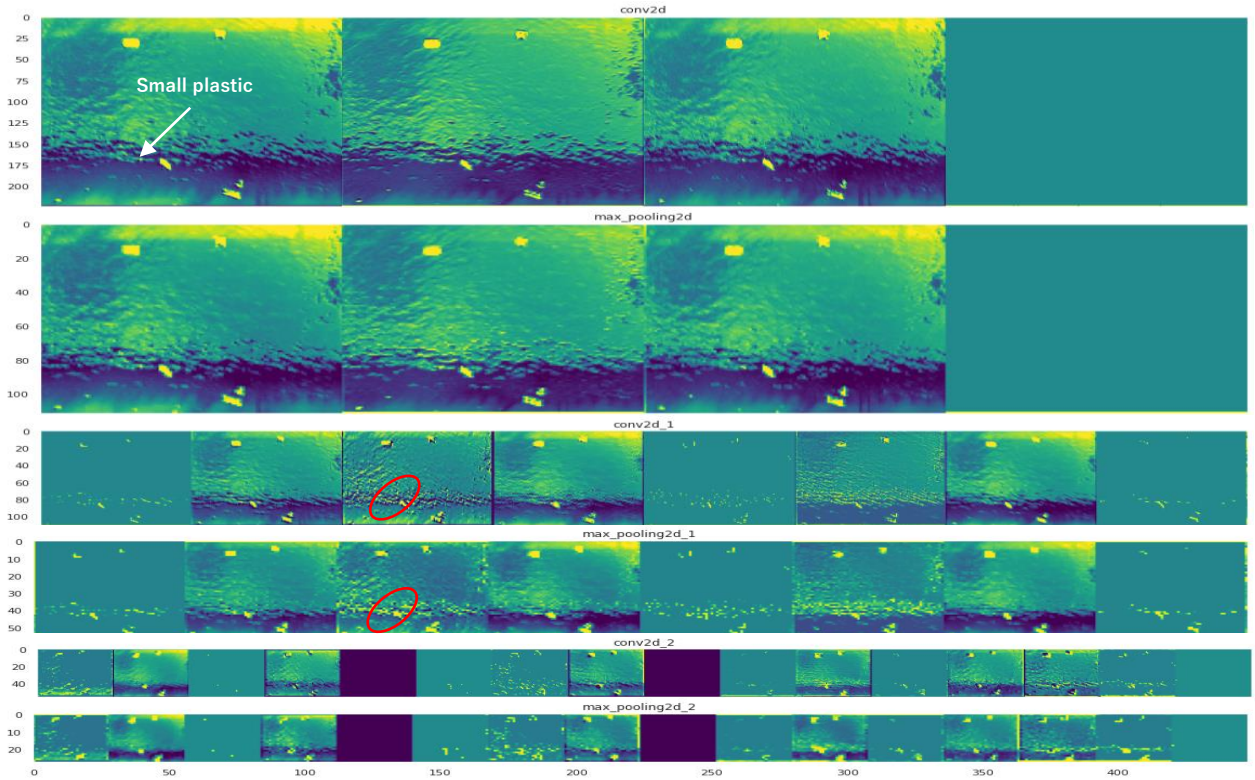
Influence of other reflected objects on the water surface (Image 3 in Figure 19)



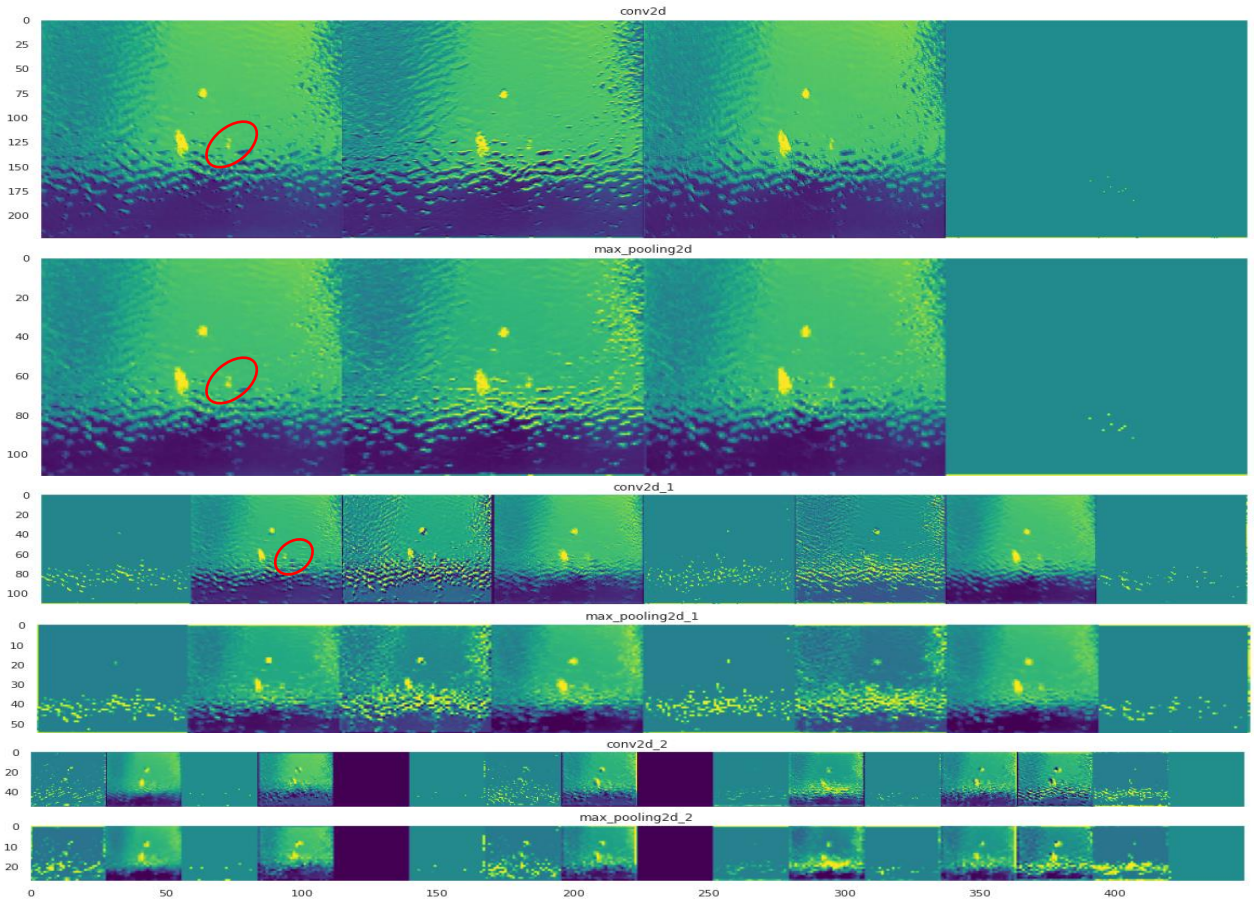
Plastic debris object with similar colour as water (Image 4 in Figure 19)



Small plastic debris object (Image 5 in Figure 19)



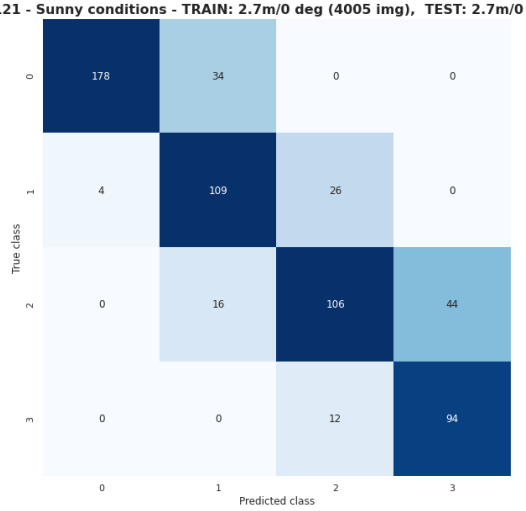
Submerged plastic debris object (Image 6 in Figure 19)



D.4 – Confusion matrices and statistical metrics for the impact of environmental and instrumental factors (Train 1: 2.7m/0 deg)

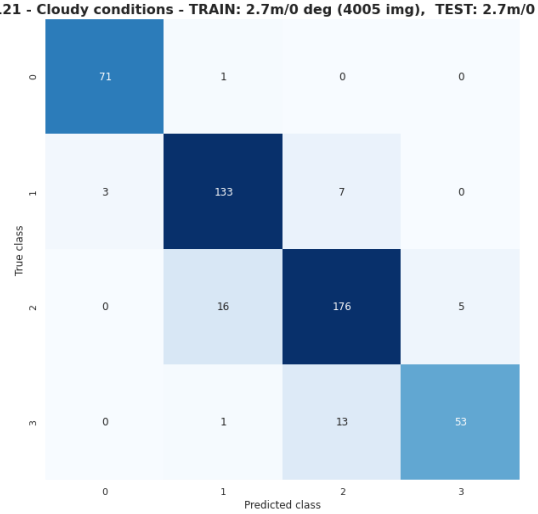
DenseNet121 - Sunny conditions - TRAIN: 2.7m/0 deg (4005 img), TEST: 2.7m/0 deg (623 img)

(A)



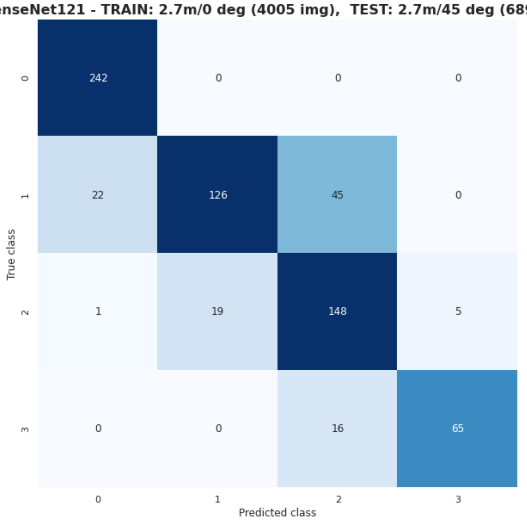
DenseNet121 - Cloudy conditions - TRAIN: 2.7m/0 deg (4005 img), TEST: 2.7m/0 deg (479 img)

(B)



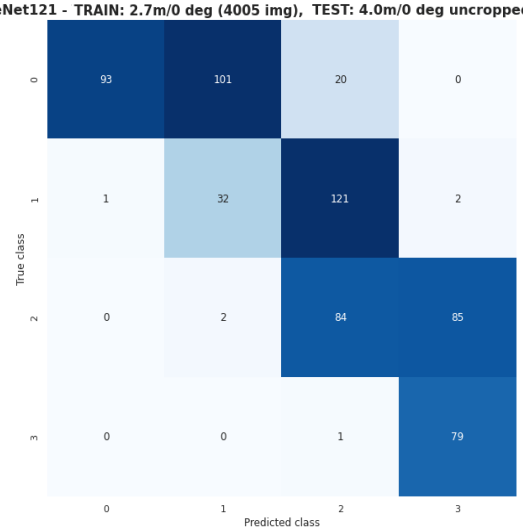
DenseNet121 - TRAIN: 2.7m/0 deg (4005 img), TEST: 2.7m/45 deg (689 img)

(C)



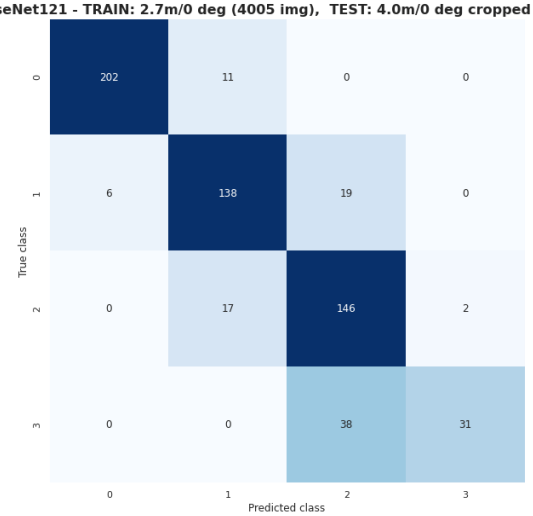
DenseNet121 - TRAIN: 2.7m/0 deg (4005 img), TEST: 4.0m/0 deg uncropped (621 img)

(D)



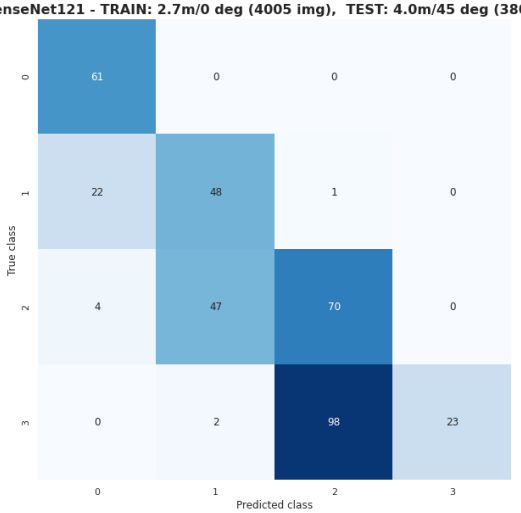
DenseNet121 - TRAIN: 2.7m/0 deg (4005 img), TEST: 4.0m/0 deg cropped (621 img)

(E)

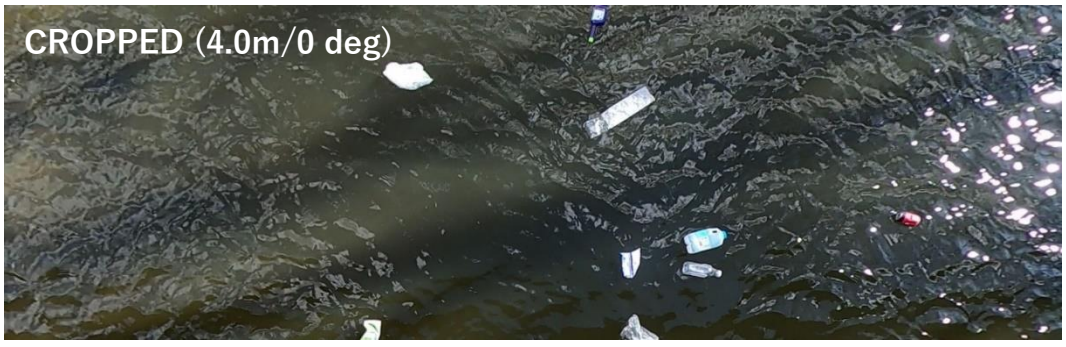


DenseNet121 - TRAIN: 2.7m/0 deg (4005 img), TEST: 4.0m/45 deg (380 img)

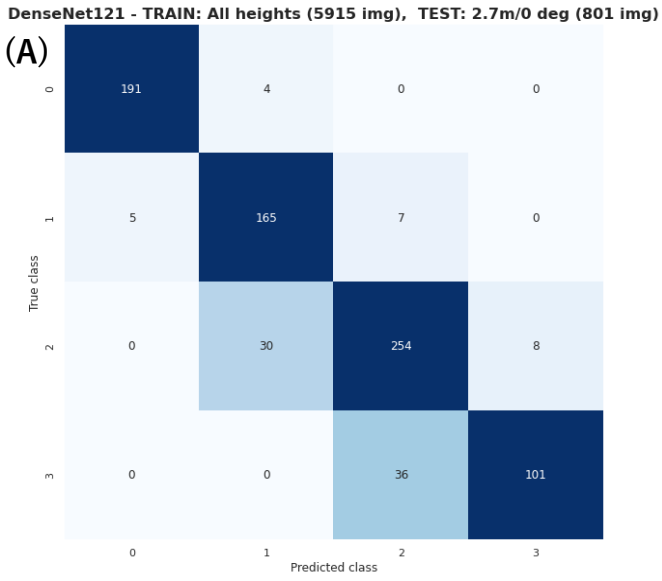
(F)

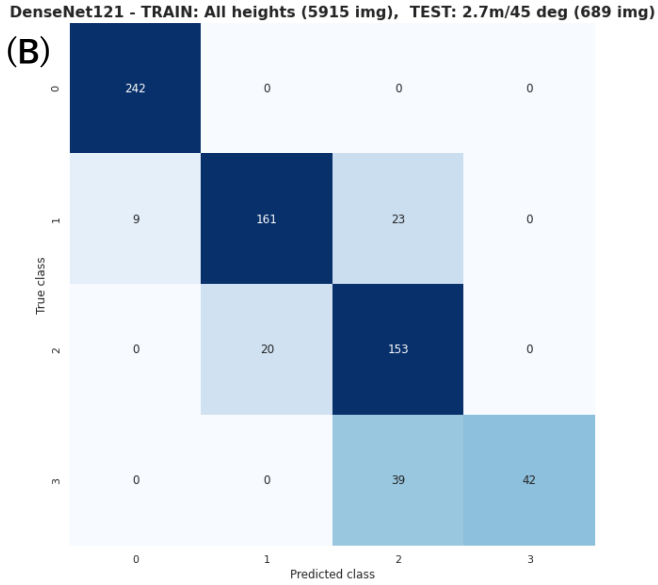


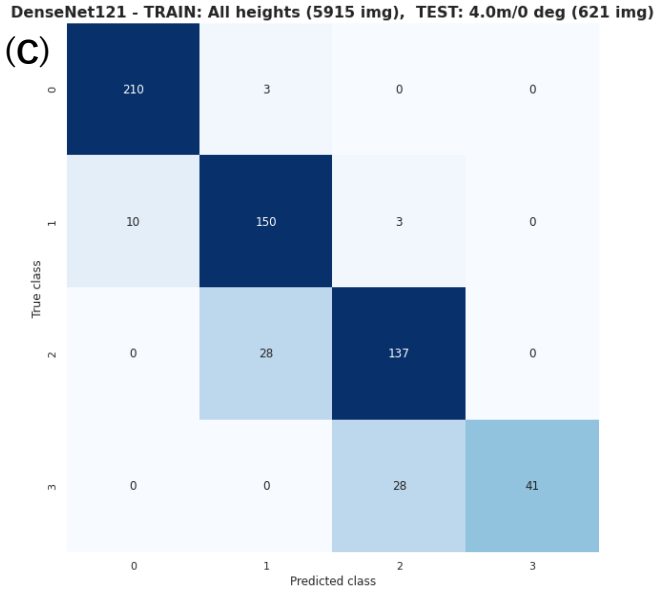
	TEST-SUNNY (A) F1-score	TEST-CLOUDY (B) F1-score	TEST 1 (C) F1-score	TEST 2 Uncropped (D) F1-score	TEST 2 Cropped (E) F1-score	TEST 3 (F) F1-score
Class 0	0.90	0.97	0.95	0.60	0.96	0.82
Class 1	0.73	0.90	0.75	0.22	0.84	0.57
Class 2	0.68	0.90	0.77	0.42	0.79	0.48
Class 3	0.77	0.85	0.86	0.64	0.61	0.32
Accuracy	0.78	0.90	0.84	0.46	0.85	0.54

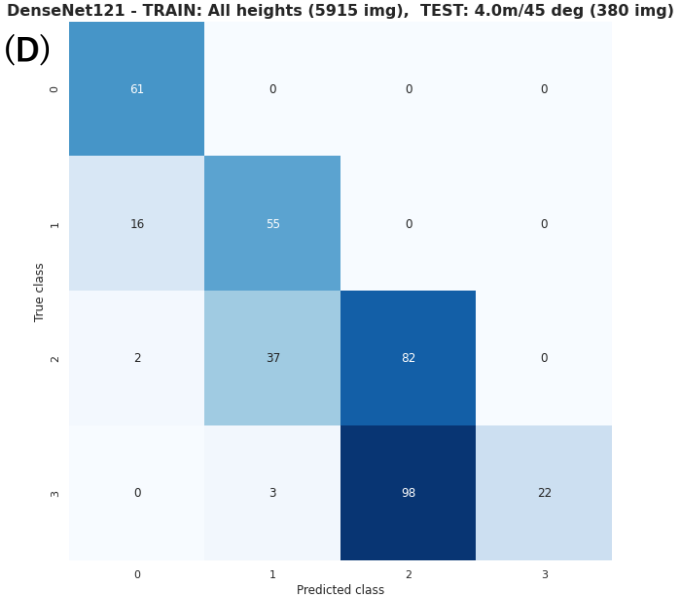


D.5 – Confusion matrices and statistical metrics for impact of environmental and instrumental factors (Train 3: All heights + angles)









	VALIDATION 1 (A) F1-score	TEST 1 (B) F1-score	TEST 2 (C) F1-score	TEST 3 (D) F1-score
Class 0	0.98	0.98	0.97	0.87
Class 1	0.88	0.86	0.87	0.66
Class 2	0.86	0.79	0.82	0.54
Class 3	0.82	0.68	0.75	0.30
Accuracy	0.89	0.87	0.88	0.59

D.6 – YOLOv4 Object Detection correct predictions for one class