UNIVERSITEIT VAN AMSTERDAM

BACHELOR OPLEIDING KUNSTMATIGE INTELLIGENTIE

Detecting Plastic Soup automatically

Using pre-trained Convolutional Neural Networks and Support Vector Machines

Bachelor thesis
Credits: 18EC

University of Amsterdam
Faculty of Science
Science Park 904
1098 XH Amsterdam

Author:
Ysbrand M. A. A. GALAMA
UvAID: 10262067

Supervisor:
Thomas MENSINK

Intelligent Sensory Information Systems
University of Amsterdam

26th June 2015

Abstract

Large amounts of plastic end up in the world's oceans. This project uses state-of-the-art Computer Vision techniques to detect Plastic Soup. Recent studies have shown that Convolutional Neural Networks (CNN) can classify images within the classes the network was trained on. While this project will not train a CNN, a CNN will be used as a feature extractor for a Support Vector Machine (SVM). The goal of this project is to develop a system based on state-of-the-art techniques that is able to distinguish plastic and animals in images. The tests conducted in this project show a high accuracy on the dataset that was constructed for this project, making it possible to detect images containing floating plastic. The resulting pipeline was able to classify images containing plastic and animals. There are even several test resulting in the localisation of Plastic Soup within the images, however more research is needed before this can be used as a real-world application.

Contents

1 Introduction

The environment is constantly being influenced by humanity and this influence has been debated since the hippie movement in the '60s. The debate whether humanity puts too much pressure on the environment has only increased in recent years; the full impact of for example oil-spills and climate change are still being discussed.

This project will focus on an environmental hazards that has recently become apparent: Plastic Soup. However, the project will not focus on how Plastic Soup affects nature, but how Artificial Intelligence can help to clean up the waste.

1.1 Plastic Soup: An environmental problem

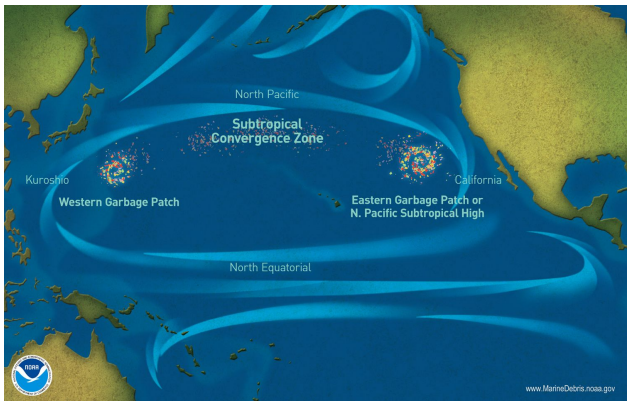
Large amounts of plastic waste end up in the world's oceans and have a significant impact on marine life (Barnes and Milner, 2005). Plastic Soup or Plastic Ocean are both collective nouns for this environmental problem. As Barnes and Milner indicate, the amount of flotsam in the oceans increases annually as does the increasing proportion of plastic.

When plastic is not deposed of correctly, but dumped near rivers, wind and water flows ensure the objects float through rivers into the ocean. The currents in the oceans transport the flotsam around the oceans where it settles in patches. In figure 1.1a the estimated location of the Great Pacific Ocean Patch is shown. Plastic does not decay in nature, therefore a sizeable portion of marine life ingests the plastic after which it ends up in their digestive system. Figure 1.1b shows how much plastic can end up in a bird. In addition to the dangers for marine life, large amounts of plastic end up on beaches, as shown in figure 1.1c. This has a sizeable impact on local society and economy.

The most significant problem is probably micro-plastic. When plastic floats in water for some time, sunlight and minerals break the plastic objects down into grains. These grains of plastic are usually only several micro meters in size and can infect the ecosystem to a large extend (Moore and Phillips, 2011). The full impact of micro-plastic is still unknown, however these plastic grains can end up in bloodstreams of animals and even humans and poison the host (von Moos et al., 2012).

To recapitulate, Plastic Soup can influence the environment to a large extent. The oceans become polluted with plastic and it kills marine life. Moreover, society is also influenced and it is even possible that humans die from ingesting micro-plastic.

The pressure of Plastic Soup on the environment has caused several organisations to form in order to address this problem and find possible solutions for the problem. These solutions range from reducing the dumping of plastic and cleaning up the waste already in the oceans. The nascency



(a) Ocean currents in the Pacific that 'collect' the Plastic Soup. Map by NOAA source: `http://education.nationalgeographic.com/education/encyclopedia/great-pacific-garbage-patch`

(b) The stomach contents of a bird that ate plastic. Photograph by Chris Jordan, U.S. Fish and Wildlife Service

(c) A polluted beach in Mumbai, India. Photograph by EPA

Figure 1.1: Three images that show the impact of Plastic Soup on the environment and society.

of organisations addressing this problem and searching for solutions have gained media attention and thus contribute to the growing awareness for the Plastic Soup. These organisations would benefit of systems that can detect plastic automatically. Therefore, this project will focus on techniques that are able to distinguish plastic from marine life in ocean water. The goal of this project is to develop a system based on state-of-the-art techniques that is able to distinguish plastic and animals in images. The reason to detect plastic is obvious, moreover because an automated system that could clean up the ocean should not interfere with local marine life, this project also focusses on detecting animals.

1.2 Project outline

To develop a system that can detect plastic, state-of-the-art imaging techniques will be used. In recent years the accuracy of imaging techniques on object recognition have steadily increased, as section 2 describes. This project will research how these techniques will perform on Plastic Soup detection.

A dataset of annotated images will be constructed to be used in this project. This dataset can be used to train and test the build system. To recognise the images containing plastic, a Convolutional Neural Network (CNN) will be used. The CNN will not be trained on the data itself but a pre-trained network will be used to construct a feature-vector of the image. These feature-vectors will be used to train a classifier to classify the images in either containing plastic or not containing plastic, or either containing animals or not containing animals. The classifier chosen for this project is a Support Vector Machine (SVM); to distinguish between plastic and marine-life, two SMVs will be trained.

Both CNNs and SVMs are known to reach high accuracy in Computer Vision, therefore it is hypothesised that this method for plastic recognition should give an appropriate baseline.

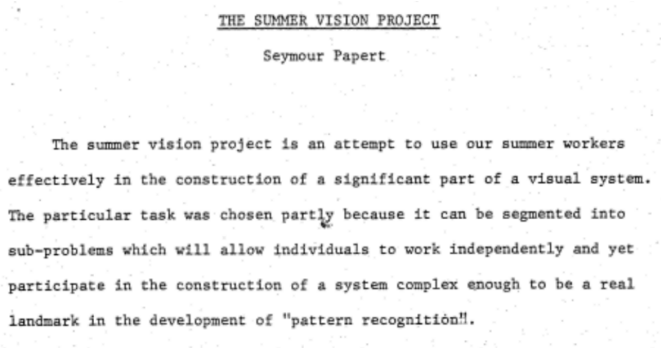
Besides the detection of images containing plastic, this project will also research if the location of the plastic within an image can be detected. As a prove of concept, the pipeline of the plastic recognition will be adapted a small amount to give results for localisation.

1.3 Outline of the thesis chapters

The data that was acquired for this projects is discussed in section 3. In section 4 the method of the project is described and the performance of the plastic-soup detector created in this project is described in section 5. In section 6 I will discuss the parts that could be improved in further research and conclude from the results. But first, in section 2 the current state-of-the-art techniques in the field of Computer Vision will be discussed.

2 Theoretical framework

Image Recognition was originally created to be a summer school of MIT. In 1966 Seymour Papert asked his students over the course of a few months to develop a system that could recognise objects. As was the case in that age, Artificial Intelligence made some wild assumptions on the reachability of their projects (Szeliski, 2010). The recent developments show how mistaken they were, because it took years of study before recent systems arose with accurate vision. Figure 2.2 shows a copy of the original memo Seymour wrote to his students.



THE SUMMER VISION PROJECT

Seymour Papert

The summer vision project is an attempt to use our summer workers effectively in the construction of a significant part of a visual system. The particular task was chosen partly because it can be segmented into sub-problems which will allow individuals to work independently and yet participate in the construction of a system complex enough to be a real landmark in the development of "pattern recognition".

Figure 2.2: The memo Seymote Papert distributed in the summer of 1966 for his Image Recognition project. source: Mensink (2012)

Through the years, multiple algorithms were created to accomplice Computer Vision. The model used to describe images has been a grid with a value (the pixels). This results in a multidimensional function, on which several linear and geometric algebra algorithms can be used. Usually three steps are taken to identify this model of an image. Firstly using the gradient in images, key-point of the image are identified. Thereafter, these key-points are used to construct features or models, describing the image. Finally these models are tested against a classifier (Szeliski, 2010).

2.1 Image representations

As stated above, local image gradients are used to make models on which classifiers can be trained. Several techniques exist that extract features from the image and make a descriptor. A widely used technique for extracting features is a 'Bag of Visual Words'; the frequency of the features result in a multidimensional vector of the size of the number of features (Csurka et al., 2004). From the data-points representing the images, a classifier can be trained.

Another possibility is using a Convolutional Neural Network (CNN). The

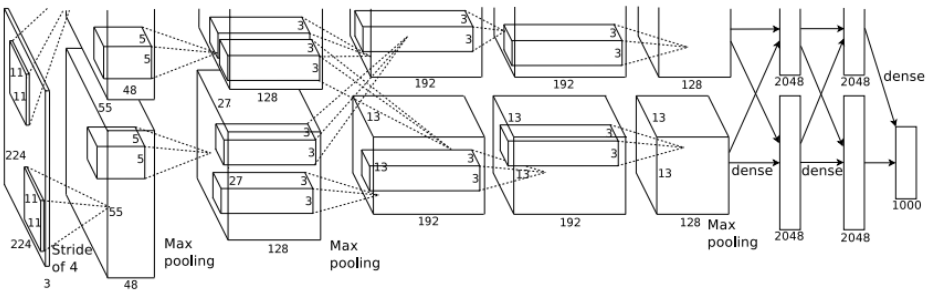


Figure 2.3: Model of the layers of neurons in the CNN as used by Krizhevsky et al. (2012)

CNN does a similar trick with the features as described above. However, it also constructs the features itself from the image-data and has the possibility to be used as a classifier. In other words, a CNN can be used for each of the steps needed for classifying images. The CNN technique is currently popular with image recognition tasks because using large amounts of data, an accurate recognition system can be trained (Girshick et al., 2014), (Razavian et al., 2014).

The biggest difference between 'classical' imaging techniques – such as Bag of Visual Words – and Convolutional Neural Networks is the fact that a CNN is self-learning. The neurons of a CNN are trained on the data and construct their own representation that describes the images, contrary to classical techniques, from which the representations are constructed by humans that build the algorithms.

In Machine Learning, the CNN has existed for quite some time (Fukushima, 1980). However, due to computational complexity is was only recently that Krizhevsky et al. (2012) showed an implementation to compute large CNNs; their study shows how to implement a CNN on a computer's GPU that can work with the massive parallelism necessary to compute many layers of nodes.

Neural Networks consist of nodes, connected with weighted edges. When data is presented to the input of the network, each node calculates with its input an output, which results in an output of the network. When the output of the network is incorrect according to the label of the input, a backward calculation in the network is performed. This calculation makes an adaptation to the weights in such a manner that the network will output the correct label in the next run. A CNN is a neural network that consists of several layers of nodes, with sometimes totalling many millions of neurons. Figure 2.3 shows the layers of neurons the CNN used by Krizhevsky et al. (2012).

Though a CNN can be used to classify a dataset of images, training

such a network uses much computing power. A pre-trained CNN has the weights of the neurons adapted to the dataset used for training. However, because a pre-trained CNN is trained on the classes of the original dataset, the resulting network is not specifically trained on the classes used in this project. Nevertheless, a method that can be applied to work around this problem is inherent to the workings of a CNN. The bottom layers of a CNN trained on large image datasets detect basic visual concepts, for instance lines, corners and simple shapes. Only higher in the network the concepts become more abstract and less local (Zeiler and Fergus, 2014), until the last layer describes the confidence of the trained classes. Therefore, using the second-to-last layer of a pre-trained CNN, an abstract representation of the image can be obtained. This second-to-last layer can be represented as a multidimensional vector in a space the size of the number of abstract features. As stated before, a feature-vector can be used to train a classifier. Effectively, this project uses the second-to-last layer of a pre-trained CNN as a feature extractor of an image to train an SVM classifier.

2.2 Image classification

To classify a feature-vector, several possible algorithms exist. The Support Vector Machine (SVM) is one of the classification algorithms in Machine Learning which is widely used in supervised image classification. The SVM uses the data-points as vectors in a high dimensional space (Boser et al., 1992). In this space, several Linear- and Geometric Algebra techniques are used to fit a hyperplane in this space, where the distance (i.e. a projection of the vector on the plane) of each data-point is maximised. Figure 2.4 shows an example in 2D of possible manners to divide the data-points in the classes; the solid line shows a maximised classifier while the dotted lines are examples of a possible worse fit. The SVM results in a classifier where the sign of the projection vector classifies the data-points (i.e. on which side of the plane the data-point is).

The hyperplane that describes the classifier can be described by, besides linear, other more complex, formulae. These other hyperplanes are sometimes needed to construct a classifier that fits the data. Figure 2.5 shows the three hyperplanes used in this project. The data in these examples show the necessity of the different hyperplanes for different data. However, these more complex planes usually need more calculation time than a linear plane.

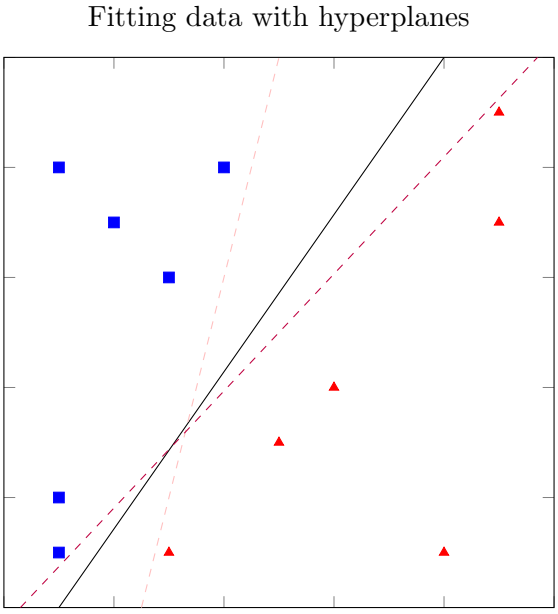


Figure 2.4: Example of possible possible classifications in an algorithm. The dotted lines are a worse fit than the continuous one.

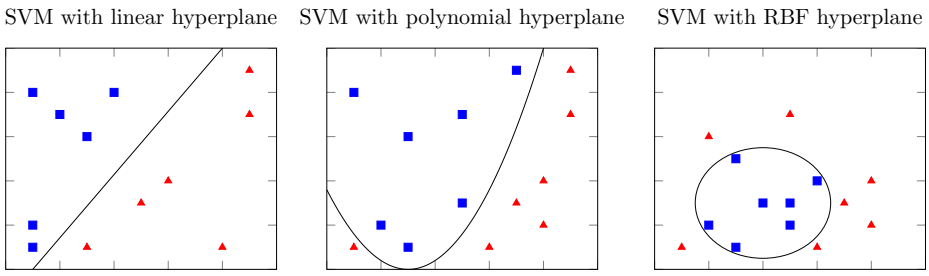


Figure 2.5: Examples of different hyperplanes of a two dimensional SVM fitted on data that need the respective hyperplane for a acceptable fit.

3 Dataset of Plastic Soup images

Because for this project no dataset was available that consists of labelled images of floating plastic, one was constructed by hand.

The dataset used in this project was created from short films by Bill MacDonnald[1]. Several of these clips consist of floating plastic from a viewpoint both above and below water. There were also clips that showed the presence of animals. These clips were segmented in single frames and selected for use in this project; images that did not fall in one of the four classes (plastic, animals, both or none) were excluded. A total of 37.165 images were left to annotate. The annotation was done by my hand, classifying each image on viewpoint and if plastic or animals were visible. To facilitate the annotation, a small Java-application was made. With the use of several key-strokes, large amounts of image-data could be annotated with relative ease. Even so, the annotation of the data took a considerable amount of time.

In total 20.635 images showed plastic only, 6.972 images showed animals only and 8.502 images show both. The above (16.553) and below (20.612) viewpoints were separated into two datasets, to both train and test separately. In the above-set, a total of 14.588 images showed plastic, 6.341 images showed animals and 863 images showed neither. In the below-set, a total of 14.549 images showed plastic, 9.133 images showed animals and 193 images showed neither. Figure 3.6 shows several examples of images in the dataset. Both above and below water viewpoints are shown, as well as the labels the images were annotated with.

Because the images were constructed from films, many images were similar in appearance. This fact should be taken in account when chances of overfitting occur.

To evaluate the results of the pipeline, the dataset was divided in a train (70%), validate (10%) and test (20%) set. Because consecutive frames were very similar, the division was done using a pseudo-random randomise algorithm which used the same seed every time. This resulted in three sets of images that had a high distribution of the original data while consisting of the same images within each run. This dataset made it possible to train the different algorithms used in this project and test the accuracy with the amount of labels computed correctly.

[1]The videos are online on youtube: `www.youtube.com/007bmac`. Thank you Bill for letting me use your films in this research

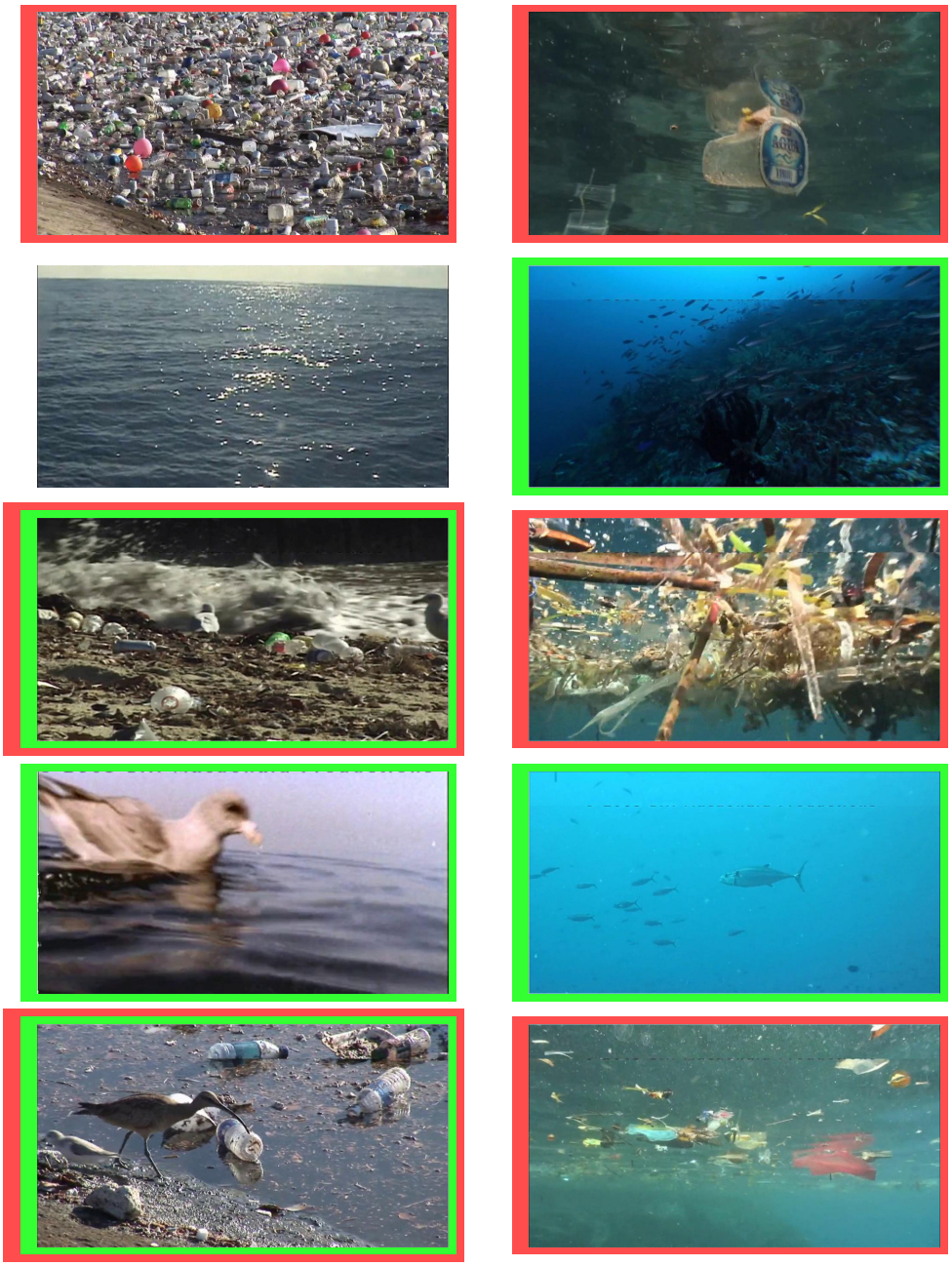


Figure 3.6: Several images from the dataset: Left images are from the above water viewpoint and right images are from below water. A red border indicates an image classified as showing plastic, where a green border indicates an image classified as showing animals.

4 Method

This section describes the steps undertaken in this project. First the pipeline is explained, after which the details of the implementation are stated.

4.1 Pipeline

As stated in section 2, a CNN was used as a feature-vector of the image, after which a Support Vector Machine was used for classification.

Several libraries exist that implement a CNN. In this project, the choice was made to use Caffe (Jia et al., 2014) because this framework is open source and free to use. Moreover, it has a large community maintaining and developing, can run both on CPU and GPU and has an interface to `Python`.

The Caffe framework has several pre-trained CNNs. In this project the Caffenet network was used, which is an implementation of Alexnet (Krizhevsky et al., 2012). Caffenet is trained on the *ilsvrc12* dataset, however in contrary to Alexnet, the order of pooling and normalisation is swapped.

The output of Caffenet is a 1×1.000 probability vector, representing the confidence of the respective class. However, the second-to-last layer of the network is used as feature-vector. This layer is a 10×4.096 vector; 10 rows because the network uses five overlapping 'sub-images' (four corners and one centre) and their mirrored image, to improve accuracy. Each row of 4.096 numbers is an abstract representation of the image, trained by the network. To reduce complexity, the mean over the 'sub-images' is used to compress the second-to-last layer data in a 1×4.096 vector. This vector was then used as a 4.096 dimensional space for the SVM. The three possible hyperplanes as described in section 2.2 were tested using several parameters. Tests were conducted by training the SVM on the test-data, and using the validate-data to find the best parameters. Figure 4.7 shows a visualisation of the pipeline used in this project.

Three different types of hyperplanes were used. A polygonal plane was tested with degrees of 2 till 9. An RBF plane was tested with gammas of 0.0 till 1.0 with steps of 0.1. Lastly a linear plane was tested. Each of the twenty SVM models were trained on part of the train-set, with intervals of order of magnitude; the sizes of the train-set were: 14, 140, 1.400 and 14.000 images.

4.2 Implementation

`Python` was used to write the code necessary for the pipeline. This choice was made not only because Caffe has an interface in `Python`, but also because of the many libraries available in `Python`. The library that allows the use of matrices and basic linear algebra operations is `numpy`. Another library

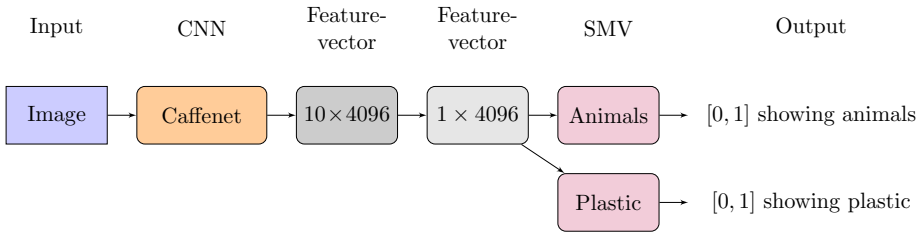


Figure 4.7: Schematics of pipeline used in this project. The image is classified using a CNN, the mean of the second-to-last layer is input for two SVMs which give the output whether detecting plastic or animals.

called `scikit.learn` has an implementation of an SVM, which has been used in this project.

All models were trained and tested on a laptop with 2^{nd} generation i5 CPU, 4GB RAM and without using a GPU. Therefore, the testing and training time of this project should be evaluated relatively to each other. Furthermore, because no GPU was used, running Caffenet took a considerable amount of time. Therefore the output of the network was saved for each image, in order to speed up the training of the SVM.

4.3 Detecting the location of plastic

After testing the pipeline to detect plastic and marine life on images, small adaptations were made in order to find the location of the classes in the images. When the system developed in this project is eventually used for practical applications, finding the exact location of the plastic will be important.

To distinguish blobs in an image, several techniques could be used; however, as a prove of feasibility, no complex algorithms were used in this project. A new piece of code was written that segmented the image in sections and treated each of them as an image in the original pipeline.

The outcome of each segmented piece of the image was cumulated to show the confidence of detection of the classes at the given location in the image. Because there was no labelled data of these segmented images, the evaluation could not be done on a large scale. Therefore no sizeable experiments could be conducted to test the accuracy of locating of the classes. However, in section 5.2 the outcome of several tests is shown.

5 Results

In this section the outcome of the tests is shown. Both the search for optimal parameters of the SVM and the outcome of several tests using the localisation of plastic are described below.

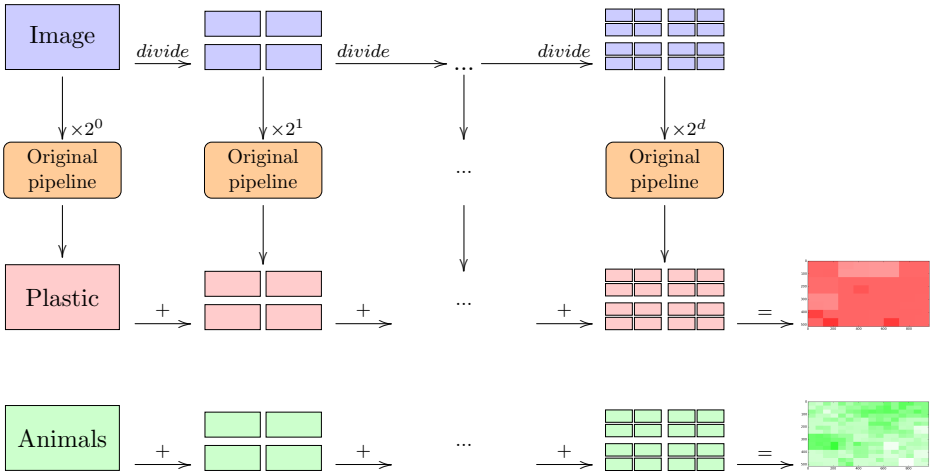


Figure 4.8: Schematics of the second pipeline for localisation. The image is segmented into pieces. Each piece is pulled through the original pipeline, resulting in images with intensity according to the output of the SVM. All the pieces are added together at the pixel-location, which results in an image showing the confidence of the location of plastic or animals

5.1 Pipeline

The outcome of the different parameters the pipeline was tested with can be found in appendix B. Several details of the raw output are visualised and shown in this section. The accuracy shown in this section is calculated by dividing the correct labelled images by the total amount of images. The formula below shows this calculation:

$$\frac{\#(Outcome_{True}\, and\, Label_{True}) + \#(Outcome_{False}\, and\, Label_{False})}{\#tested\, images}$$

The time the training and testing of the SVM took was calculated by using the difference in *System time* from start to finish of a run. Because the data from the CNN was saved on a hard drive and imported in RAM, that time is not included in these tests.

Figure 5.9 shows how each of the SVM models perform and what time it took to train and test. All these tests were fitted on the data of the under-water viewpoint. The blue dots represent the accuracy, which can be read out on the left y-axis. The green and red bars represent time taken to train and test respectively, that can be read out on the right y-axis. Each instance on the x-axis is one of the twenty different hyperplanes. The graphs show how the accuracy increases with larger train-sets. However, the RBF kernel – irrespectively to γ – never reaches a satisfactory accuracy, while needing much time to train. On the other hand, the linear kernel shows a

high accuracy while using a small amount of time to train and test. The polynomial kernel shows a decrease in accuracy with higher degrees.

In figure 5.10 the results of the orders of magnitude of the train size are shown when a linear SVM was used. The figure consists of two graphs, the left one showing the accuracy on each class individually and the total accuracy, the right one showing the time taken to test and train. The x-axis shows the size of the train-set on a logarithmic scale. In other words, the first column of the first four graphs is plotted in a single image, only using a different set of test-data. This figure shows how the accuracy of the individual classes is higher than the combination. It shows an increase in accuracy with larger train-sets while also increasing the time needed to train.

In figure 5.11 a test that accounts for possible overfitting was executed. These tests were trained on one train-set of one of both viewpoints, and tested on the test-set of the other viewpoint. These graphs show a low accuracy for recognising the animal class and therefore a low accuracy on both. The plastic class however does shows a better accuracy between the 70% and 90%.

Figure 5.12 shows a graph of the final model, this is also a linear SVM tested on several orders of magnitude of train data. The tests in this figure were conducted on a combination of both viewpoints. These results show the same trend as the graphs in figure 5.10.

5.2 Detecting the location of plastic

As stated in section 4.3, the localisation of plastic within the image could not be evaluated truly. However, several images were pulled through the pipeline and the results can be seen in figure 5.13. The middle column in the figure shows the tested images. The more red is showing in the figure on the right, the more confidence the program has in detecting plastic on that location in the image. The images on the left represent the same with the greenness for locating animals. Testing the localisation took a considerable amount of time, because a run through the pipeline could take up to 5 seconds per sub-image. Because these tests were conducted on a pyramid-segmentation of depth 4, a total amount of 341 images were run through the pipeline per test.

The results show that the confidence of detecting plastic or animals within the image does not always agree with what is shown in the original image. However, because of the lack of annotation, there is no manner in which these results can evaluated more thoroughly.

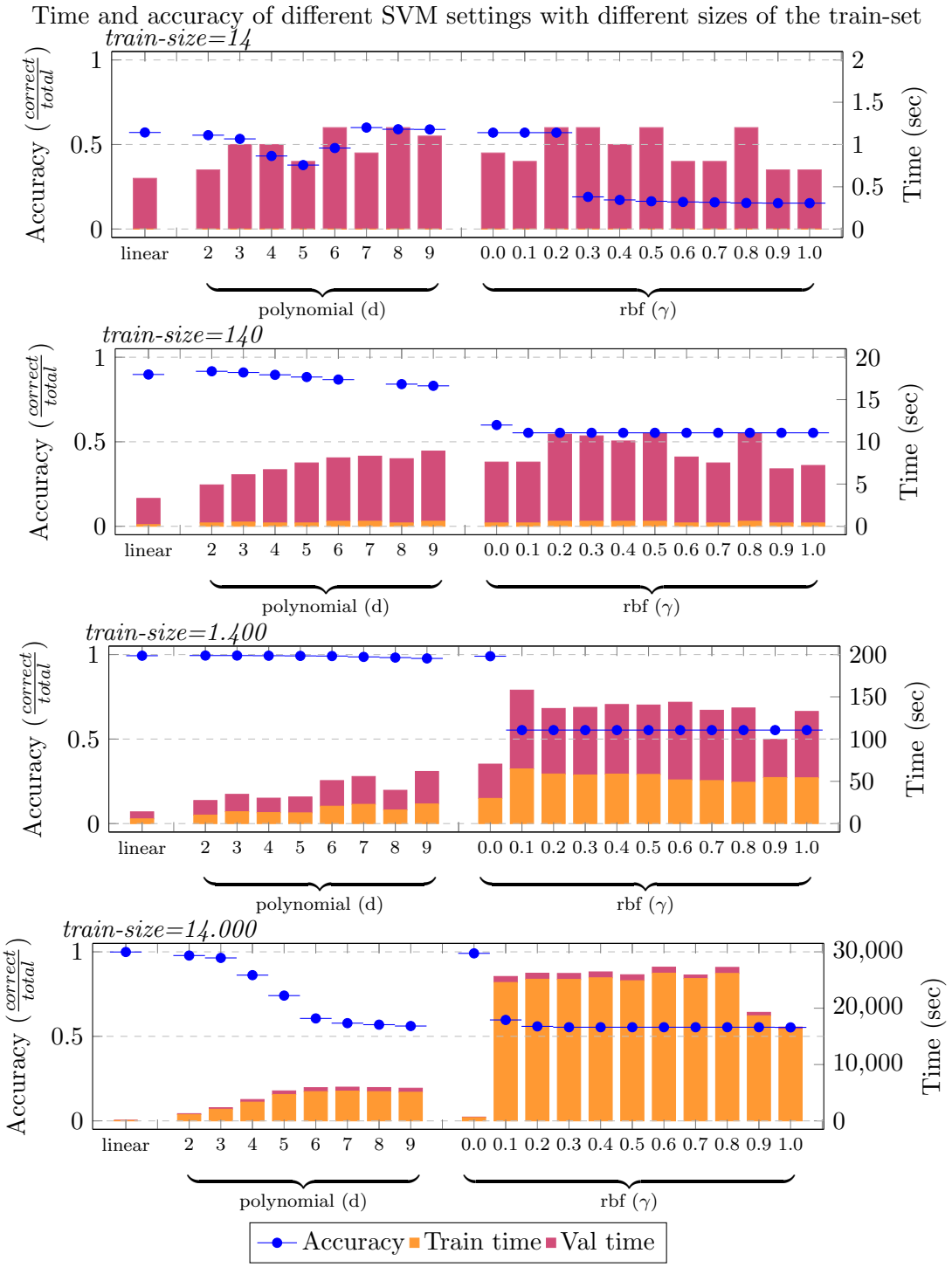


Figure 5.9: Outcome of the different parameters of the SVM trained on different slices of train-data and tested on the validation-set. The linear kernel has one of the highest accuracies, while using the least time to train. Opposing to the trend of the RBF kernel

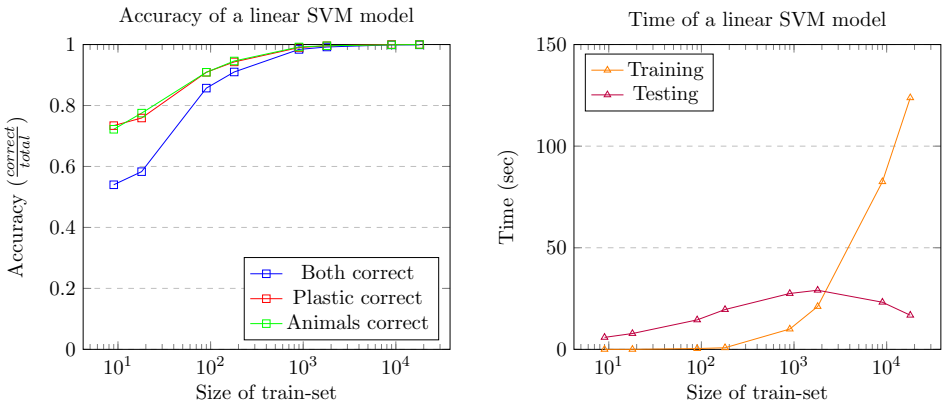


Figure 5.10: Graph of time and accuracy of a linear SVM on different sizes of data from the under water viewpoint. The test-data consisted of 4.123 images. Large datasets show a high accuracy, however even small amounts of data show satisfying results.

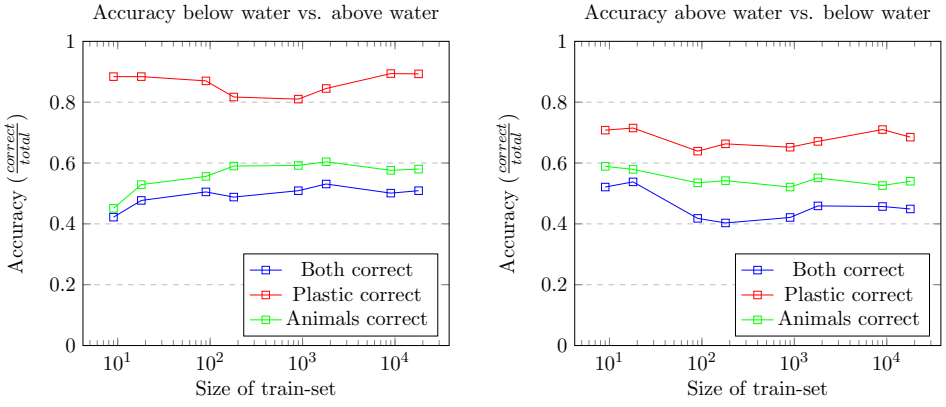


Figure 5.11: Graph of time and accuracy of a linear SVM on different sizes of data. The test set and train set came from the different datasets. The animal class does not gain high accuracy, the plastic class does.

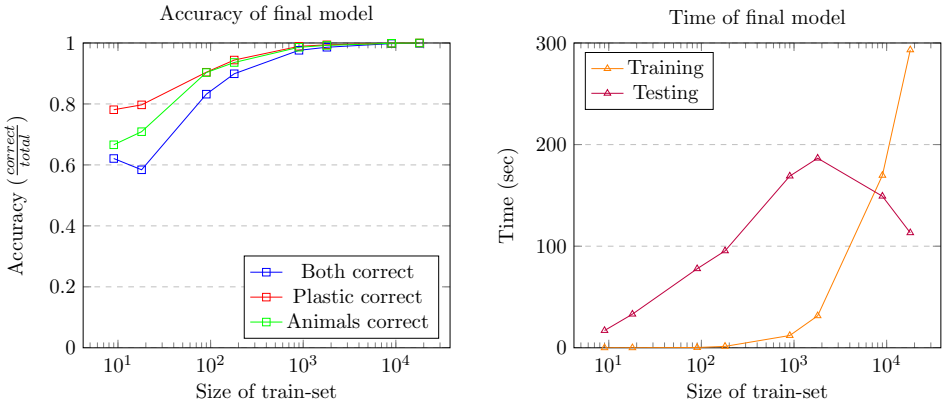


Figure 5.12: Graph of time and accuracy of a linear SVM on different sizes of data from a set of mixed images of the above and below water viewpoint. The test-data consisted of 18.583 images. The same trend as the graphs in figure 5.10 is seen.

6 Discussion

This project tried to contribute on solving the Plastic Soup problem. The amount of floating plastic in the world's oceans could be very dangerous to the environment and society (Moore and Phillips, 2011). Automating the process could help with the clean-up and therefore a system that could detect plastic was constructed in this project. For this purpose a dataset was build, on which a CNN was used for feature extraction. On these features a SVM classifier was trained and tested against the labels of the dataset.

As shown in section 5, the best kernel for an SVM in this dataset is a linear hyperplane. This kernel had a high accuracy, even while being trained on small amounts of data, while not using much time to train or test. The test of figure 5.10 shows an accuracy on plastic and animal detection with an error of less than 1‰. Therefore it can be concluded that this method, using the second-to-last layer of a pre-trained Convolutional Neural Network on an SVM, makes it possible to detect Plastic Soup in images.

A precise conclusion from the localisation experiment is difficult to draw. The conducted test shows some results: several of the images show a higher confidence on locations where the concentration of plastic is higher. However, this is not always the case; especially with the location of marine-life. The current localisation pipeline does not work satisfactory, more tests should be conducted with this technique, with possibly more annotated data, to achieve better results.

The accurate results of the SVM were be expected. SVMs are known to work well in high dimensional data, as long as the amount of classes to

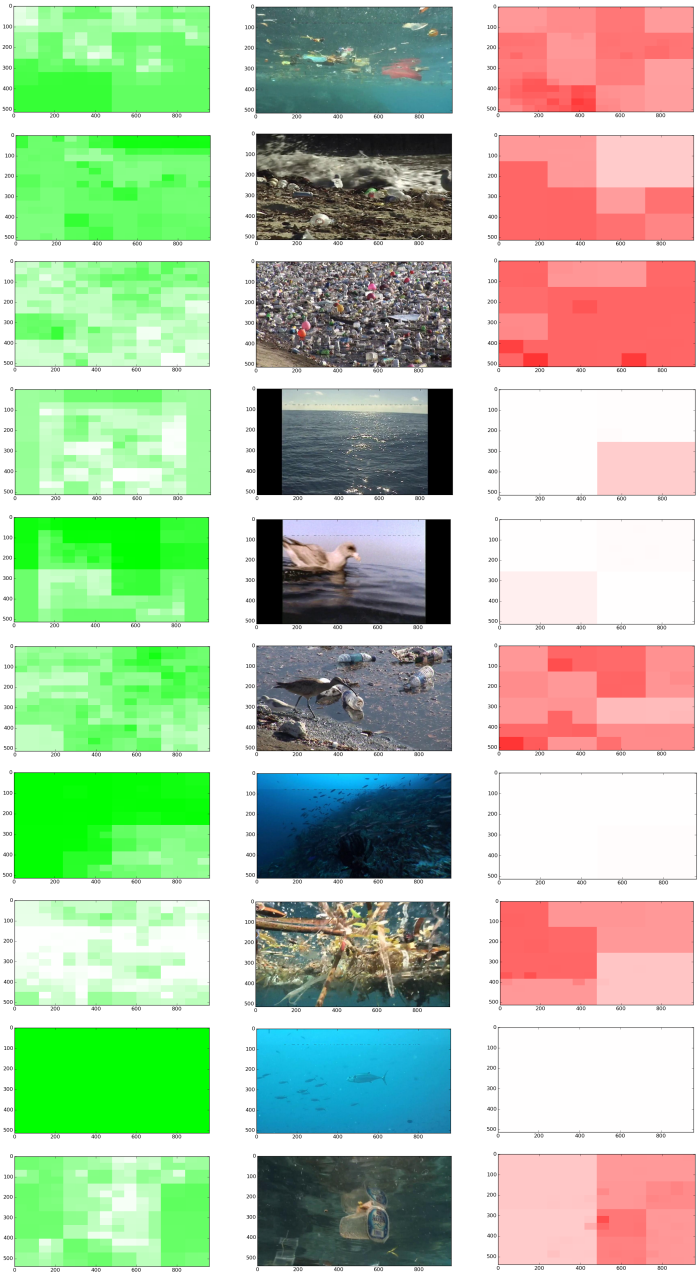


Figure 5.13: Example of outcome of running an image through the segmented image pipeline. Green locates the animals and red the plastic

be trained on is small. The 4.096 dimensional vector of the second-to-last layer of the CNN needed to be trained on two classes; each class had its own SVM trained on either showing the class or not. However, because the model was trained on 70% the images in the dataset, which consisted mostly of consecutive frames of short film clips, there is a substantial amount of overfitting possible. Nevertheless, as the test of figure 5.11 shows, the pipeline shows a satisfactory accuracy when trained on the other viewpoint than tested and therefore refute a high accuracy solely based on over-fitting. The low accuracy of the animal-class can be explained because the system is trained on fish and tested on birds, or vice versa. Therefore a test is done on a mixed set of both viewpoints, which results are shown in figure 5.12. The usage of both the above water and below water viewpoints from the dataset in a single test, increases the amount of variance between the images. The results show the same trend as before: gaining a high accuracy, even even trained on a small part of the dataset. In this case overfitting when half the dataset is used is also probable, nevertheless, the model also shows accuracy of more than 90% when trained on merely a small part of the train-set.

The localisation experiments however had less satisfying results. The confidence of detecting plastic or animals does not conform consequently to what is shown on the image. This could explained by the fact that the system learned to detect plastic from a distance. While training for image classification, many pixels described the existence of plastic. In the localisation experiments however, a smaller amount of pixels was used for classification, making it harder for the pipeline to recognise the plastic on a different scale. Furthermore, the sub-images are more zoomed than a full image. Therefore the plastic has to be recognised in a more closed-up fashion than the system was trained on; instead of detecting much plastic together, single plastic objects now need to be recognized. This could explain why the system had difficulties localising the plastic and animals within the images.

In this project, several aspects could be addressed in order to improve the results. Firstly the used dataset consisted of many similar frames, which increased the chance of overfitting to this particular dataset. Furthermore, the classes that were trained on were not fairly distributed in the dataset. An improved dataset containing more variance of images and fairer distributed classes, could be used to improve the results of this project. This dataset should also contain labels on parts of the images, which could improve the training and testing of the localisation of plastic within images.

This project did not research the usage of different CNNs. A standard CNN was used to construct the feature-vector; no time was spent on researching if other networks would perform better. Not only was assumed that the CNN worked properly and was therefore not further tested, also the parameters with which the SVM was tested were not statistical analysed; no cross-fold validation was conducted. Nevertheless, as the accuracy of the pipeline was close to 100%, no more time was taken to research these

parameters than necessary. The results show nonetheless how this pipeline could be used to detect Plastic Soup from image data.

This project showed that a pre-trained CNN in combination with a linear SVM could detect images containing either of two classes. Further research in this domain could try to improve the system made in this project. An improved dataset with more distributed classes and less similarity between the images in combination with training the data on more than one scale, could increase the performance on localisation of plastic within an image. Projects using this method, should also keep in mind that cross-validation on the hyper-parameters could improve the results if they are not satisfactory.

An other example to improve the localisation of the plastic within the images, are the use of other algorithms. If more complex algorithms are used for object detection in the image, the blobs resulting from these algorithms could be input for the pipeline, instead of the now used segmentation-pyramid. One of the algorithms that could be used for this is BING (Cheng et al., 2014), however this project did not have the time to implement one of these algorithms.

Before the system proposed in this project could be used for real-world applications, more research is needed. However, this project showed the possibility of using state-of-the-art Computer Vision techniques to detect Plastic Soup. More research building on the results of this project could result in applications that can detect Plastic Soup.

This project is one of the many recent projects that show the possibilities of Convolutional Neural Networks. The usability of the promising method of using pre-trained CNNs to train on classes that were not in the original CNN is shown in this project. Using a pre-trained CNN as a feature extractor and training an SVM is a simple technique to gain high accuracy while training on a small dataset. Besides that, it also shows how Artificial Intelligence can be used to help solving environmental problems. I do not think the hippie movement in the '60s could imagine that AI could solve their problems for them today.

References

D. Barnes and P. Milner. Drifting plastic and its consequences for sessile organism dispersal in the atlantic ocean. *Marine Biology*, 146(4):815–825, 2005.

B. E. Boser, I. M. Guyon, and V. N. Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 144–152. ACM, 1992.

M.-M. Cheng, Z. Zhang, W.-Y. Lin, and P. H. S. Torr. BING: Binarized normed gradients for objectness estimation at 300fps. In *IEEE CVPR*, 2014.

G. Csurka, C. Dance, L. Fan, J. Willamowski, and C. Bray. Visual categorization with bags of keypoints. In *Workshop on statistical learning in computer vision, ECCV*, volume 1, pages 1–2. Prague, 2004.

K. Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological cybernetics*, 36(4):193–202, 1980.

R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, pages 580–587. IEEE, 2014.

Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014.

A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

T. Mensink. *Learning Image Classification and Retrieval Models.* PhD thesis, Université de Grenoble, INRIA-Grenoble, and Xerox Research Centre Europe, 2012.

C. C. Moore and C. Phillips. Plastic ocean. *How a sea captain's chance discovery launched a determined quest to save the oceans. Avery, New York*, 2011.

A. S. Razavian, H. Azizpour, J. Sullivan, and S. Carlsson. Cnn features off-the-shelf: an astounding baseline for recognition. In *Computer Vision and Pattern Recognition Workshops (CVPRW), 2014 IEEE Conference on*, pages 512–519. IEEE, 2014.

R. Szeliski. *Computer vision: algorithms and applications*. Springer Science & Business Media, 2010.

N. von Moos, P. Burkhardt-Holm, and A. Koöhler. Uptake and effects of microplastics on cells and tissue of the blue mussel mytilus edulis l. after an experimental exposure. *Environmental science & technology*, 46(20): 11327–11335, 2012.

M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *Computer Vision–ECCV 2014*, pages 818–833. Springer, 2014.

A `Python` code of the project

Code can be found on `github.com/Tomaat/plasticSoup`.

B Output of `Python` code

B.i SVM

abbrivation	meaning
type	which type of model used for the fitting hyperplane; poly mean polynomial, where the last digit stands for the degree; rbf has the last digit stand for gamma
Ttrain	the amount of time in seconds the complete training took
Ttest	the amount of time in seconds the complete testing took
Dsize	the amount of data-points used for training the model
Vsize	the amount of data-points used for testing the model
Bco	the amount of test-points that were correct[2] on both classes
Pco	the amount of test-points that were correct only on the plastic class
Aco	the amount of test-points that were correct only on the animal class
Bac	the accuracy on both classes i.e. $\frac{Bco}{Vsize}$
Pac	the accuracy on the plastic class i.e. $\frac{Bco+Pco}{Vsize}$
Aac	the accuracy on the animals class i.e. $\frac{Bco+Aco}{Vsize}$

B.i–1 Train and validate below water

type	Ttrain	Ttest	Dsize	Vsize	Bco	Pco	Aco	Bfa	Bac	Pac	Aac
poly,1.0,2	0.0	0.7	14	2061	1144	378	453	86	0.555	0.738	0.775
poly,1.0,3	0.0	1.0	14	2061	1099	410	497	55	0.533	0.732	0.774
poly,1.0,4	0.0	1.0	14	2061	890	593	536	42	0.432	0.720	0.692
poly,1.0,5	0.0	0.8	14	2061	779	698	548	36	0.378	0.717	0.644
poly,1.0,6	0.0	1.2	14	2061	988	489	539	45	0.479	0.717	0.741
poly,1.0,7	0.0	0.9	14	2061	1237	235	352	237	0.600	0.714	0.771
poly,1.0,8	0.0	1.2	14	2061	1216	252	204	389	0.590	0.712	0.689
poly,1.0,9	0.0	1.1	14	2061	1214	251	108	488	0.589	0.711	0.641
rbf,1.0,0.0	0.0	0.9	14	2061	1174	271	47	569	0.570	0.701	0.592
rbf,1.0,0.1	0.0	0.8	14	2061	1174	271	47	569	0.570	0.701	0.592
rbf,1.0,0.2	0.0	1.2	14	2061	1174	271	47	569	0.570	0.701	0.592
rbf,1.0,0.3	0.0	1.2	14	2061	392	1053	594	22	0.190	0.701	0.478
rbf,1.0,0.4	0.0	1.0	14	2061	354	1091	594	22	0.172	0.701	0.460
rbf,1.0,0.5	0.0	1.2	14	2061	338	1107	594	22	0.164	0.701	0.452
rbf,1.0,0.6	0.0	0.8	14	2061	330	1115	594	22	0.160	0.701	0.448
rbf,1.0,0.7	0.0	0.8	14	2061	325	1120	594	22	0.158	0.701	0.446
rbf,1.0,0.8	0.0	1.2	14	2061	317	1128	594	22	0.154	0.701	0.442
rbf,1.0,0.9	0.0	0.7	14	2061	316	1129	594	22	0.153	0.701	0.442
rbf,1.0,1.0	0.0	0.7	14	2061	316	1129	594	22	0.153	0.701	0.442
linear,1.0	0.0	0.6	14	2061	1177	350	401	133	0.571	0.741	0.766
type	Ttrain	Ttest	Dsize	Vsize	Bco	Pco	Aco	Bfa	Bac	Pac	Aac
poly,1.0,2	0.4	4.5	140	2061	1890	71	45	55	0.917	0.951	0.939

poly,1.0,3	0.5	5.6	140	2061	1876	77	50	58	0.910	0.948	0.934
poly,1.0,4	0.4	6.3	140	2061	1847	91	56	67	0.896	0.940	0.923
poly,1.0,5	0.4	7.1	140	2061	1820	105	68	68	0.883	0.934	0.916
poly,1.0,6	0.6	7.5	140	2061	1789	114	76	82	0.868	0.923	0.905
poly,1.0,7	0.6	7.7	140	2061	1759	123	85	94	0.853	0.913	0.895
poly,1.0,8	0.4	7.6	140	2061	1733	133	78	117	0.841	0.905	0.879
poly,1.0,9	0.6	8.3	140	2061	1713	143	75	130	0.831	0.901	0.868
rbf,1.0,0.0	0.4	7.2	140	2061	1235	253	89	484	0.599	0.722	0.642
rbf,1.0,0.1	0.4	7.2	140	2061	1140	305	23	593	0.553	0.701	0.564
rbf,1.0,0.2	0.6	10.3	140	2061	1140	305	23	593	0.553	0.701	0.564
rbf,1.0,0.3	0.6	10.1	140	2061	1140	305	22	594	0.553	0.701	0.564
rbf,1.0,0.4	0.6	9.5	140	2061	1140	305	22	594	0.553	0.701	0.564
rbf,1.0,0.5	0.6	10.4	140	2061	1140	305	22	594	0.553	0.701	0.564
rbf,1.0,0.6	0.4	7.8	140	2061	1140	305	22	594	0.553	0.701	0.564
rbf,1.0,0.7	0.4	7.1	140	2061	1140	305	22	594	0.553	0.701	0.564
rbf,1.0,0.8	0.6	10.4	140	2061	1140	305	22	594	0.553	0.701	0.564
rbf,1.0,0.9	0.4	6.4	140	2061	1140	305	22	594	0.553	0.701	0.564
rbf,1.0,1.0	0.4	6.8	140	2061	1140	305	22	594	0.553	0.701	0.564
linear,1.0	0.2	3.1	140	2061	1851	101	54	55	0.898	0.947	0.924
type	Ttrain	Ttest	Dsize	Vsize	Bco	Pco	Aco	Bfa	Bac	Pac	Aac
poly,1.0,2	10.1	17.2	1400	2061	2048	8	5	0	0.994	0.998	0.996
poly,1.0,3	14.1	20.5	1400	2061	2049	7	5	0	0.994	0.998	0.997
poly,1.0,4	13.0	17.1	1400	2061	2047	9	5	0	0.993	0.998	0.996
poly,1.0,5	12.7	18.8	1400	2061	2044	11	6	0	0.992	0.997	0.995
poly,1.0,6	20.6	30.3	1400	2061	2042	13	6	0	0.991	0.997	0.994
poly,1.0,7	22.8	32.9	1400	2061	2032	18	11	0	0.986	0.995	0.991
poly,1.0,8	16.1	23.3	1400	2061	2024	21	11	5	0.982	0.992	0.987
poly,1.0,9	23.2	38.5	1400	2061	2013	23	15	10	0.977	0.988	0.984
rbf,1.0,0.0	29.8	40.6	1400	2061	2040	10	11	0	0.990	0.995	0.995
rbf,1.0,0.1	64.7	93.2	1400	2061	1140	305	38	578	0.553	0.701	0.572
rbf,1.0,0.2	58.7	77.5	1400	2061	1140	305	24	592	0.553	0.701	0.565
rbf,1.0,0.3	57.6	79.9	1400	2061	1140	305	22	594	0.553	0.701	0.564
rbf,1.0,0.4	58.6	82.3	1400	2061	1140	305	22	594	0.553	0.701	0.564
rbf,1.0,0.5	58.2	82.1	1400	2061	1140	305	22	594	0.553	0.701	0.564
rbf,1.0,0.6	51.7	91.9	1400	2061	1140	305	22	594	0.553	0.701	0.564
rbf,1.0,0.7	50.9	83.2	1400	2061	1140	305	22	594	0.553	0.701	0.564
rbf,1.0,0.8	48.9	87.9	1400	2061	1140	305	22	594	0.553	0.701	0.564
rbf,1.0,0.9	54.5	45.1	1400	2061	1140	305	22	594	0.553	0.701	0.564
rbf,1.0,1.0	54.2	78.5	1400	2061	1140	305	22	594	0.553	0.701	0.564
linear,1.0	5.6	8.3	1400	2061	2047	6	8	0	0.993	0.996	0.997
type	Ttrain	Ttest	Dsize	Vsize	Bco	Pco	Aco	Bfa	Bac	Pac	Aac
poly,1.0,2	1124.4	155.7	14000	2061	2015	24	22	0	0.978	0.989	0.988
poly,1.0,3	2068.0	277.0	14000	2061	1987	47	23	4	0.964	0.987	0.975
poly,1.0,4	3331.6	459.8	14000	2061	1777	162	84	38	0.862	0.941	0.903
poly,1.0,5	4678.5	646.8	14000	2061	1527	231	87	216	0.741	0.853	0.783
poly,1.0,6	5198.9	696.6	14000	2061	1249	278	55	479	0.606	0.741	0.633
poly,1.0,7	5282.2	706.6	14000	2061	1192	305	22	542	0.578	0.726	0.589
poly,1.0,8	5193.3	719.4	14000	2061	1173	305	24	559	0.569	0.717	0.581
poly,1.0,9	5083.9	714.2	14000	2061	1157	305	22	577	0.561	0.709	0.572

rbf,1.0,0.0	586.3	86.2	14000	2061	2043	6	12	0	0.991	0.994	0.997
rbf,1.0,0.1	24560.3	1067.4	14000	2061	1230	285	130	416	0.597	0.735	0.660
rbf,1.0,0.2	25139.6	1069.5	14000	2061	1152	304	57	548	0.559	0.706	0.587
rbf,1.0,0.3	25104.7	1059.1	14000	2061	1142	305	31	583	0.554	0.702	0.569
rbf,1.0,0.4	25389.9	1052.6	14000	2061	1142	305	26	588	0.554	0.702	0.567
rbf,1.0,0.5	24877.4	1055.4	14000	2061	1142	305	21	593	0.554	0.702	0.564
rbf,1.0,0.6	26216.1	1062.9	14000	2061	1142	305	20	594	0.554	0.702	0.564
rbf,1.0,0.7	25266.2	618.3	14000	2061	1142	305	20	594	0.554	0.702	0.564
rbf,1.0,0.8	26166.7	1072.2	14000	2061	1141	305	21	594	0.554	0.702	0.564
rbf,1.0,0.9	18654.6	607.0	14000	2061	1141	305	21	594	0.554	0.702	0.564
rbf,1.0,1.0	16250.9	399.1	14000	2061	1140	305	22	594	0.553	0.701	0.564
linear,1.0	74.6	8.4	14000	2061	2058	2	1	0	0.999	1.000	0.999

B.i–2 Train and test below water

type	Ttrain	Ttest	Dsize	Vsize	Bco	Pco	Aco	Bfa	Bac	Pac	Aac
linear,1.0	0.0	5.9	9	4123	2228	797	748	350	0.540	0.734	0.722
linear,1.0	0.0	7.8	18	4123	2402	726	795	200	0.583	0.759	0.775
linear,1.0	0.4	14.5	90	4123	3534	215	213	161	0.857	0.909	0.909
linear,1.0	0.8	19.6	180	4123	3753	136	148	86	0.910	0.943	0.946
linear,1.0	10.0	27.5	900	4123	4057	25	32	9	0.984	0.990	0.992
linear,1.0	21.1	29.1	1800	4123	4092	13	15	3	0.992	0.996	0.996
linear,1.0	82.5	23.2	9000	4123	4118	3	2	0	0.999	1.000	0.999
linear,1.0	123.8	16.8	18000	4123	4119	2	2	0	0.999	1.000	1.000

B.i–3 Train on below water, test on above water

type	Ttrain	Ttest	Dsize	Vsize	Bco	Pco	Aco	Bfa	Bac	Pac	Aac
linear,1.0	0.0	4.4	9	3311	1396	1530	97	288	0.422	0.884	0.451
linear,1.0	0.0	6.8	18	3311	1578	1348	173	212	0.477	0.884	0.529
linear,1.0	0.4	12.6	90	3311	1673	1208	167	263	0.505	0.870	0.556
linear,1.0	0.6	15.5	180	3311	1615	1090	339	267	0.488	0.817	0.590
linear,1.0	9.8	21.7	900	3311	1684	999	277	351	0.509	0.810	0.592
linear,1.0	20.7	23.1	1800	3311	1757	1042	242	270	0.531	0.845	0.604
linear,1.0	79.0	18.7	9000	3311	1659	1301	248	103	0.501	0.894	0.576
linear,1.0	118.4	13.5	18000	3311	1685	1272	237	117	0.509	0.893	0.580

B.i–4 Train on above water, test on below water

type	Ttrain	Ttest	Dsize	Vsize	Bco	Pco	Aco	Bfa	Bac	Pac	Aac
linear,1.0	0.0	5.9	9	4123	2147	771	282	923	0.521	0.708	0.589
linear,1.0	0.0	7.6	18	4123	2218	731	171	1003	0.538	0.715	0.579
linear,1.0	0.4	14.5	90	4123	1724	909	482	1008	0.418	0.639	0.535
linear,1.0	0.6	17.0	180	4123	1660	1073	573	817	0.403	0.663	0.542
linear,1.0	6.6	20.6	900	4123	1735	954	413	1021	0.421	0.652	0.521
linear,1.0	13.5	18.2	1800	4123	1894	872	378	979	0.459	0.671	0.551
linear,1.0	46.2	12.4	9000	4123	1885	1041	284	913	0.457	0.710	0.526
linear,1.0	55.9	9.9	18000	4123	1851	975	375	922	0.449	0.685	0.540

B.i–5 Train and test mixed dataset

type	Ttrain	Ttest	Dsize	Vsize	Bco	Pco	Aco	Bfa	Bac	Pac	Aac
linear,1.0	0.0	17.0	9	18583	11538	2972	846	3227	0.621	0.781	0.666
linear,1.0	0.0	33.1	18	18583	10853	3951	2316	1463	0.584	0.797	0.709
linear,1.0	0.3	77.8	90	18583	15465	1336	1308	474	0.832	0.904	0.903
linear,1.0	1.4	95.4	180	18583	16713	827	686	357	0.899	0.944	0.936
linear,1.0	12.1	169.0	900	18583	18134	243	196	10	0.976	0.989	0.986
linear,1.0	31.5	186.6	1800	18583	18325	139	112	7	0.986	0.994	0.992
linear,1.0	169.7	149.2	9000	18583	18544	24	14	1	0.998	0.999	0.999
linear,1.0	293.2	113.2	18000	18583	18568	7	7	1	0.999	1.000	1.000