UNIVERSITEIT VAN AMSTERDAM

BACHELOR OPLEIDING KUNSTMATIGE INTELLIGENTIE

Detecting Plastic Soup automatically

Using pre-trained Convolutional Neural Networks

Bachelor thesis
Credits: 18EC

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

Author:
Ysbrand M. A. A. GALAMA
10262067

Supervisor:
Thomas MENSINK

Intelligent System Lab
Amsterdam
Faculty of Science
University of Amsterdam
Science Park 904
1098 XH Amsterdam

22nd June 2015

Contents

Abstract

This is the abstract... Lorem ipsum dolor sit amet, consectetur adipiscing elit. Mauris placerat erat vitae semper fringilla. Integer rhoncus sed libero quis hendrerit. Duis et nunc convallis, porttitor urna quis, ultrices nunc. Ut quis ornare neque. Vivamus rutrum nunc arcu, ac pretium nulla lobortis ac. Duis a lectus ex. Fusce eget nulla at metus commodo posuere. Proin et velit sed augue interdum lobortis. Mauris ut nisi suscipit purus dictum rutrum vitae non erat. Nulla non semper turpis, nec eleifend mi. Proin ut bibendum est, quis auctor ante.

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 on the weight humanity has on the environment has also been fuelled in recent years; the full impact of oil-spills and climate change are still being debated about. [citation needed?]

This project will focus on one of the environmental hazards: Plastic Soup. However, the project will not focus on the problem, or how Plastic Soup affects nature, but how Artificial Intelligence can help to clean up the waste.

1.1 Plastic Soup: 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 a collective noun for this environmental problem. As Barnes and Milner indicate, the amount of flotsam in the oceans grows annually as does the increasing proportion of plastic.

When plastic objects are disposed of in rivers, they float through rivers into the ocean. The currents in the oceans transport the flotsam around the world to influence marine life. In figure 1.1 the probable location of the Great Pacific Ocean Patch is shown[1]. Because the plastic does not decay in nature, a lot of marine life ingests the plastic into their digestive system. Figure 1.2a 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 1.2b. This has a sizeable impact on the

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 several micro meters in size and can infect the ecosystem to a large extend Moore and Phillips (2011).

The weight of Plastic Soup on the environment has caused several organisations to form in order to address this problem and find solutions to the problem. The nascency of organisations addressing the problem and searching for solutions has gained media attention and thus cause the growing awareness for the Plastic Soup.

These organisations would benefit with 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.

[1] source: `http://education.nationalgeographic.com/education/encyclopedia/great-pacific-garbage-patch`

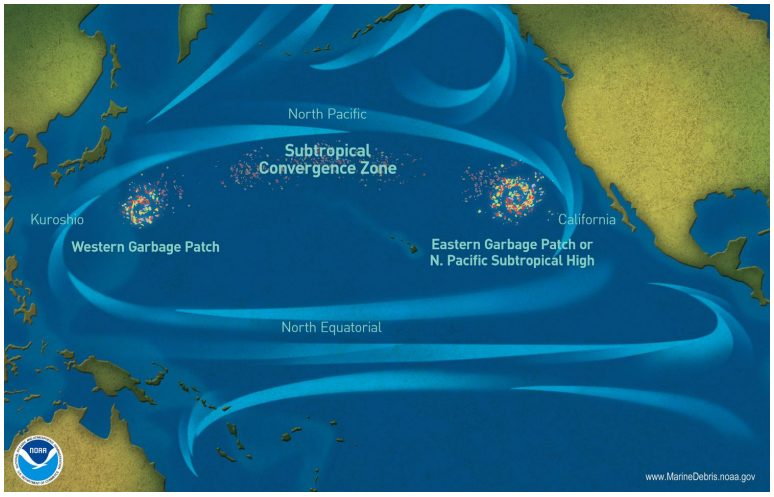

Figure 1.1: Ocean currents in the Pacific that 'collect' the Plastic Soup. Map by NOAA

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

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

Figure 1.2: Impact of Plastic Soup on the environment and society

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 been growing, as section **??** describes. This project will research how these techniques will perform on Plastic Soup detection.

A dataset of annotated images has been constructed to be used in this project. This dataset can be used to train and test the algorithm. To

recognise the plastic containing images, a Convolutional Neural Network (CNN) will be used. The CNN will not be trained on the data itself, however 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 ether containing or not containing plastic. The classifier used in this project is a Support Vector Machine (SVM); to distinguish between plastic and marine-life, two SMVs will be trained.

This method for image recognition should give an appropriate baseline. TODO: uitbreiden

1.3 Outline of the thesis chapters

The performance of the plastic-soup detector made in this project are described in section 6 from the results in section 5. In section 4 the method of the project is shown, and in section 7 I will go into the parts that can be improved in further research. But first, in section 2 the current state-of-the-art techniques in the field of Computer Vision.

2 Theoretical framework

The history of Image Recognition begins as a summer school of Stanford. As was the case in that age, Artificial Intelligence made some wild assumptions on the reachability of their projects. [citation needed] The recent developments show how mistaken they were.

Through the years multiple algorithms have been made 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 on. Usually three steps are taken to identify an image. First using the gradient in images, key-points 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. [citation needed]

2.1 Image representations

As stated above, local image gradients are used to make models on which classifiers can be trained. Several techniques exists that extract features from the image and make a descriptor. A wildly used technique for extracting the features is a 'Bag of Features'; the frequency of the features result in a multidimensional vector of the size of the number of features. From the data-points representing the images, a classifier can be trained. [citation needed]

Another possibility is using a Convolutional Neural Network. The CNN does a similar trick with the features. 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. From 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 Features– and Convolutional Neural Networks is the self-learning of the CNN. The neurons of a CNN are trained on the data and construct their own representation that describes the images. In contrary to classical techniques, from which the representations are constructed by humans that build the algorithms.

In the Machine Learning the Convolutional Neural Network (CNN) has existed for quite some time (**?**). However, due to computational complexity is was only recently that Krizhevsky et al. (2012) showed an implementation to compute large CNNs. Krizhevsky et al. 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 many millions of neurons. Figure 2.3 shows the layers of neurons the CNN used by Krizhevsky et al..

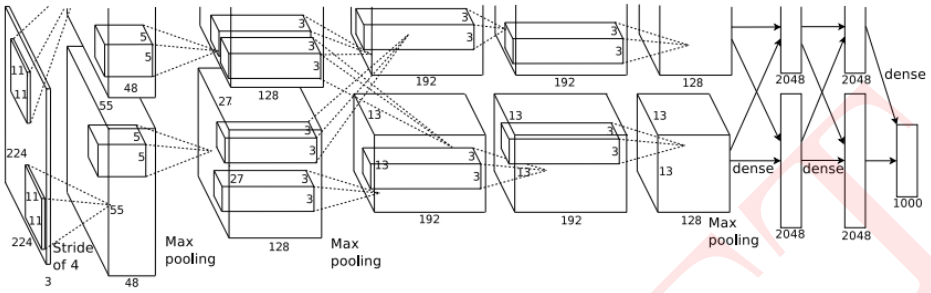

Figure 2.3: Model of the layers of neurons in the CNN as used by Krizhevsky et al. (2012)

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. This results in a network trained on other classes than used for this project. However, 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). 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 a pre-trained CNN as a feature extractor of an image to train an SVM classifier.

2.2 Image classification

Classifying feature-vectors has also several possible algorithms. The Support Vector Machine (SVM) is one of the classification algorithms in Machine

Learning that is wildly used in supervised image classification.

The SVM uses the data-points as vectors in a high dimensional space. In this space, several Linear 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. 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).

Fitting data with hyperplanes

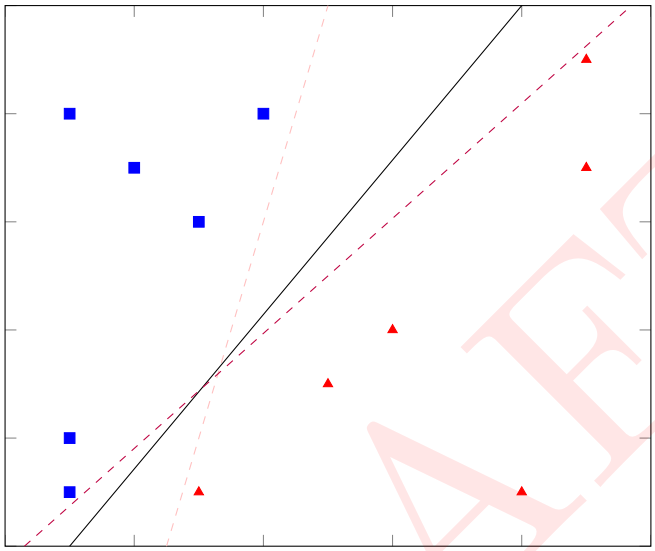

Figure 2.4: Example of possible possible classifications in an algorithm. The dotted lines are a worse fit than the continuous one.

The hyperplane that describes the classifier can be described by, besides linear, other, more complex, formulae. However, these planes usually need more calculation time than a linear plane. 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.

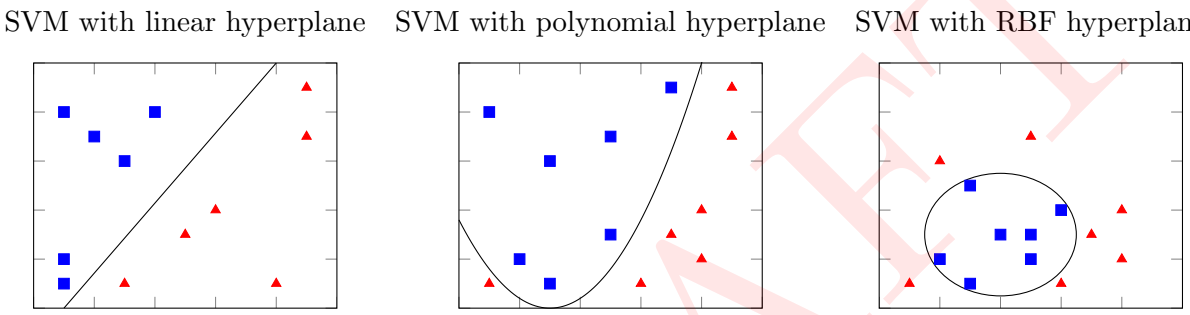

Figure 2.5: Examples of different hyperplanes of a two dimensional SVM

3 Dataset of Plastic Soup images

Because no dataset was available for this project, one was constructed by hand.

The dataset used in this project is made from short films by Bill MacDonnald[2]. Several of these clips consist of floating plastic from a viewpoint both above and below water. These clips have been segmented in single frames and selected for use in this project. A total of 37165 images were left to annotate. The annotation has been 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 20635 images show plastic only, 6972 images show animals only and 8502 images show both. The above (16553) and below (20612) viewpoints were separated in two datasets, to train and test on both separately. In the above-set a total of 14588 images show plastic, 6341 images show animals and 863 images show none of both. In the below-set a total of 14549 images show plastic, 9133 images show animals and 193 images show none of both. 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 are constructed from films, many images are similar in appearance. This fact should be taken in account when chances of overtfitting occurs.

To evaluate the results of the pipeline, the dataset is divided in a train (70%), validate (10%) and test (20%) set. Because consecutive frames are similar, the division is done by using a pseudo-random randomise algorithm with each time the same seed. Resulting in three sets of images that have a high distribution of the original data while remaining the same on each run. The results of the algorithms can therefore be tested on accuracy with the amount of labels computed correctly.

[2]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 taken 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 Convolutional Neural Network has been 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 is made to use Caffe (Jia et al., 2014). 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 is 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. [citation needed]

The output of Caffenet is a 1×1000 normalised 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 10×4096 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 4096 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 1×4096 vector.

This 4096 dimensional vector is then used as the space for the SVM. The three possible hyperplanes as described in section 2.2 were tested with several parameters. Tests were conducted by training the SVM on the test-data, and using the validate-data to find the best parameters.

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, 1400 and 14000 images.

4.2 Impelemtation

`Python`has been used to write the code necessary for the pipeline. 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 called `scikit.learn` has an implementation of an SVM, which has been used in this project. The code used in this project can be found in appendix A

All models are trained and tested on a laptop with 2^{nd} generation i5 CPU, 4GB RAM and without using a GPU. Therefore time testing and training took in 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 had been saved for each image, to speed up the training of the SVM.

4.3 Detecting location of plastic

After testing the pipeline to detect plastic and marine life on images, small adaptations were made to find the location of the classes in the image. When the system developed in this project is used for practical applications, location of the plastic is 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 segments the image in sections and treats each of them as an image in the original pipeline.

The outcome of each segmented piece of the image are cumulated to show the confidence of detection of the classes at the given location in the image.

Because there is not 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

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 labled images by the total amount of images. Or in formula form:

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

The time that has been taken into account is the difference in *System time* from beginning the training or testing and finishing. Because the data from the CNN was saved on a hard drive, that time is not included in these figures.

Figure 5.7 shows how each of the SVM model performes 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, that 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.

In figure 5.8 the results of the orders of magnitude of the train size are shown when a linear SVM was used. The figure consists of two graps, the left one shows the accuracy on each class individually and the total accuracy, the right one shows 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.

In figure 5.9 there is moreover a graph of 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.

5.2 Detecting location of plastic

As stated in section **??** the localisation of the plastic within the image could not be evaluated truly. However, several images have been pulled through the pipeline and the results can be seen in figure 5.10. The middle column in the figure shows the tested images. The more red in the figure on the right, the more confidence the program has in detecting image on that location. The images on the left represent the same with the greenness for locating animals.

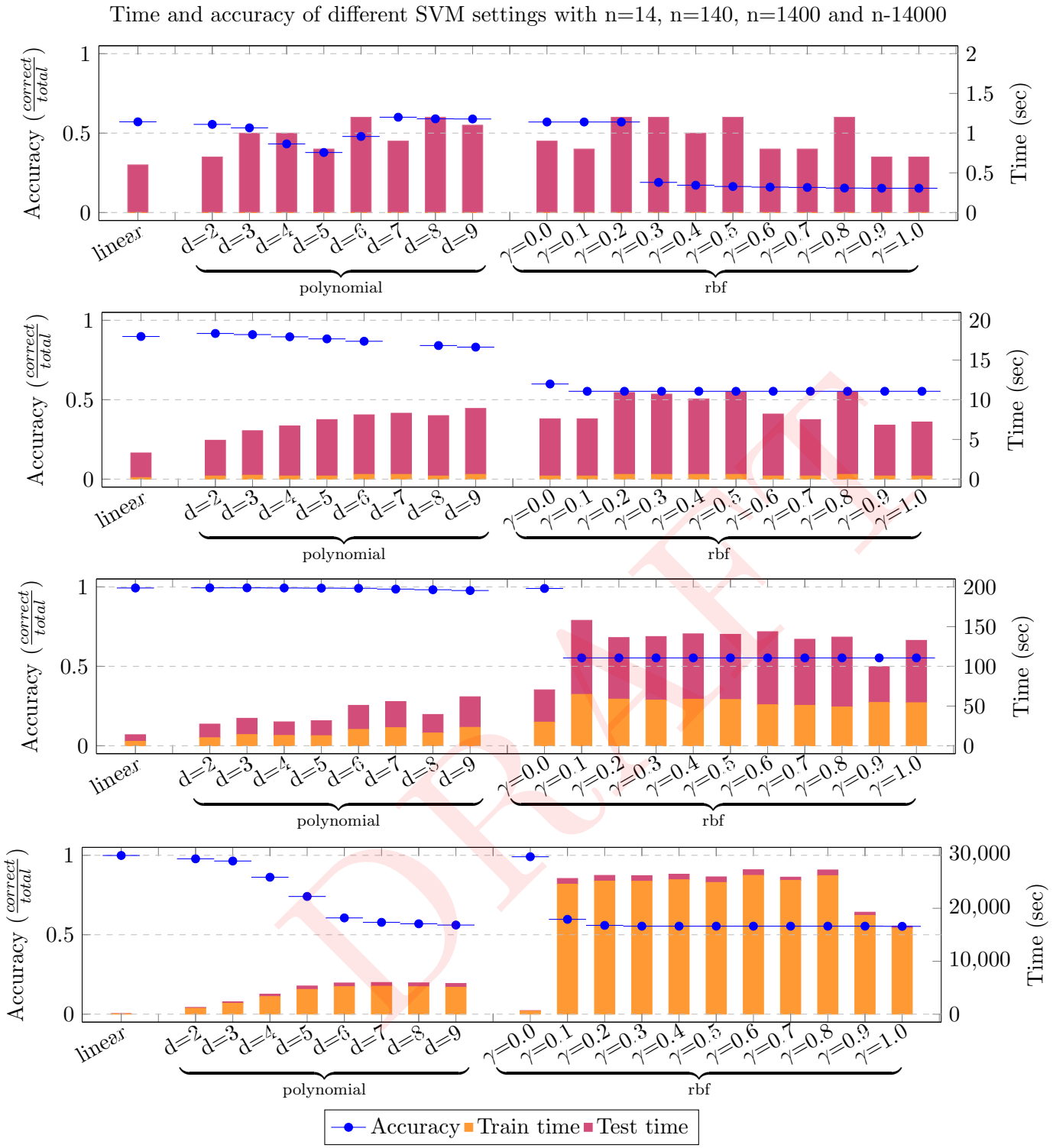

Figure 5.7: Graph of the different parameters of the SVM

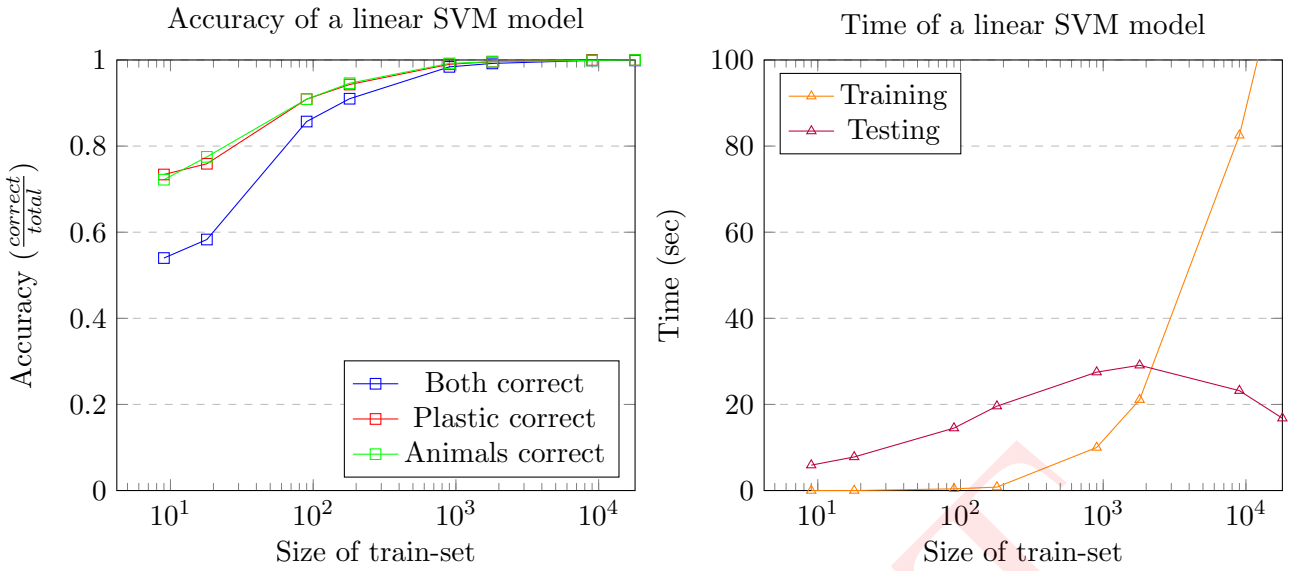

Figure 5.8: Graph of time and accuracy of a linear SVM on different sizes data of the under water viewpoint. The test-data consisted of 2061 images.

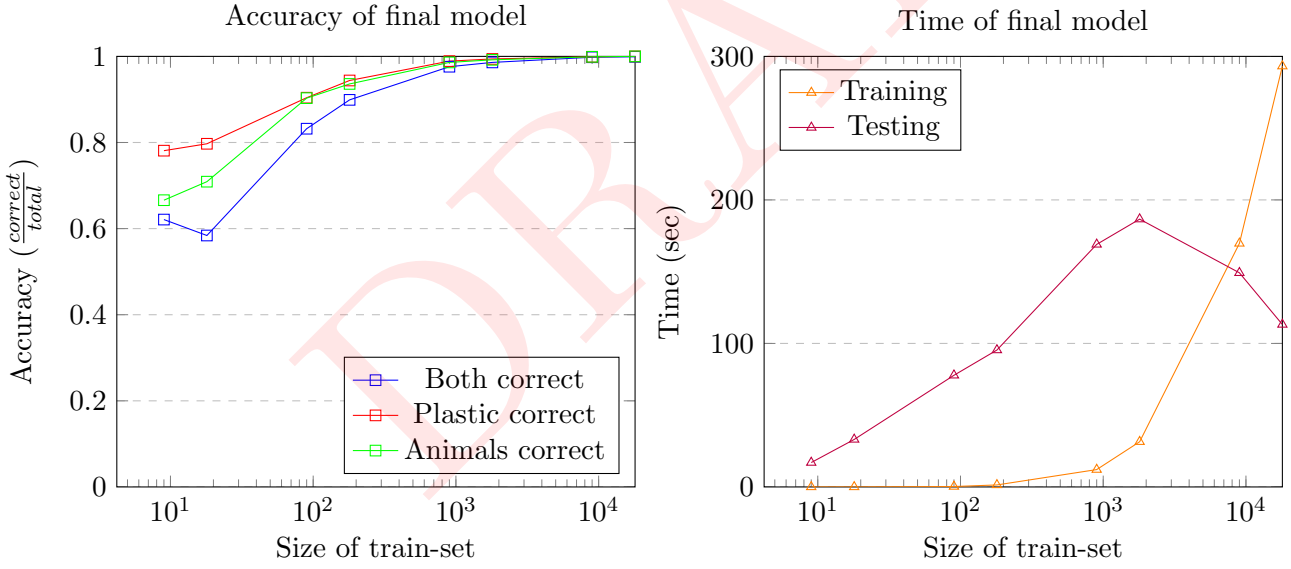

Figure 5.9: Graph of time and accuracy of a linear SVM on different sizes data of the under water viewpoint. The test-data consisted of 18583 images.

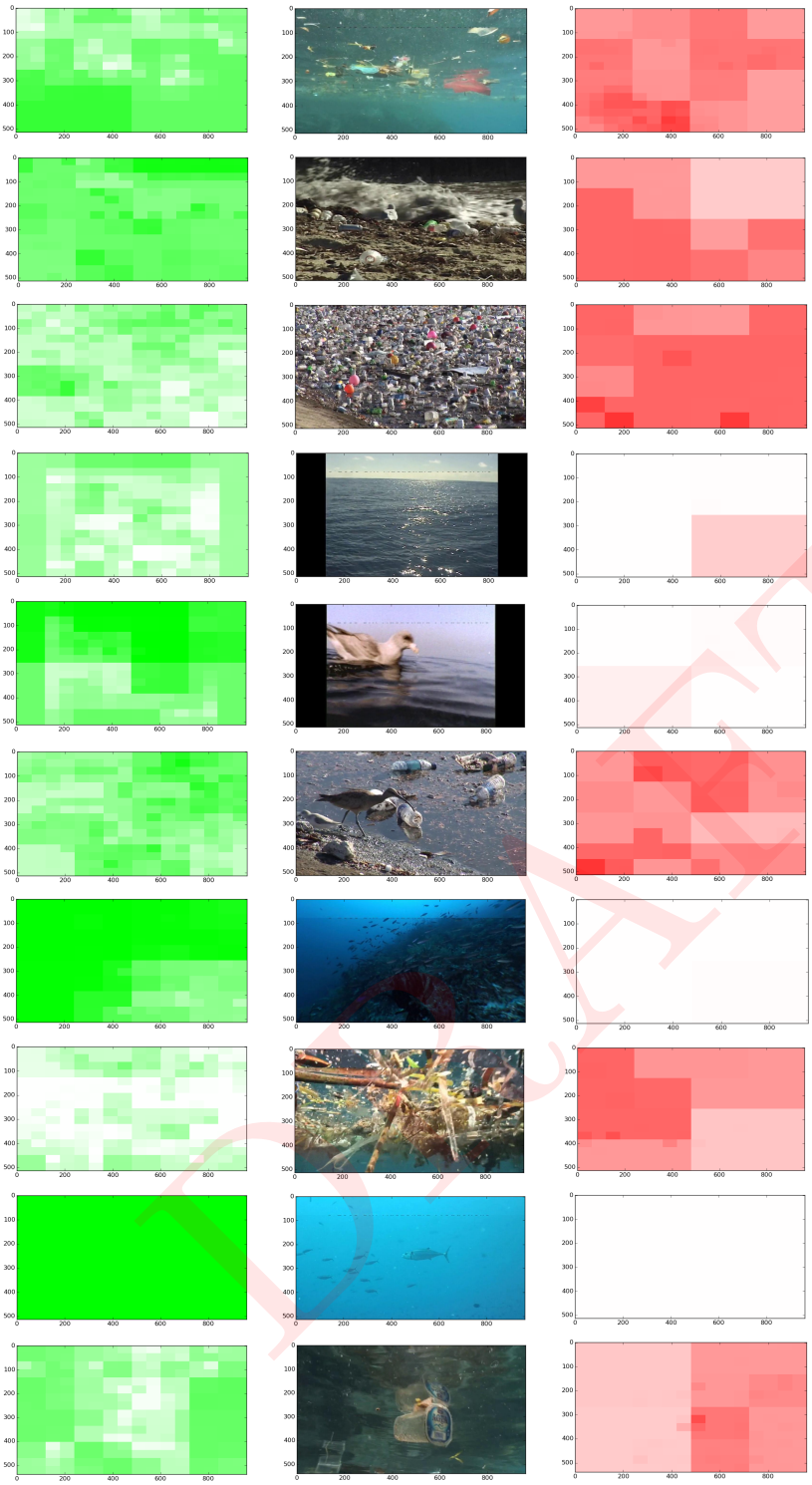

Figure 5.10: Example of outcome of running an image through the segmented image pipeline. Green locates the animals and red the plastic

6 Conclusion

Several conclusions can be made from this project.

Firstly, the Support Vector Machine had promising results. The tests of figure 5.7 show that the linear model has one of the highest accuracies of the different hyperplanes, while also having the smallest time to train and test. The RBF model shows less satisfying results. It takes a large amount of time to train and test on the model, while not being able to gain a high accuracy. The polynomial model trained on the train-set of 14k images shows an interesting trent: the higher the degree of the polynome, the lower the accuracy becomes. In other words, for this problem – detecting two classes from a large amount of feature-vectors – the usage of a linear SVM works best.

The test of figure 5.8 shows an accuracy on plastic and animal detection of less than 1‰. However, because the model is trained on 70% the images in the dataset, which consists mostly of consecutive frames of short film clips, there is a substantial amount of overfitting possible.

Therefore, as shown in figure 5.9 another test using the complete dataset from both above and below water viewpoints, shows how the linear SVM performs on different amounts of train data. 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. 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.

Concluding from the localisation of plastic within the images is difficult. The conducted test shows some results. Several of the images show a higher confidance on locations where the consentration of plastic is higher. However, this is not always the case; especialy with the location of marine-life. More tests should be conducted with this technique, with possibly more annotated data, before a conclusion can be made.

Finally 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 another classifier is a simple technique to gain high accuracy while training on a small dataset.

7 Discussion

TODO: kijken of subsections nodig zijn This project tried to begin the build of autonomous agents that could clean up the ocean of plastic. An agent was not build, however a sizable start was made with the vision of such an agent. Several state-of-the-art algorithms were used on a dataset of images containing floating plastic, animals, both or none to classify the images in one of the classes. One of the algorithms – the Feed Foreward Network – trained on the second-to-last layer of a Convolutional Neural Network, did not work satisfactory.

There are several possible reasons the Feed Forward Network used in this project did not perform well. Because the implementation was not part of a `Python` library, there could be bugs. This however does not seem probable, because the implementation was tested with a xor dataset which performed properly.

More likely is the normalisation of the data. The second-to-last layer of the CNN was not normalised, while the FNN implementation used $tanh()$ to approximate the Sigmoid function. However, even a normalisation of the data did not affect the outcome. Bottom-line, the network was not able to learn the classes from the data.

The accurate results of the Support Vector Machine could be expected. SVMs are known to work well in high dimensional data, as long as the amount of classes to be trained on is small. The 4096 vector of the second-to-last layer of the CNN only needed to be trained on two classes, for which both had an SVM trained on ether showing or not.

TODO: stukje over evaluatie onderzoek

TODO: stukje over vervolgonderzoek, oa verbeteren plastic-herkenning (ook locatie) en maken van autonomous agent voor schepen

This project is one of the many recent projects that show the possibilities with Convolutional Neural Networks. Besides that, it also shows how the 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.

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.

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.

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

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 is gaussian, where the last digit stands 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[3] 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. $\frac{Bco+Aco}{Vsize}$

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
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
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