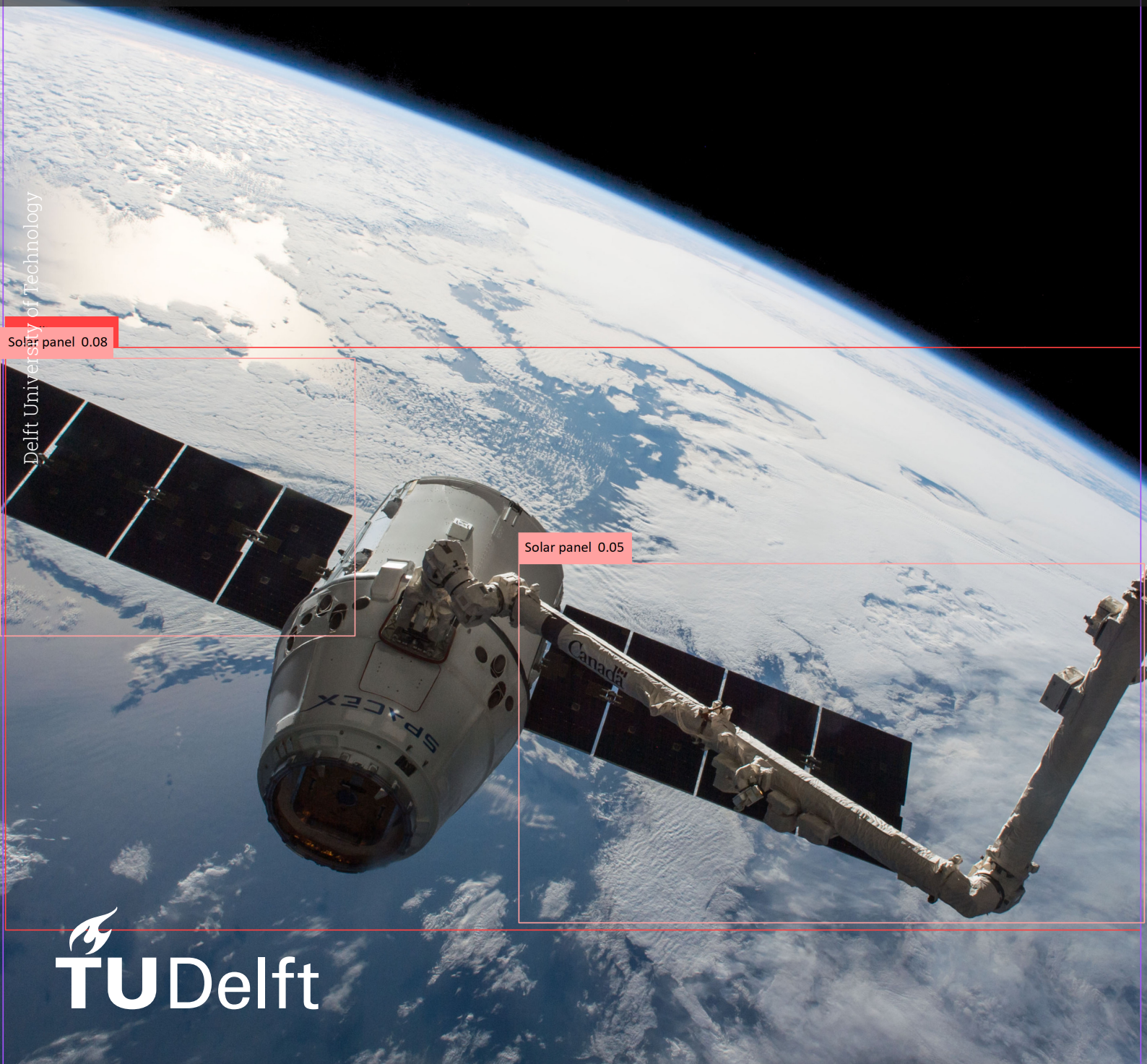
On the Sensitivity of Object Detectors to Background Changes

Marijn de Schipper



Delft University of Technology

TU Delft

On the Sensitivity of Object Detectors to Background Changes

by

Marijn de Schipper

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Wednesday July 17, 2024 at 14:45.

Student number:	4578279	
Project duration:	September 5, 2022 – July 17, 2024	
Thesis committee:	Dr. ir. Jan van Gemert,	TU Delft, supervisor
	Dr. Cynthia C. S. Liem MMus,	TU Delft

Cover: Canadarm 2 Robotic Arm Grapples SpaceX Dragon by NASA under CC BY-NC 2.0 (Modified)
Style: TU Delft Report Style, with modifications by Daan Zwaneveld

An electronic version of this thesis is available at http://repository.tudelft.nl/.

TU Delft

Preface

With this thesis report, I hope to complete my Master in Computer Science at Delft University of Technology. Therefore, I would like to express my gratitude. Firstly, I would like to thank Xin Liu for helping me during the start of my thesis. In addition, I am grateful for the support and guidance Ombretta Strafforello provided throughout my project. Furthermore, I appreciated that Dr. Jan van Gemert gave me feedback and redirection when I felt stuck during the process. For the same reasons, I would like to thank both Michel Rodrigues and the group of students he brought together with whom I was able to finalise the writing. Moreover, I am thankful for my friends and family, who continuously supported me.

Finally, I would like to acknowledge my graduation committee consisting of Dr. Jan van Gemert and Dr. Cynthia Liem.

Marijn de Schipper
Delft, July 2024

Contents

1

Introduction

Object detectors are tasked with identifying and localizing objects in digital images or video. Object detection is crucial for various applications, from autonomous driving to surveillance systems. Modern object detectors are based on models that learn from data. In this process, the model makes predictions and many mistakes, but the model is corrected with data where the answer is known. When the model is sufficiently trained, it can be used to make predictions on new images for which the output is unknown.

Predictions made on unknown data are expected to perform similarly to the performance on known training data when this data is comparable. Changing the background makes the unknown data less similar to the training data. It is easy to see how an object detector can make a mistake and classify an airplane as a boat when the airplane has landed on the water, and the object detector is trained on airplanes in the sky and ships in the water. However, if the object detector is trained on data where objects and backgrounds do not correlate, one would expect the object detector to focus more on the object than the background and be more resilient to changing this background. In this thesis, we look at the sensitivity of object detectors to background changes in a setting without correlations between backgrounds and objects.

The work in Chapter 2 uses carefully controlled synthetic data to show that object detectors are vulnerable to background changes in the uncorrelated setting. Furthermore, it demonstrated that the sensitivity to changing the background depends on the number of unique backgrounds during training, and training on more backgrounds resolves the vulnerability. The research in this chapter is presented in the form of a scientific article and was written for experts in the field of computer vision. Chapter 3 explains relevant technical concepts to make the full report understandable for a non-expert audience. In this chapter, the reader is given some understanding and insight into the workings of a simple model in machine learning. Then, concepts and challenges that also apply to larger models are introduced. The chapter describes what convolutional neural nets are and how these are used in an object detector. Finally, the reader is given some understanding of the object detector used in the scientific article.

2

Scientific article

On the Sensitivity of Object Detectors to Background Changes

Marijn de Schipper

Abstract

Object detectors have come a long way and are used for various applications. In pictures and videos, an object detector must deal with the background. In some settings, this background is indicative of the object; in others, it's not and can even be disruptive. For models trained on data containing correlations between objects and backgrounds (background bias), it makes sense that changing the background can disrupt learned correlations. This paper is interested in how sensitive object detectors are to background changes, specifically when the training data does not contain correlations between objects and backgrounds. Models were trained on carefully controlled synthetic data, so only the backgrounds differed and correlations could be controlled. The results show that models perform better when tested with seen backgrounds than unseen backgrounds. This performance difference diminishes when the model is trained on more unique backgrounds.

1 Introduction

Object detection is the task of predicting the location and class of certain objects (such as cars, traffic lights, or pedestrians) in a picture or video. Object detectors are used in many applications, such as self-driving vehicles, traffic monitoring, or factory automation. In these applications, the objects will appear in front of a background. The background of an image or video can negatively influence deep-learning models. For example, in figure 1, an insect was misclassified as an instrument because the insect appeared in front of a black background

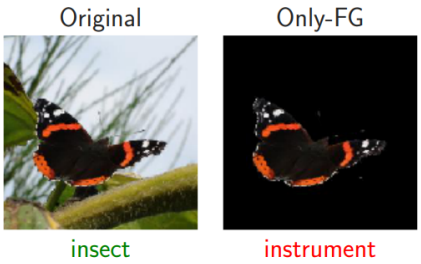


Figure 1: Example of an insect being misclassified when the background is replaced with a uniform black background so only the object (foreground) is accessible. (source Xiao et al. [1])

[1]. Since the model was trained on data where pictures of instruments often have only the instrument on top of a black background, and insects often were photographed in nature, it is easy to see how the background affected this prediction. Another example shows this problem for video. In action recognition, a video of a man singing in a baseball stadium might be misclassified as playing baseball [2]. Wang et al. [3] fittingly called this *background cheating*. This perfectly describes that the model has not actually learned the action but just the associated background. In these examples, the models have learned correlations between objects and backgrounds (background biases) during training. When a learned bias is violated, the model can make mistakes (e.g., an insect classified as an instrument). In a setting where the training data does not contain these biases, changing the background should not be a problem because there is no such bias to violate. However, Chandran [4] has demonstrated that a video object detector can over-

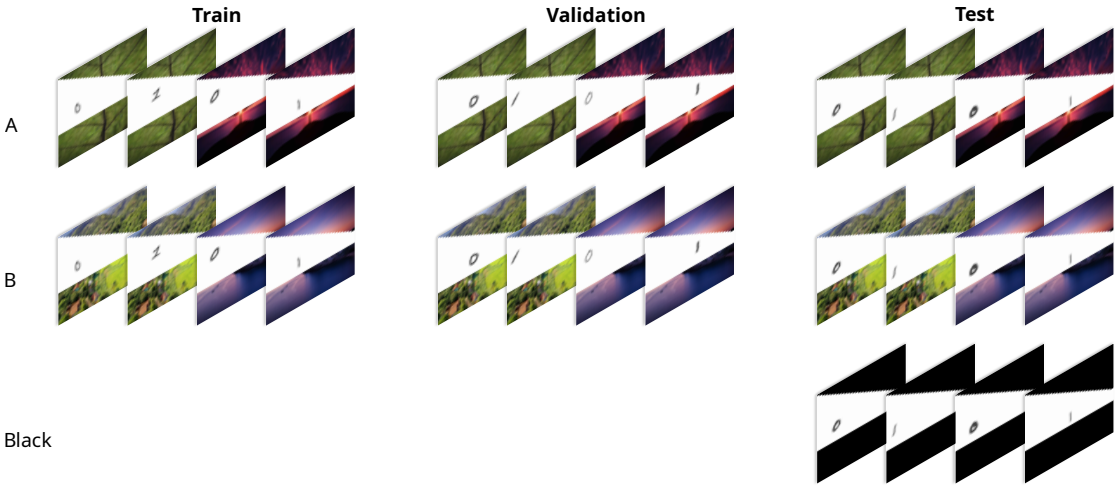


Figure 2: An illustration of some instances of the synthetic dataset. Each row shows an instance where the first column states which background is used. In the case of "A" and "B", this means that backgrounds are sampled from specific subsets from the BG-20K dataset, where in "Black" all backgrounds are full black. Train, Validation, and Test show some examples from the corresponding set. The foreground consists of a static white trapezoid and a digit sampled from MNIST placed uniformly random within the white shape. In contrast to a rectangle, the chosen shape makes cropping a less simple solution. Note that all the foregrounds are the same in the train, validation, and test sets; only the backgrounds differ.

fit on background regardless. This paper aims to research how sensitive object detectors are to uncorrelated background changes, meaning that the background does not contain correlations to the object during training and, therefore, should be without background bias. We made the following contributions:

- We implement a way to generate carefully controlled synthetic data that can be used to test the sensitivity to background changes.
- We show that object detectors can be vulnerable to uncorrelated background changes.
- We show that with enough unique backgrounds during training, object detectors overcome the vulnerability to uncorrelated background changes.

The synthetic data can be downloaded from: `https://drive.google.com/file/d/1fay4FdUbq4sW3Hl6m13e4TmmhrT9R9co/view?usp=sharing`

2 Related work

2.1 Background and other deep learning tasks

Generally, in public datasets, correlations occur between the annotation and (context from) the background. CNNs trained for various tasks can learn to use these correlations. In a deep metric learning task, where the task is to retrieve images of objects similar to a query image of an object, the object's background is usually not interesting and should not matter. However, Kobs and Hotho [5] refutes this expectation. When querying a sportive-looking bike in front of a white background, they received other sportive bikes in front of white backgrounds. But when they queried the same bike in front of a brick wall, they retrieved images with (parts of) bikes in front of brick walls. This is undesirable because the retrieval system should only consider the object, not the background. The authors tackled this problem with data augmentation by replacing the back-

ground at training time.

For classifying objects, it is shown that when the foreground is partly covered, the background can still provide information and thereby perform significantly better than randomly guessing the class [1]. Furthermore, when the foreground is visible but the background is swapped for a background from another class, the accuracy decreases significantly. When evaluating model robustness to adversarially chosen backgrounds, for 87.5% of foregrounds, a background could be chosen such that the new image was misclassified as the background class.

Aniraj et al. [6] also showed reduced performance when testing classification models in a setting with Out-of-Distribution backgrounds. This decreased performance was especially harmful in fine-grained classification tasks, such as identifying bird species, because bird species are often closely related to their habitat. This paper applied two masking strategies to improve the Out-of-Distribution results: early masking and late masking. With early masking, the background is masked before the feature maps are generated and used by the classifier. With late masking, the same segmentation is resized and used to mask the feature maps which are then used by the classifier.

These papers show the (negative) impact that background can have in various deep learning tasks. They further come up with different strategies to make their models more robust to background changes. The papers above provided research insight into the influence of the background when there is a correlation between the background and the object. However, in this paper the effect of changing the background when the background and foreground are not correlated is specifically researched.

2.2 Image object detectors

Object detectors predict the identity and location of an unknown amount of objects based on a picture or video. Object detectors often use a Convolutional Neural Network (CNN). CNNs typically have a fixed output size (for example, a classifier would have one output neuron per class) which can be a problem when you want to predict an unknown amount of objects in an image. A common approach for a detector is to propose regions of interest and classify those regions (two-stage). R-CNN [7] used selective search [8] to extract around 2000 regions which were then classified and the box prediction refined. Fast-R-CNN [9] improves the speed of training and testing but still uses selective search which bottlenecks the speed. In Faster-R-CNN [10] the selective search is replaced by a region proposal network increasing both the speed and accuracy even further.

Another approach is to predict the boxes with classification in one step (one-stage). For example, YOLO [11] generates feature maps from the image and splits these features in a S x S grid. Each cell in this grid corresponds to part of the image and a fixed number of predictions (both for box and classification) is made per cell. Therefore, it is hard to detect a lot of small objects (especially when they are close together). On the other hand, this approach is generally even faster than two-stage models and therefore more interesting for real-time applications.

Both approaches have plenty of successors, including models specialized for video object detection. In this paper, we will use an object detector based on YOLO because of its access to the full background. Since it is shown that CNNs can learn and exploit absolute spatial location [12], full access to the background could be exploited by an object detector to learn where objects can appear. Furthermore, the fact that there was a code base available that could easily train on custom datasets was another practical reason.

2.3 Background and object detection

Kayhan and van Gemert [13] showed that object detectors with the means to take the background into account can and will use the background. These object detectors either use the whole image or crop the feature map and therefore have access to all or part of the information

in the background. In a setting where the object correlates to the background, these object detectors exploit the background to gain better performance than object detectors that are unable to use the background. Exploitation of the background could be an advantage for a higher score on a benchmark dataset if the objects in the dataset (partially) correlate to (things in) the background. Object detectors that do not access the background are less vulnerable when the correlation with the background changes.

Chandran [4] showed that a video object detector, SELSA [14], is vulnerable to background changes, even if trained on a synthetic dataset (SB-MNIST) containing no correlation between object and background. In a video from a stationary camera, the background where a moving object has no chance of appearing is called the static background. Changing this static background at runtime should not interfere with object detection, however, this paper showed reduced performance when changing the static background at test time. This is interesting because the background where moving objects can appear remains unchanged.

The papers discussed show that object detectors can both benefit from and be disadvantaged by the background. In this paper, we will further look at the impact of uncorrelated backgrounds on object detection. We will do this for images, and not in video, to find out when the problem occurs and do this using synthetic data based on SB-MNIST.

3 Methodology

3.1 Why synthetic data?

Object detection datasets, such as MS COCO [15] and Pascal VOC [16], often contain images where the background can provide useful information about the object's location or class. For instance, images with a lot of water in the background are more likely to contain boats than airplanes. To test the impact of the background on object detection, we constructed synthetic datasets to control these class-background dependencies.

3.2 Construction of synthetic datasets

For each experiment, we constructed specific datasets for the different requirements, however, the core parts of all these datasets are the same. See figure 2 for a concept visualization. Each dataset has a total of 1500 images where 500 are used for training, 500 for validation, and the remaining 500 for testing. Each image is 64 by 64 pixels and made up of a foreground and a background. The foreground consists of a digit image sampled from MNIST [17]. The image is scaled down to 10 by 10 pixels and placed in a static white area. The static white area always has the shape of the same trapezoid at the same location. A trapezoid was used instead of a square to ensure that simply cropping is not a solution. The digits' location is uniformly distributed within the trapezoid such that a square around the digit does not intersect with the border of the trapezoid. The background is sampled from a set of background images and scaled down to 64 by 64 pixels. When constructing different instances of the same dataset, the same random seed is used for this process but different sets of backgrounds. This way, the multiple datasets have exactly the same foregrounds (meaning the same digits sampled in the same locations) for their train, validation, and test sets but different backgrounds.

3.3 Model implementation details

For the experiments, we trained YOLOv5. Specifically, the YOLOv5 without pre-trained weights version from the repository managed by Ultralytics [18]. Training with pre-trained weights gave worse results on the synthetic dataset. Each model was trained with early stopping [19] for a maximum of 2000 epochs. Training was stopped early when there was no improvement for 100 epochs. For evaluation, we use mean Average Precision [20, 21] at an Intersection of Union (IoU) of 0.5 (mAP@50).

4 Experiments

4.1 (When) does an uncorrelated background affect object detection?

In object detection, background bias can be useful. For example, if the object you want to identify is only partially visible or blurred. Therefore, it is logical that these biases are learned by object detectors; then, in a setting that does not conform to these biases, the object detector is vulnerable to making mistakes. Now, in a setting with no such biases to be learned, is an object detector still vulnerable to changing the background at test time? If so, when is this the case?

For this experiment, we created multiple instances of the synthetic dataset. For the backgrounds, we used the BG-20K [22] dataset, which contains high-resolution backgrounds without salient objects. From this dataset, 5 subsets (A, B, C, D and E) were sampled without intersections. These subsets are then used to sample the backgrounds for the instances. The foreground and background were sampled in such a way that each background was used an equal amount of times for each class. Furthermore, all classes were perfectly balanced. Then, for each instance, a model is trained and tested on its own test set (seen) and the test sets of the other instances (unseen). This was done for different amounts of distinct backgrounds sampled (N).

In figure 3, we see that with only a few distinct backgrounds seen at training time, changing to unseen backgrounds significantly decreases performance. Also, note that increasing the number of backgrounds solves this decrease quickly, since with 10 different backgrounds the difference is already minimal. Note that with fewer unique backgrounds, the standard deviation is high. In table 1, we see this is due to the wildly varying performance on unseen backgrounds. With increasing numbers of unique backgrounds, the performance range on unseen backgrounds narrows; thus, the standard deviation decreases.

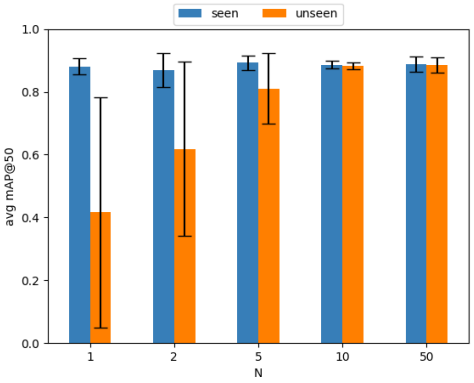


Figure 3: **Seen backgrounds vs unseen backgrounds** Average performance (mAP@50) of YOLOv5 when tested with backgrounds encountered during training (seen) as well as tested with backgrounds not encountered during training (unseen), for different numbers of backgrounds (N). Testing on different backgrounds results in a significant performance gap compared to testing on backgrounds seen at training time for models trained on only a few different backgrounds. With fewer unique backgrounds, the performance on unseen backgrounds can vary wildly and thus have a high standard deviation.

Although only the background may differ, one could argue that the test set should be identical to compare fairly. Therefore, the models were also tested on three additional test sets. Firstly, "black" is a test set where the background is masked black, see figure 2. Secondly, "stratified" is a test set where the background consists of 50 different backgrounds sampled from a different distribution than the backgrounds from the train sets. For this, the stratified category of the DTD dataset [23] was used. Lastly, "rest" is a test set where the backgrounds consist of 50 different backgrounds sampled from the same distribution as the backgrounds from the train sets. For this, 50 different backgrounds were sampled from the BG-20K dataset such that the sampled backgrounds did not occur in the A, B, C, D and E subsets from which the train set backgrounds were sampled.

mAP@50 N=1		Tested on A	B	C	D	E	Rest	Stratified	Black
Trained on	A	**85.6**	79.4	70.8	73.8	3.1	61.1	68.7	54.9
	B	53.8	**88.4**	84.9	**88.4**	1.6	78.1	85.0	81.8
	C	0.0	0.3	**91.6**	0.6	0.1	20.3	4.5	5.9
	D	78.5	86.6	83.3	**90.0**	1.2	63.8	70.8	87.2
	E	2.4	38.5	65.0	19.5	**84.9**	15.4	30.9	2.1

mAP@50 N=2		Tested on A	B	C	D	E	Rest	Stratified	Black
Trained on	A	84.3	61.4	64.5	83.2	3.9	55.4	58.4	**84.5**
	B	76.6	**77.2**	72.1	75.1	7.2	60.9	44.2	75.4
	C	12.7	31.1	**91.0**	90.8	32.2	31.7	9.8	86.4
	D	87.8	86.2	90.0	**91.4**	57.9	77.8	62.7	89.7
	E	84.6	73.1	67.5	78.4	**90.2**	50.9	59.4	75.4

mAP@50 N=5		Tested on A	B	C	D	E	Rest	Stratified	Black
Trained on	A	**86.5**	84.3	86.2	86.0	85.1	84.7	84.4	86.2
	B	67.1	**89.4**	88.0	65.4	50.0	85.1	87.9	89.0
	C	74.8	**87.7**	**87.7**	86.5	66.9	84.6	86.1	87.5
	D	92.8	91.2	92.1	**93.3**	76.4	91.5	89.9	92.8
	E	71.8	89.1	89.7	89.3	89.7	86.6	80.3	**89.8**

mAP@50 N=10		Tested on A	B	C	D	E	Rest	Stratified	Black
Trained on	A	88.4	88.3	**88.5**	87.7	87.6	87.8	**88.5**	88.4
	B	87.5	88.4	88.3	88.3	**88.8**	88.1	88.1	88.4
	C	89.2	90.5	91.0	90.8	90.3	90.3	90.2	**91.4**
	D	86.7	**87.5**	87.2	87.1	87.0	**87.5**	87.3	87.3
	E	87.4	88.3	88.4	**88.8**	88.4	88.1	88.0	88.7

mAP@50 N=50		Tested on A	B	C	D	E	Rest	Stratified	Black
Trained on	A	**91.5**	91.3	91.0	91.4	91.0	91.0	91.2	90.2
	B	84.0	**85.4**	85.1	85.2	85.0	85.1	84.9	85.2
	C	90.0	90.0	90.6	90.1	**90.8**	90.3	90.2	90.3
	D	90.0	**90.7**	90.1	90.3	90.2	90.1	90.1	90.0
	E	86.4	86.8	86.5	86.8	86.5	87.0	86.9	**87.5**

Table 1: **Extensive results**
These tables show the performance (mAP@50, scaled from 0 to 100) of YOLOv5 trained on different instances (A, B, C, D and E) of the same data and tested on the test sets of these instances and three fixed test sets, for different numbers of unique backgrounds (N). For each model, the best performance is shown in bold. The performance on seen backgrounds (A on A, B on B, etc.) is consistently among the highest for each model. Performance on unseen backgrounds (A on B, A on C, etc) is sometimes similar to the performance on seen backgrounds, but other times decreases slightly or even dramatically. This explains the high standard deviation in figures 4 and 3. Note that the model trained on instance C with $N = 1$ performs especially badly when tested on unseen backgrounds.

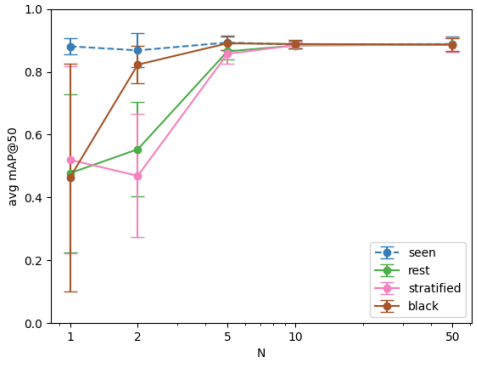


Figure 4: **Testing on unseen backgrounds but fixed** N Average performance (mAP@50) of YOLOv5 tested on 3 test sets without background encountered during training. These test sets have a fixed number of backgrounds. The "black" test set always has a black background. The "rest" test set has 50 distinct backgrounds all sampled from BG-20K in such a way that there is no intersection with the backgrounds selected for the training sets. The "stratified" test set also has 50 distinct backgrounds but is sampled from the stratified category of the DTD dataset. The performance tested with backgrounds during training (seen) is also included as a baseline. The test sets used for the baseline have the same number of backgrounds as the train set each model was trained on. Testing on different backgrounds results in a significant performance gap compared to testing on backgrounds seen at training time for models trained on only a few different backgrounds. With fewer unique backgrounds, the performance on unseen backgrounds can vary wildly and thus have a high standard deviation.

The results are shown in figure 4, where for comparison the performance of "seen" is also included. Note that models trained on a few different backgrounds also perform worse on the additional datasets with unseen backgrounds. Furthermore, from 10 different backgrounds onward, the performance gap disappears. Lastly, the performance gap on black decreases faster than rest and stratified. The standard deviation follows the same trend as the previous experiment. The initial performance gap and the diminishing of this gap over the number of training backgrounds are also similar to the results from the previous experiment. Therefore, we can conclude that for a small number of backgrounds used in training, uncorrelated background changes significantly decrease performance. The problem is resolved when enough different backgrounds are used during training.

5 Limitations and Conclusion

In this paper, we investigated the influence of changing uncorrelated backgrounds in object detection. We used the YOLOv5 model and demonstrated that the model is vulnerable to uncorrelated background changes when trained on a small number of unique backgrounds. While we understand that testing on only YOLOv5 makes it harder to translate our findings to other object detectors, many object detectors use a CNN as a backbone and have access to backgrounds comparable to YOLOv5. Therefore, we can speculate that our results would generalize to other object detection models. For the experiments, we created synthetic data to ensure that there were only changes in the background and there were no correlations between objects and backgrounds. This has the drawback that it is harder to relate to real data. In real data, backgrounds generally have some correlations with the objects. Furthermore, we observed that the problem is resolved when the model is trained on enough unique backgrounds.

To conclude, we showed that object detectors can be sensitive to background changes, even when the background is uncorrelated. We did this by carefully generating synthetic data for our experiments. Lastly, we showed that the problem is resolved with enough unique backgrounds.

References

[1] K. Xiao, L. Engstrom, A. Ilyas, and A. Madry, "Noise or signal: The role of image backgrounds in object recognition," *arXiv preprint arXiv:2006.09994*, 2020.

[2] J. Choi, C. Gao, J. C. E. Messou, and J.-B. Huang, "Why can't i dance in the mall? learning to mitigate scene bias in action recognition," in *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds., vol. 32. Curran Associates, Inc., 2019. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2019/file/ab817c9349cf9c4f6877e1894a1faa00-Paper.pdf

[3] J. Wang, Y. Gao, K. Li, Y. Lin, A. J. Ma, H. Cheng, P. Peng, F. Huang, R. Ji, and X. Sun, "Removing the background by adding the background: Towards background robust self-supervised video representation learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2021, pp. 11 804–11 813.

[4] G. R. Chandran, "Significance of static backgrounds for video object detection," 2022. [Online]. Available: http://resolver.tudelft.nl/uuid:612171db-f125-4f60-ad9e-eab38bf3b7e1

[5] K. Kobs and A. Hotho, "On background bias in deep metric learning," in *Fifteenth International Conference on Machine Vision (ICMV 2022)*, W. Osten, D. P. Nikolaev, and J. J. Zhou, Eds., International Society for Optics and Photonics. SPIE, 2023, p. 1270114. [Online]. Available: https://doi.org/10.1117/12.2679270

[6] A. Aniraj, C. F. Dantas, D. Ienco, and D. Marcos, "Masking strategies for background bias removal in computer vision models," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) Workshops*, October 2023, pp. 4397–4405.

[7] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2014.

[8] J. R. Uijlings, K. E. Van De Sande, T. Gevers, and A. W. Smeulders, "Selective search for object recognition," *International journal of computer vision*, vol. 104, pp. 154–171, 2013.

[9] R. Girshick, "Fast r-cnn," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, December 2015.

[10] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *Advances in neural information processing systems*, vol. 28, 2015.

[11] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[12] O. S. Kayhan and J. C. v. Gemert, "On translation invariance in cnns: Convolutional layers can exploit absolute spatial location," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[13] O. S. Kayhan and J. C. van Gemert, "Evaluating context for deep object detectors," *arXiv preprint arXiv:2205.02887*, 2022.

[14] H. Wu, Y. Chen, N. Wang, and Z. Zhang, "Sequence level semantics aggregation for video object detection," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

[15] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and

C. L. Zitnick, "Microsoft coco: Common objects in context," in *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*. Springer, 2014, pp. 740–755.

[16] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes (voc) challenge," *International Journal of Computer Vision*, vol. 88, no. 2, pp. 303–338, Jun. 2010.

[17] Y. LeCun, C. Cortes, and C. Burges, "Mnist handwritten digit database," *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist*, vol. 2, 2010.

[18] G. Jocher, "Yolov5 by ultralytics," 2020. [Online]. Available: https://github.com/ultralytics/yolov5

[19] Y. Yao, L. Rosasco, and A. Caponnetto, "On early stopping in gradient descent learning," *Constructive Approximation*, vol. 26, no. 2, pp. 289–315, 2007.

[20] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein *et al.*, "Imagenet large scale visual recognition challenge," *International journal of computer vision*, vol. 115, pp. 211–252, 2015.

[21] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes (voc) challenge," *International journal of computer vision*, vol. 88, pp. 303–338, 2010.

[22] J. Li, J. Zhang, S. J. Maybank, and D. Tao, "Bridging composite and real: towards end-to-end deep image matting," *International Journal of Computer Vision*, vol. 130, no. 2, pp. 246–266, 2022.

[23] M. Cimpoi, S. Maji, I. Kokkinos, S. Mohamed, , and A. Vedaldi, "Describing textures in the wild," in *Proceedings of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2014.

3

Supplementary materials

This chapter aims to provide the reader with a basic understanding of the technical concepts used in the scientific article (Chapter 2). The article assumes the reader is knowledgeable about training a neural network, deep learning, and the basics of object detection. In the following sections, the basic ideas about these topics are explained.

3.1. Neural networks

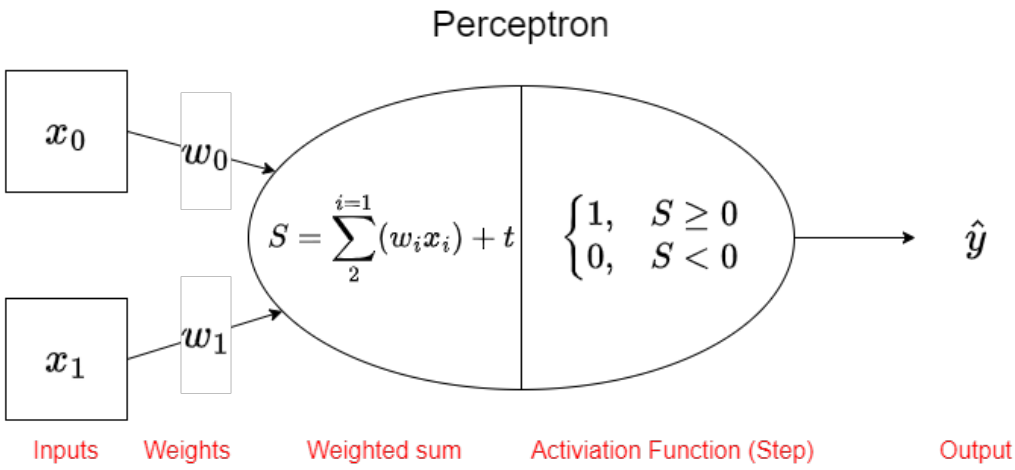


Figure 3.1: Simplified illustration of a perceptron with 2 inputs and 2 corresponding weights, which are used in the body of the perceptron for the weighted sum. There, a step function is applied as the activation function to determine the final output of the perception, 0 if the sum S is negative otherwise the output is 1. (Made with draw.io)

In machine learning, a neural network is a model inspired by how actual neurons function in a brain. To understand a neural network, we first look at a perceptron, which represents a single neuron. In biology, a neuron is a cell that receives and transmits signals. A neuron has dendrites that pick up signals. When the combined signals are strong enough, the neuron transmits a signal through its axon terminals. Note that each input signal is not necessarily treated equally, meaning some inputs are weighted heavier than others. This concept is the basis for a perceptron. In a perceptron, each input has its own weight, so the neuron can be more or less sensitive to some input signal. In figure 3.1, a simplified perceptron is illustrated, which could be used for linear binary classification. For example, we want to classify an animal as either a cat or a dog based on some score for size and domestication. In this example, the inputs (x_0 and x_1) represent some features of the animal we want to classify, such as size and domestication. For each input (x_i), there is a weight (w_i); the input and corresponding weights are multiplied and then summed, and a threshold (t) is added. An activation function is applied to the resulting sum (S), determining if the neuron should fire. In the figure, a step function is used where if the total input is equal to or above 0, the perceptron's output ($\hat{y}$) is 1. Then, the animal is interpreted as a cat. If the weighted sum (S) is below 0, the output of the activation function is 0, and the animal is interpreted as a dog. The threshold (t) can be a negative number and determines how high the line representing the class boundary is drawn; see figure 3.2. The weights determine the angles of this line. Weights can also be negative, and then the corresponding input has an inhibitory effect.

The trick is finding good values for the weights and threshold. When these are initiated randomly, these can result in mistakes. In supervised machine learning, data for which the output is known (labelled data) is used to update the weights and threshold. This iterative process is shown in figure 3.2. To update the weights, the predicted output is compared to the known label (y). The weights are updated with the following formula:

$$w_i = w_i + \alpha * (y - \hat{y}) * x_i$$

The predicted output ($\hat{y}$) is compared to known label (y). If these are equal ($y - \hat{y} = 0$), the weight (w_i) remains unchanged. Otherwise, this error determines the direction in which the weights are moved. The input (x_i) and the learning rate (a) determine the amount the weights are moved. The learning rate is a positive number, generally less than one. When the learning rate is exactly one, that would be the same as not having a learning rate, which is not required for perceptron. The learning rate controls the pace of the update process. Perceptrons are proven to converge when the data is linearly separable [4], meaning it is possible to draw a straight line to separate both classes. If the classes overlap (inseparable), a perceptron cannot solve the problem perfectly. Figure 3.3 visually shows this. When the problem is not linearly separable, sometimes an imperfect line can still be drawn, and most points can be classified. However, the perceptron can have problems converging, such as thrashing

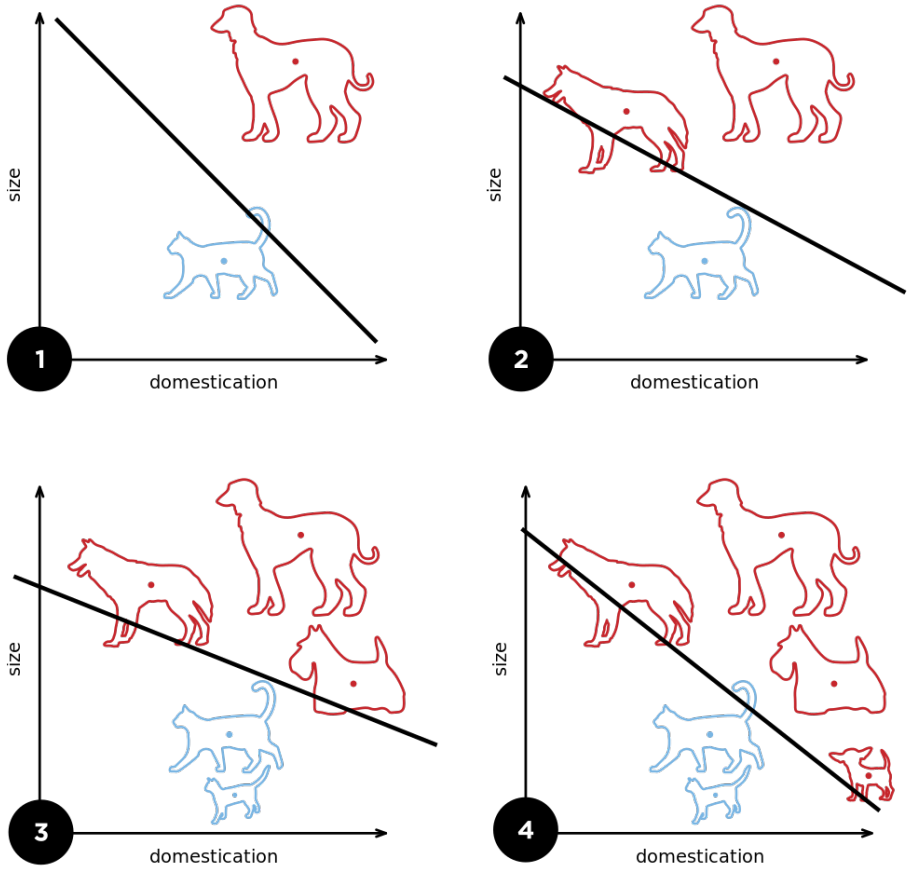


Figure 3.2: Illustration of how the boundary line in a perceptron updates after training on some data points. In these figures an animal is classified based on its size and domestication. For example, when an animal is large and scores somewhere in the middle on domestication, it will be classified as a dog. (Source Wikipedia [2])

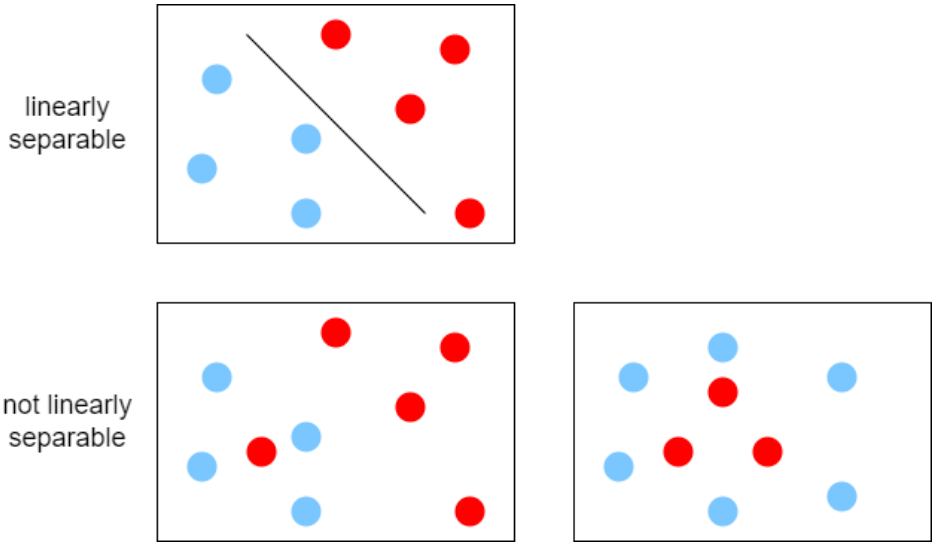


Figure 3.3: Visualisation of linear separability. Above, a clear boundary line can be drawn to separate the data points by class. Below, two examples are shown where this separation is not possible with a straight line. (Made with draw.io)

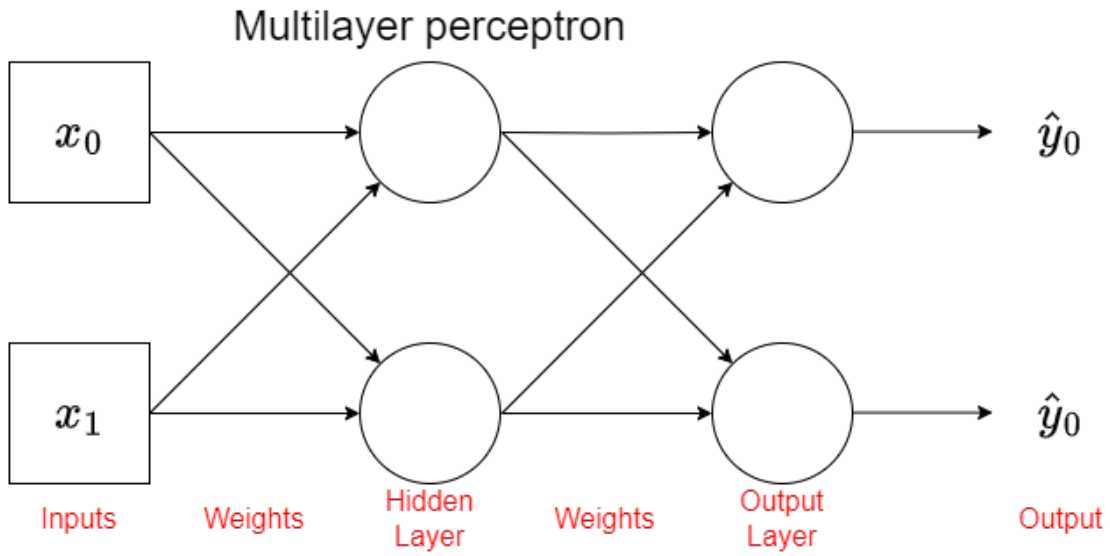


Figure 3.4: A schematic diagram of a multilayer perceptron (MLP) is shown. The MLP consists of multiple neurons (circles) that have weights for each incoming arrow, but these weights are not explicitly shown. (Made with draw.io)

between boundary positions. A lower learning rate can reduce this thrashing. This is not a perfect solution and shows a limitation of a perceptron.

The multilayer perceptron (MLP) improves on the perceptron since it can deal with not linearly separable data. In figure 3.4, an MLP is shown. This model consists of multiple neurons whose outputs are the inputs for the next layer. The setup of each neuron is similar to how the perceptron functions, where each neuron takes the weighted sum of its inputs and applies an activation function. The outputs of the neurons in one layer are passed to the neurons in the next layer as inputs. An MLP has at least one hidden layer but can have more. The last layer is the output layer. The amount of output neurons for classification is typically the same as the number of classes. Then, each output neuron corresponds with a class. For example, in figure 3.4, the two output neurons could correspond with the classes cat and dog. Then, the MLP predicts a cat if the corresponding neuron outputs the highest value. Otherwise, it predicts a dog.

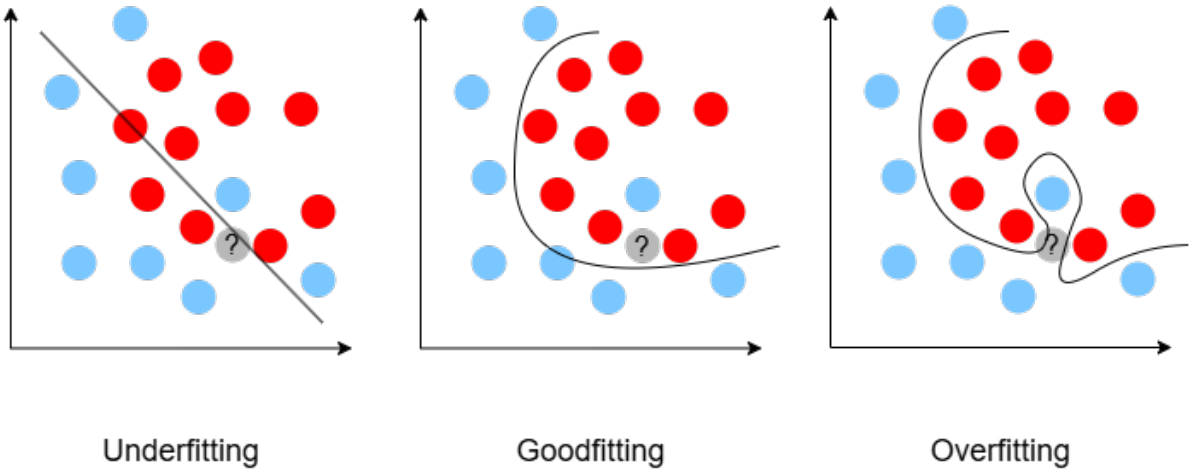


Figure 3.5: Visualisation showing underfitting, goodfitting and overfitting. With a good fit, the model can accurately make predictions for new data, in other words, it can generalize well. A new data point (shown with "?"), will be classified as the class in red. With underfitting, the model is too simple to capture the complexity of the data. With overfitting, the model fits the data too tightly and therefore is less able to generalize. In both these cases, the new data point will be classified as the class in blue. (Made with draw.io)

3.1.1. Training

Updating weights with data is called training. Training data (input data with expected output) is used to update the weights. The weights are updated differently in an MLP than in a perceptron because the output of the neurons in any layer depends on the neurons in the previous layers. Instead of a single formula, backpropagation is used. In backpropagation, the gradient of a loss function with respect to each weight is calculated. This is done one layer at a time, starting from the output layer and moving backwards. Then, these gradients are used to update the weights in a process called gradient descent. This uses a learning rate to limit how fast the weights are updated to converge to a minimum.

3.1.2. Overfitting

Overfitting in machine learning means the model cannot generalize and fits the training data too closely. This can occur when a model is trained too long on the training data. To prevent this, the training data is often split into a train set and a validation set. The train set is strictly used to train on and the validation set is used to calculate the loss in each training iteration. During training, both the loss on the train set and the validation set will go down, but the validation error will rise when overfitting occurs.

3.1.3. Bias

When the training data contains biases, machine learning models can learn these biases. This can cause problems for the use cases of the models. For example, a model trained on historical data about salaries can learn that women earn less than men, which can be unwanted because you probably want the model to be based on other factors such as education, experience, and skills.

3.2. Deep Learning

Deep learning is a subset of machine learning. An MLP can be considered a deep neural network if it has three or more layers. Adding more layers to a neural network increases its complexity and, therefore, increases its potential to learn more complex patterns (also increases its potential for overfitting). More layers also increase the computational cost. The way weights are updated for an MLP with only one hidden layer or many layers is the same, therefore the size of the network can be easily changed.

3.2.1. Convolutional Neural Networks

Convolutional neural networks (CNN) are a specific kind of neural network often used to extract features from images and thus used as a backbone in various deep learning tasks. Say you want to make a classifier to classify black-and-white pictures of animals as either cats or dogs. Then for each pixel, you have a number representing how bright that pixel is. An MLP can be constructed with an input for each pixel. This will result in an MLP with many neurons and thus parameters, increasing the computational cost. Instead of learning weights of neurons directly connected to the input, a CNN learns convolutional kernels. These kernels are matrices of values that slide across the input. For each position of the kernel on the input, the kernel is elementwise multiplied with the corresponding input, and summed. Just like in a neuron, an activation function is applied to suppress irrelevant data. A convolutional layer can have and apply multiple kernels. The amount of kernels determines the depth of the output. Without padding, the width and height of the output will shrink, but by adding zeros around the input (zero padding) the width and height of the output remain the same as the input. Reducing the output size can be beneficial for aggregating information for feature extraction and it reduces computational cost. The output size can also be reduced by skipping a few pixels when sliding the kernel over the input (stride) or by adding max pooling layers. Max pooling layers slide a window over its input, and then for each input within the window, the maximum is taken as the result. A typical max pooling layer would have a window size of 2 and a stride of 2, therefore the height and width of the output is half of the inputs.

In a CNN, convolutional layers are often alternated with max pooling layers. In the previous example, we want to make a classifier for images of cats and dogs. For this, we need to extract features from the input images. By alternating some convolutional layers and max pooling layers, we can extract these

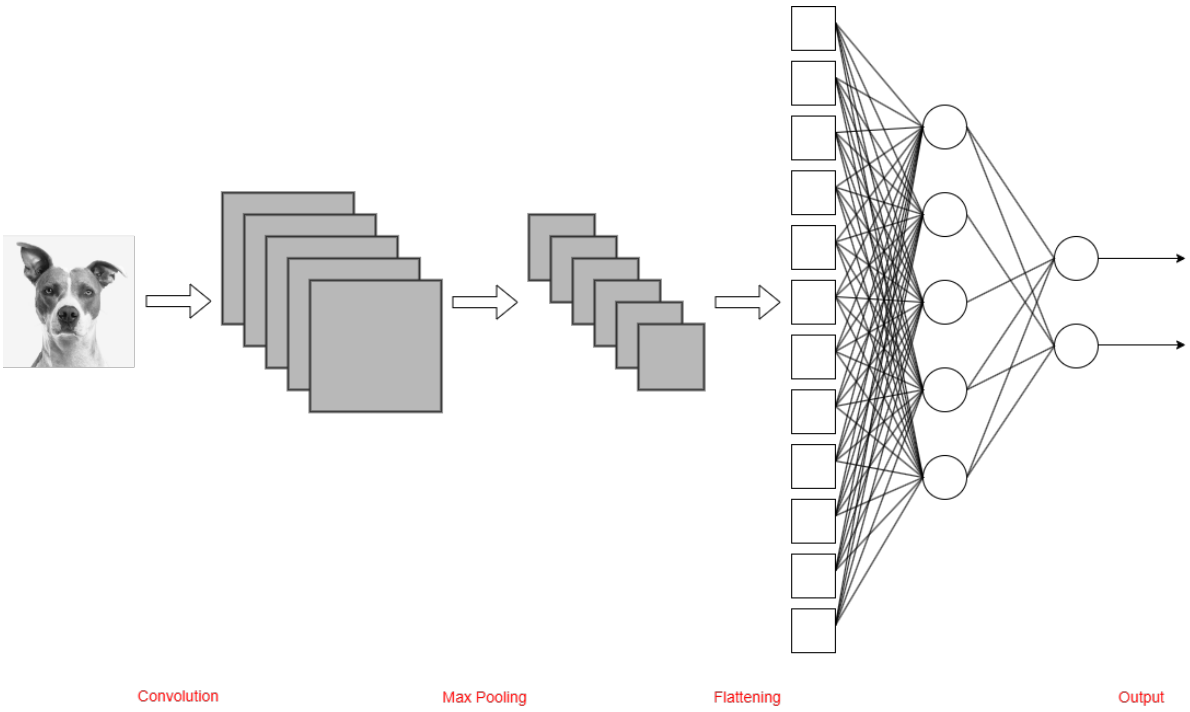


Figure 3.6: Illustration of a classifier using a CNN. On the left, an image is shown as input, followed by a visualisation of the output of a convolutional layer and that of a max pooling layer. The output is then flattened, so it can be used as input for an MLP. (Made with draw.io)

features shrinking the output in the width and height dimensions. These layers form the backbone of the classifier. The part that will make the final prediction is called the head. When the output size is small enough, the output is flattened to a single vector. Then the head consists of an MLP which takes the flattened output of the backbone as input to make its predictions. Figure 3.6 shows an illustration of this example.

CNNs work particularly well for grid-like data such as images because they exploit the prior knowledge that information from images is often translation invariant. The kernel values are learned similarly to the weights of neurons in an MLP.

3.3. Object Detection

The task of object detection is different from that of classification. Object detection is partly a classifying task but is also a regression task. Regression is the task of predicting a continuous variable, in the case of object detection, this means predicting values for the position and shape of an object. Deep learning is generally well-suited for regression tasks, and modern object detectors incorporate CNNs. The difficulty lies in making an efficient model that can predict an unknown amount of objects since CNNs and MLPs have a fixed output size.

3.3.1. YOLO

In the scientific article, we used YOLOv5 [3]. This model is based on the model named You Only Look Once (YOLO) [5]. In YOLO, the image is processed in a $S \times S$ grid, see figure 3.7. The input passes through several convolutional layers and max pooling layers, after which a fully connected network makes the final prediction. This output is reshaped into the $S \times S$ grid. Then, the output contains a vector of length $B * 5 + C$ for each cell, where C values are interpreted as the class probabilities. B is the amount of bounding boxes predicted per cell, and for each bounding box, 5 values are interpreted as the x, y, width, height, and confidence. This architecture is visualized in figure 3.8. When the box predictions overlap, the Intersection over Union (IoU) is calculated, and if this is above a threshold, the box predictions presumably point to the same object. Only the box prediction with the highest confidence is kept.

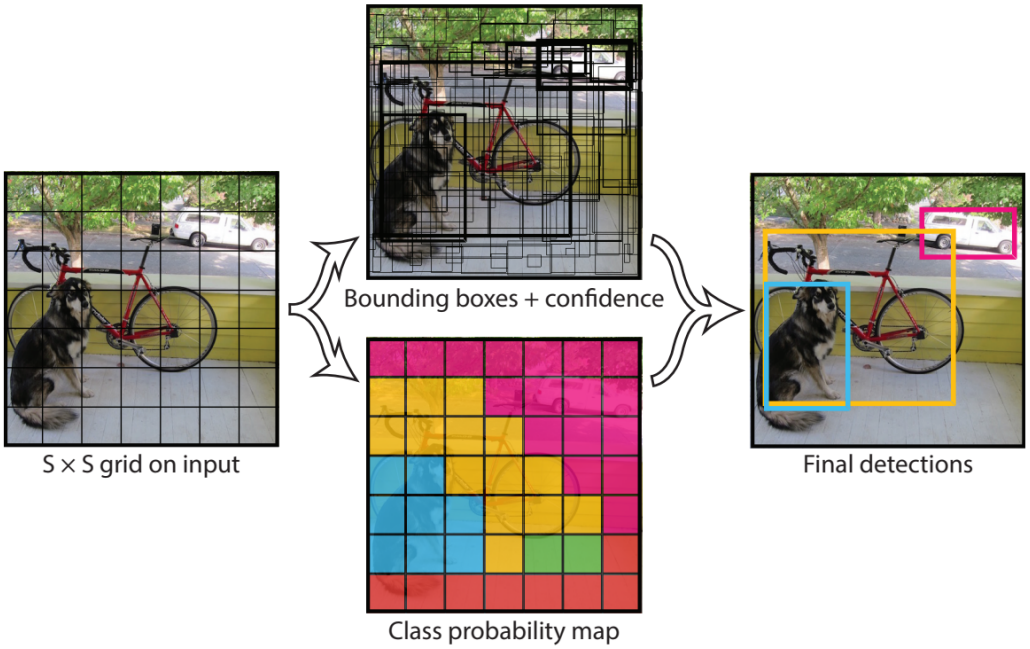


Figure 3.7: Illustration of how YOLO divides the input into a grid. For each cell in the grid, both bounding boxed, confidence scores, and class probabilities are predicted. Together, they are used to create the final predictions. (Source Redmon et al.[5])

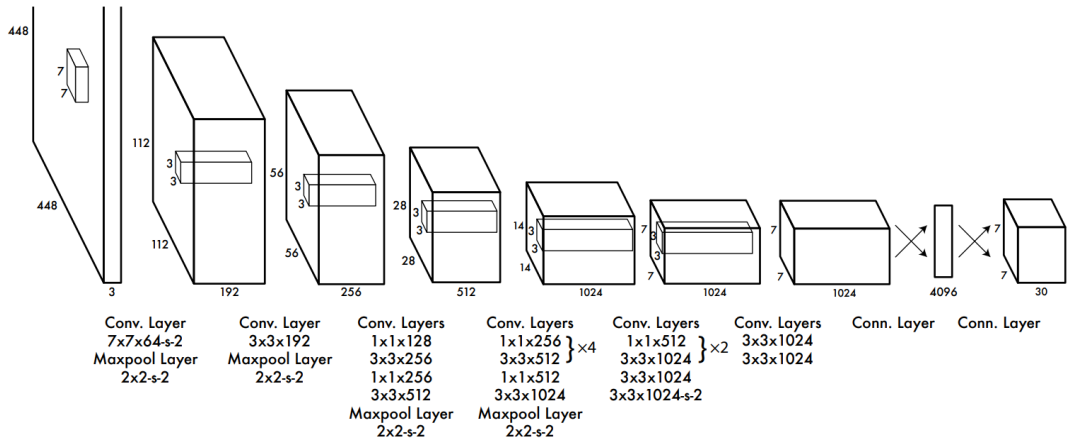


Figure 3.8: Illustration of the architecture of YOLO. It shows all layers and the sizes of the data in each stage. The first block represents the input image of 448 by 488 pixels in 3 channels (RGB). (Source Redmon et al.[5])

3.3.2. mean Average Precision

Different metrics can be used to evaluate the performance of object detection models. One such metric is mean Average Precision (mAP) [6, 1]. To understand this measure, we first need to understand precision and recall. Precision is the ratio of correctly predicted objects (true positives) and all predicted objects (true positives + false positives).

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

Recall is the ratio of correctly predicted objects (true positives) and the ground truth objects (true positives + false negatives).

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

A threshold for the IoU between the prediction and ground truth determines if the prediction is correct. The precision and recall are used to create a precision-recall curve, which shows the tradeoff between the two for different thresholds. The Average Precision (AP) for a class is the same as the Area under the Curve of a precision-recall curve. The mAP is the mean of the AP for all classes.

References

[1] Mark Everingham et al. “The pascal visual object classes (voc) challenge”. In: *International journal of computer vision* 88 (2010), pp. 303–338.

[2] Elizabeth Goodspeed. [Online; accessed July 28, 2024; Licensed under Creative Commons Attribution-Share Alike 4.0 International license, visit `https://creativecommons.org/licenses/by-sa/4.0/deed.en`]. URL: `https://commons.wikimedia.org/wiki/File:Perceptron_example.svg`.

[3] Glenn Jocher. *YOLOv5 by Ultralytics*. Version 7.0. 2020. DOI: `10.5281/zenodo.3908559`. URL: `https://github.com/ultralytics/yolov5`.

[4] Charlie Murphy, Patrick Gray, and Gordon Stewart. “Verified perceptron convergence theorem”. In: *Proceedings of the 1st ACM SIGPLAN International Workshop on Machine Learning and Programming Languages*. 2017, pp. 43–50.

[5] Joseph Redmon et al. “You Only Look Once: Unified, Real-Time Object Detection”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2016.

[6] Olga Russakovsky et al. “Imagenet large scale visual recognition challenge”. In: *International journal of computer vision* 115 (2015), pp. 211–252.