DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Thesis type (Bachelor's Thesis in Informatics, Master's Thesis in Robotics, . . .)

Thesis title

Author

DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Thesis type (Bachelor's Thesis in Informatics, Master's Thesis in Robotics, ...)

Thesis title

Titel der Abschlussarbeit

Author:	Author
Supervisor:	Supervisor
Advisor:	Advisor
Submission Date:	Submission date

Abstract

Contents

1 Introduction

Machine Learning and Computer Vision has gained much importance in the last years. It can be used in many application fields, such as smart plant monitoring. In this context, it can be used to simplify certain workflows. One example is the hail damage detection of sugar beets.

The idea is to develop a system or application which automatically predicts the damage of plants. This has several advantages, for example less time has to be spent analyzing the fields and possible more accurate results can be achieved.

More concrete, a mobile application was developed, which allows to take images of sugar beet plants. The fotos taken should then be preprocessed and sent to a backend server which analyzes the images and predicts the damage.

In the context of this report, the preprocessing step of such a mobile application is presented. The general idea of the preprocessing step in this application is to standardize the image format of the fotos taken. This means that factors like the number of sugar beet plants, the angle in which the foto was taken and many more things are not often optimal for the model that is predicting the damage. In general, the images that we use for training are taken from directly above the plant in a 90 degrees angle. In the best case, the image only contains one plant which is directly in the center. To be therefore consistent with training images, the new pictures of the plants should ideally be in the same format with similar settings. Therefore, the preprocessing step helps to get better results by standardization of the images.

2 Theoretical background

There are different types of machine learning algorithms which have various application cases. In this case, an algorithm which performs object detection is used. In the following, the open source architecture YOLO (you only look once) is presented. First, the general principle of object detection is introduced, followed by the description of the architecture of deep YOLO networks. The principle of object detection is to combine Classification with detecting the boundaries of class instances which is exactly what needs to be done in the preprocessing step of sugar beet plants. By detecting the object boundaries of those plants, the image can be cropped to the right size which results in a standardized image format for the following regression task predicting the value of the damage (between 0 and 1).

The state of the art architecture for object detection is YOLO which was first introduced by [1]. Since then, two improved versions were published by the original author. These architectures were called YOLOv2 [2] and YOLOv3 [3]. After that, other authors published the next Version YOLOv4 [4] and also a newest version YOLOv5 can be found.

The advantage of YOLO-networks is that they are much faster than other object detection algorithms like for example R-CNNs [5] which let the model run on a very high number of regions in the image. The idea of YOLO is that the network runs once for the complete image, predicting the bounding boxes and corresponding classes at once. It finds regions in the image which are assigned predicted bounding boxes and probabilities. In the following, the general idea of YOLO is introduced, followed by the incremental improvements in the following versions.

2.1 General architecture of YOLO

As proposed in the first introduction to YOLO in [1], the network is designed to find objects and the corresponding boxes in a global manner meaning that the image is processed as whole once. Therefore, a $S \times S$ grid divides the image into smaller parts. Each of those cells is responsible to predict B bounding boxes and corresponding confidences. This confidence is defined as $Pr(Object) * IOU^{truth}_{pred}$. The bounding boxes consist of five predictions each, x, y, w, h and the confidence. The first four numbers represent the bounding box with the coordinates of the center (x, y) and the width and height (both relative to the whole image). Additionally, the cell predicts $Pr(Class_i|Object)$, which are the conditional class probabilities. Only one set of class probabilities are predicted per cell. All in all, the the class probabilities and box confidence predictions are multiplied resulting in the class-specific confidence scores for each box: $Pr(Class_i|Object) * Pr(Object) * IOU^{truth}_{pred} = Pr(Class_i) * IOU^{truth}_{pred}$.

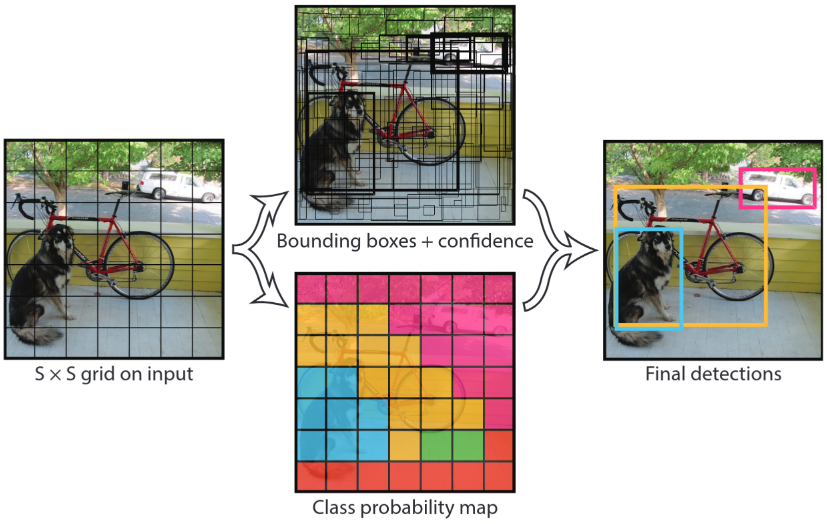

Figure 2.1: Description of idea of YOLO. Taken from [1]

All in all, a tensor of size $S \times S \times (B * 5 + C)$ results. Figure 2.1 depicts the idea of predicting bounding boxes with corresponding classes.

To achieve this, a convolutional neural network is used in combination with fully connected layers. The first part extracts features while the latter part predicts the probabilities and coordinates of the bounding boxes. It consists of 24 convolutional layer and 2 fully connected ones. The concrete network can be seen in figure 2.2.

All in all, the first network has comparable accuracies as other object detection algorithms such as R-CNN or Fast R-CNN. The most important advantage of this architecture is the real time capability. Especially for this preprocessing task, the time plays an important role.

2.1.1 Improvements in latest versions

The first architecture was improved incrementally in different versions. The original authors were involved until version 3 ([1]–[3]). After that, other authors published

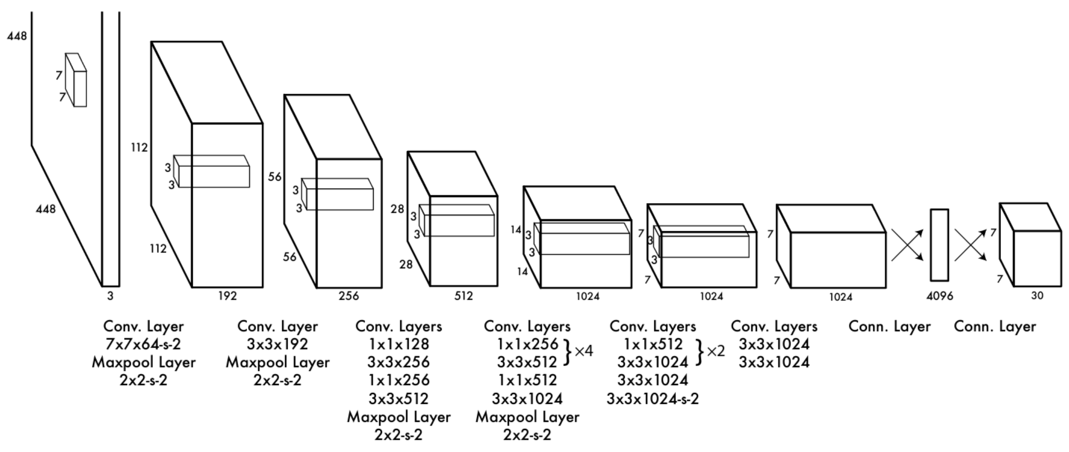

Figure 2.2: Description of network of YOLO. Taken from [1]

improvements as YOLOv4 [4]. There is no publication to YOLOv5 yet, although there is already an implementation [6].

With the first improvement which is published in [2], the problem of the localization errors of the boundaries of objects are addressed. The focus is laid on recall and localization while still keeping high classification accuracy.

YOLOv3 improvements

YOLOv4 improvements

YOLOv5 improvements

3 Methodology

3.1 Dataset

3.2 Training

3.3 Evaluation

4 Results

5 Conclusion and Outlook

Bibliography

[1] J. Redmon, S. K. Divvala, R. B. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," *CoRR*, vol. abs/1506.02640, 2015. arXiv: `1506.02640`.

[2] J. Redmon and A. Farhadi, "YOLO9000: better, faster, stronger," *CoRR*, vol. abs/1612.08242, 2016. arXiv: `1612.08242`.

[3] ——, "Yolov3: An incremental improvement," *CoRR*, vol. abs/1804.02767, 2018. arXiv: `1804.02767`.

[4] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "Yolov4: Optimal speed and accuracy of object detection," *CoRR*, vol. abs/2004.10934, 2020. arXiv: `2004.10934`.

[5] R. B. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," *CoRR*, vol. abs/1311.2524, 2013. arXiv: `1311.2524`.

[6] *Yolov5 github repository*, `https://github.com/ultralytics/yolov5`, Accessed: 2022-06-22.