List of Abbreviations

Introduction

Data and Preprocessing

Methods

Results and Discussion

Further analysis

Conclusion

## Data and Preprocessing

The most important data for object detection is the pictures that contain the objects that should be identified. In this project the object of interest is a weigher located in a chicken pen. Pictures of from chicken pens taken from above was made available at the beginning of this project. The data contained pictures from 10 different farmhouses and took up around 1 TB of storage. For the sake of keeping the training times of the models within reasonable limits this project chose to use data from only 1 of the 10 farmhouses.

Along with the pictures bounding box coordinates was supplied for some of the images. These bounding box coordinates were stored in a python dictionary where with a timestamp for every time the weigher location changed in the pictures. The first part of preprocessing was to make a dataframe with timestamps from all photos and a weigher location.

To ensure that the coordinates for the bounding box was correctly paired up with the filenames in the dataframe a small video was made. A small script with some tools from OpenCV was used to make the movie, by fetching the photo from each line in the dataframe, drawing the bounding box on it and then finally converting those images to a movie. This allowed a quick inspection of a large number of photos.

When using the YOLOv5 models there is a service called RoboFlow that can be used for data preparation. The pictures that one wishes to use for a YOLO model is uploaded to the RoboFlow website. Annotations can also be uploaded but RoboFlow also has an annotation tool. Finally, pictures are scaled to the right format and picture augmentation can also be performed. After going through the listed steps, the dataset can be exported, and the user can choose how much of the data should be used for training, validation and testing. The data can then be downloaded with pictures in the right format and train, validation and test data in separate folders and a data.yaml file containing the paths to the three parts of the dataset.

RoboFlow accepts most annotation formats but as a dataframe with filenames and bounding box coordinates is not one of them, it was necessary to convert the dataframe to a commonly accepted annotation format. COCO JSON was chosen as the annotation format as it is human readable and has a fairly simple structure and thus would be easy to make with a small script.

## Baseline model

The first model that was implemented was the baseline model. This model represented a quick way of getting started with the data and object detection. The baseline model is also, as the name suggest, used to set a baseline performance which the more advanced model can be measured against.

The baseline model was made using OpenCV, which library available for Python, which is widely used for image analysis.

In OpenCV there is several tools, which could be used to detect the weigher in the images for this project.

The first approach taken to make a baseline model was a Haar-cascade classifier. The problem with the Haar classifier is, that it detects features within the object that is being identified by finding areas where pixel intensity change. This is a problem in this case because the weigher is a smooth flat surface and thus has almost no features at all.

The second approach was Hough transformation. Hough transformation finds perfect circles, and thus seemed like a good use for this case as the weigher is perfectly round. The function has some parameters that can be tweaked, e.g., how round the object must be. Still, it didn’t prove very effective at solving the problem at hand.

The third option was blob-detection which, simply finds “blobs” that live up to some criteria that can be defined by the model parameters. This is things such as size, circularity and convexity. Blob detection creates series of binary images with different thresholds and count how many times each blob is detected. If the number of times a blob is detected is higher than a set limit it is saved. Blob detection did manage to do some detection but in the end was not very successful either.

The last attempt at making a baseline model was contour detection. This is the same function that blob detection is based on, but it has fewer parameters.

Contour detection proved to be the most effective. The detection rate is not super high, but the false positive rate is on the other hand very low.

## YOLO Model

You Only Look Once (YOLO) is a family of object detection models based on convolutional neural networks. The first YOLO model was introduced in 2016 and since then several iterations of the models have been released. In the latest release (YOLOv5) there are four models to choose from YOLOv5s, YOLOv5m, YOLOv5l and YOLOv5x, where the s, m, l and x is short for small, medium, large and x-large and refers to the size and complexity of the neural network.

The main focus of the YOLO models is to be “light weight”, which means that they should be relatively fast to train and also be quick when using the trained model for detection. This is indeed true of the YOLO models which can perform object detection at up to 150 fps, which is more than enough to do live tracking on film and videos.

In this project the YOLOv5s and the YOLOv5x, which is the least and most complex models of the YOLOv5 family, is compared to the baseline model. This comparison will highlight how a simple model compares to the more advanced neural network-based models and how the complexity of the advanced models affects their performance.

The YOLO package comes with pretrained models (on coco dataset?) that can detect common objects. The object detection task of this project, which is finding a weigher among chickens in a pen is not

YOLOv5s

The YOLOv5s is the least complex of the four convolutional neural network models in the YOLOv5 suite. Despite it being the smallest model, it still has 270 layers and more than 7 million parameters. This means that even this “small” model is of a size that would be very difficult and time consuming to create from scratch.

YOLOv5x

## Abstract

Describe in few words what you have done. Should correspond well with the conclusions made in the last section.

## List of Abbreviations

NN – Neural Network

CNN – Convolutional Neural Network

mAP – Mean Average Precision

## Introduction

Describe the problem and why it is relevant. Describe object detection. Describe the relevant theory for the models. Maybe write a couple of lines about Scio+.

## Data and Preprocessing

### Data

Et billede, der indeholder indendørs, plante

Automatisk genereret beskrivelseData is a crucial element of any machine learning task, and it can have great effects on the quality of the predictions from the final model. In this project the machine learning task at hand is object detection where images are the data. The images in this project are infrared photos from chicken pens taken directly from above as shown in Figure 1.

Figure 1 – Example of one of the infrared pictures from the dataset. The chickens are seen standing around the feeders. The round weigher that this project seeks to detect is seen in the middle of the picture.

The aim of this project is to build an object detection model that is able to find the weigher that is seen in the middle of Figure 1. The model should return the location coordinates as well as the width and height of the weigher as output.

The pictures that were used for this project was supplied by Scio+ and was taken at a chicken farm in Japan that breeds grandparent chickens. In the supplied data there were pictures from 10 different pens and form XX days from each pen. This resulted in more than 500.000 pictures taking up more than 1 TB of storage.

Along with the images, the supplied data also contained annotations for some of the pictures, which could then be used as training data. The annotations were stored in a Python dictionary and contained timestamps for every time the location of the weigher changed and coordinates for the location as well as height and width of the bounding box.

### Preprocessing

Different machine learning models require different preprocessing steps, however one good starting point for the preprocessing for all the models was to convert the annotations dictionary to a dataframe. The goal with the dataframe is to have a location of the weigher for each photo and not only for when it changes. In the resulting dataframe each line represents one image with some relevant information including weigher location and width and height of the bounding box.

To ensure that the coordinates for the bounding box was correctly paired up with the filenames in the dataframe a small video was made. A small script with some tools from OpenCV (Viola & Jones, 2001) was used to make the video, by fetching the photo from each line in the dataframe, drawing the bounding box on it and then finally converting those images to a video. This allowed a quick inspection of a large number of photos, where mistakes of bounding box locations were readily visible.

The first model that was tried out, was a Haar Cascade Classifier implemented in the OpenCV library (Bradski, 2000). The relatively high resolution of the images (1280x720) was scaled down for this model to 256x144 pixels. The images were also converted to grayscale. Although the images were captured without color a conversion to gray scale was still necessary for the model to accept the photos as input. Both of these preprocessing steps were performed in OpenCV.

Other approaches for the baseline model included Hough transformation, blob-detection and contour detection, all of which are a part of the OpenCV package. The preprocessing required for these models were limited to conversion to binary images which was easily done in OpenCV.

The Convolutional Neural Network (CNN) models that was used trained during this project comes from the YOLOv5 package. These are pre-build models i.e., the architecture of the model has already been built and optimized for object detection, such that all that is left to do is to train the models.

To make streamline preprocessing for the YOLO models the people behind YOLO has developed RoboFlow, which is a web-based tool for preprocessing. In RoboFlow users can upload photos and annotations. If annotations are not available, RoboFlow has an annotation tool. As a last step the input images can be resized, and augmentations can be applied.

For this project the free tier of RoboFlow was used, which restricts the number of images that can be uploaded to 10,000. 10,000 images are just a fraction of the images that were available, but as training of the CNN models is quite time consuming, training on more than 10,000 images would not be possible within the time frame of this project.

A total of 8,500 images from this project was uploaded to RoboFlow. These were resized to 640x640 pixels which is the default input size for RoboFlow. To maintain the aspect ratio of the original images, black edges were added to the top and bottom, which was also done in RoboFlow. No augmentation was applied to the images.

After completing the above steps, the data can be downloaded and split into training, validation, and test sets. This split was done with 70%, 20% and 10% for training, validation and testing data respectively.

The data that is downloaded from RoboFlow is separated in different folders for the validation, train and test data with a YAML-file storing the paths to these folders. This way the YAML-file can be passed as an argument when running the YOLO models and then the model will use the relevant data automatically.

The annotations for the images that is uploaded to RoboFlow must be in a common annotation format. As previously mentioned, though, the annotations for this project were stored in a dataframe. Thus, the information from the annotations dataframe needed to be converted to a common annotation format. COCO JSON was chosen as the annotation format as it is human readable and has a fairly simple structure, which made it easy to make a small script that could perform the conversion.

Preprocessing

Haahr: finding negatives. Converting to grayscale and resizing.

## Methods

Describe all the work that has been carried out and the reasoning behind it

## Results and Discussion

Show all results, tables and figures and describe them without any conclusions

## Further analysis

Describe how further work on the model and dataset could be carried out and what advantages it may bring. E.g., using more data will give better generalization to other farms.

## Conclusion

Describe what you have found.

## References

Bradski, G. (2000). The OpenCV Library. *Dr. Dobb’s Journal of Software Tools*.

Viola, P., & Jones, M. (2001, 8-14 Dec. 2001). *Rapid object detection using a boosted cascade of simple features.* Paper presented at the Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001.