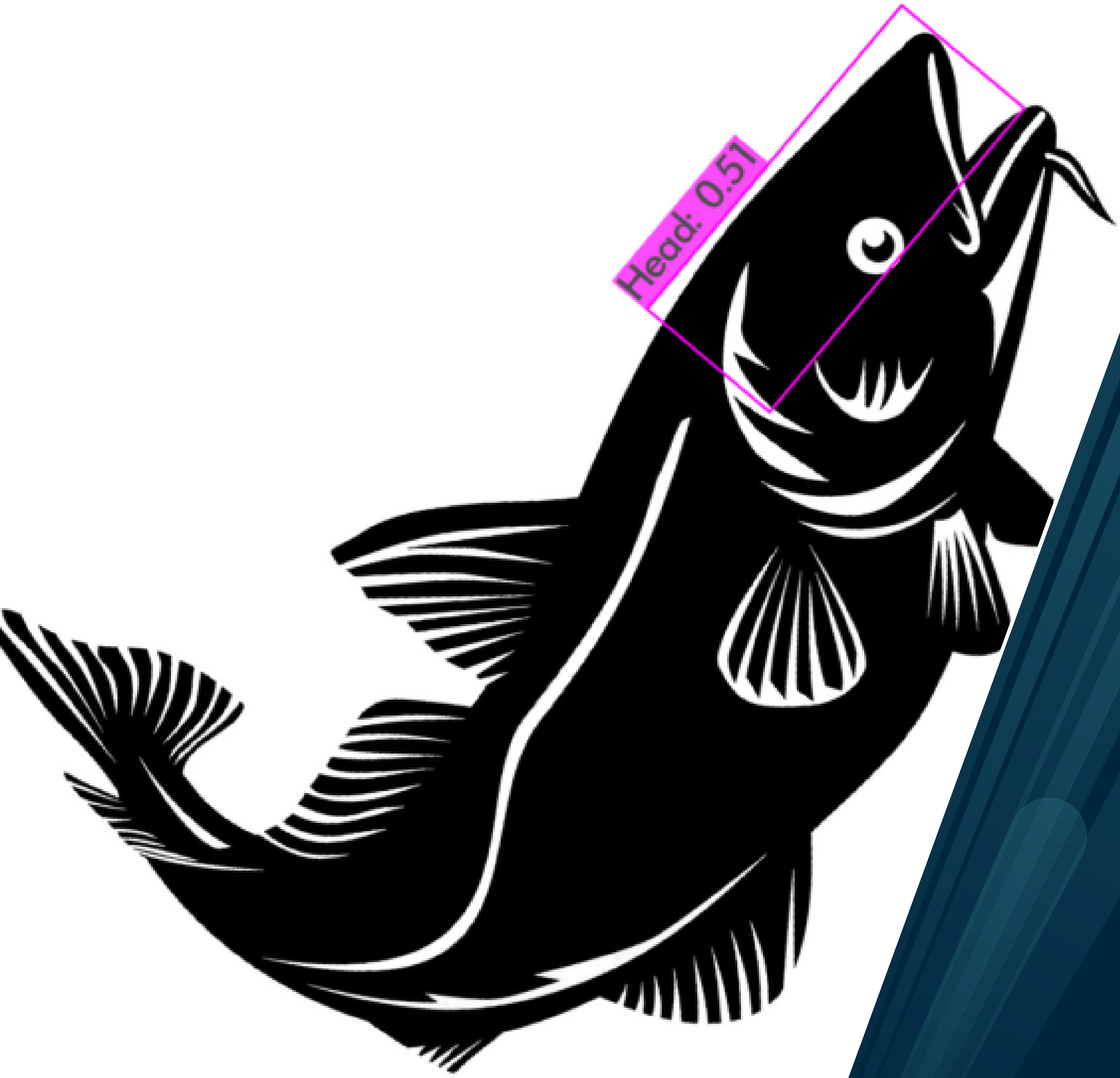
Faculty of Science and Technology
Department of Physics and Technology

Towards automation in the fish processing industry using machine learning.

Jostein Henriksen

FYS-3941 Master thesis in applied physics and mathematics – 30 SP – Mar.2023



To my fellow students and Associate Professor Puneet Sharma (my supervisor) at UiT The Arctic University of Norway, Havfront AS (collaborating company), Norges Råfisklag, my familiy and friends.

Thank you so much for all your support!

“

In the fish processing industry,
Where time is money and speed is key,
The need for innovation and efficiency,
Led to the rise of computer vision’s proficiency.

Machines equipped with cameras and sensors,
Analyze fish with precision and candor,
Sorting by size, species, and quality,
Reducing labor and boosting productivity.

No longer do workers need to strain their eyes,
Or rely on their experience to realize,
Which fish to keep, which fish to discard,
Computer vision makes the task less hard.

The machines work tirelessly day and night,
Ensuring that the process runs just right,
Sorting fish into categories so neat,
Making sure the products are always top-grade and complete.

With computer vision on their side,
The fish processing industry takes in stride,
The challenges that come with each catch,
Ensuring that their products meet every batch.

”

– Generated by ChatGPT from the text: ’write a poem about using computer vision in the fish processing industry’ https://chat.openai.com/chat

Abstract

This master project was inspired by challenges faced by commercial fisheries in the north of Norway of controlling food quality and food safety. In this thesis, four different *ML* models' ability to do object and keypoint detection on specific anatomy parts of fish, has been studied. With the aim of recommending a suitable model to be part of a *CV* system for an industrial fish gutting machine that cuts open the fish belly between the pelvic fins and the anus. Requirement that the rotating knife shall not cut into the flesh behind the anus opening, and cut should end (or start) maximum 5 millimeters from the anus opening. Likewise, at the pelvic fins, the cut shall start (or end) 15 millimeters from target along the centerline of the fish, and a sideways offset of roughly ±5 millimeters can be acceptable, depending on the length of the fish.

The experiments were performed with two *YOLOv7* and two *Detectron2* models, *YOLOv7* for object detection with bounding boxes, and *Detectron2* for keypoint detections. The results showed that only one of the *Detectron2* models was able to do keypoint detection repeatedly, but the achieved accuracy was not good enough. Both the *YOLOv7* models were able to meet the cut length requirements and both got recommended for use in the suggested *CV* solution.

More work still remains before one of the *YOLOv7* models can be taken in use, such as determining the object detection speed, finding a suitable embedded computer with GPU to run the *CV* system on, determining the best way of communication between the PLC in Folla and the *CV* system and finding a suitable location for a camera inside the Folla machine.

Contents

List of Figures

List of Tables

Glossary

data annotation is the technique of marking specific regions of data so to supervise a *ML* model to learn to recognize the different regions. Each type of region need to be labelled with some meaningful information, see *data labels*.

aquaculture in the context of this project, is fish farming in Norwegian coastal areas, ref. fig.1a in [1].

ChatGPT is a language ML model trained to produce text, optimized for human dialogue. Developed and deployed by the AI research and deployment company OpenAI..

Cod Cluster is a national and state-funded innovation cluster operated by Innovation Norway, Siva and the Research Council of Norway.

COCO is an abbreviation for Common Objects in Context. It is a large-scale object detection, segmentation, and captioning dataset from Microsoft.

COCO-annotator is a web-based image annotation tool that exports annotations in the well-known COCO format.

COCO-format is a data format made for the COCO dataset. The annotations are stored using JSON, structured in the key-value pairs {"info": info, "images": [image], "annotations": [annotation], "licenses": [license]}, with associated subkey-subvalue pairs such as {"bbox": [x,y,width,height]} and {"keypoints": [x1,y1,v1,...]}.

CV is an acronym for Computer Vision. *IBM*[2] defines *CV* as: "Computer vision is a field of artificial intelligence (AI) that enables computers and systems to derive meaningful information from digital images, videos and other visual inputs — and take actions or make recommendations based on that information. If AI enables computers to think, computer vision enables them to see, observe and understand."

CVAT is an acronym for Computer Vision Annotation Tool. It is an interactive video and image annotation tool for computer vision.

Det2-1cls-6kpts is a shorthand notation of "Detectron2 model of 1 class with 6 keypoints", a custom model used in this project.

Det2-3cls-2kpts is a shorthand notation of "Detectron2 model of 3 classes with 2 keypoints each", a custom model used in this project.

Det2-4cls-2kpts is a shorthand notation of "Detectron2 model of 4 classes with 2 keypoints each", a custom model used in this project.

data labels is in this context, meaningful information about key features present in different types of data for training a supervised *ML* model. Every *data annotation* need to be assigned a data label.

Detectron2 is Facebook AI Research's next generation library that provides state-of-the-art detection and segmentation algorithms.

FHF (Norwegian Seafood Research Fund) is a state-owned limited company owned by the Ministry of Trade, industry and fisheries, and financed by the industry through a levy on exports of Norwegian Seafood at 0,3 %. FHFs goal is to create added value to the seafood industry through industry-based research and development (R&D).

fisheries in the context of this thesis, fisheries is the professional fishing conducted by fishing vessels in the ocean around the Norwegian coast, ref. fig.1b in [1].

fish wholesale is the sale of fish in large quantities to retailers.

fish grading is sorting the fish on size, before and/or during processing.

gutting is the process of cutting open the fish' belly in order to remove it's intestines.

JSON is an acronym for JavaScript Object Notation. It is a lightweight data-interchange text format that is completely independent of the programming language. It is easy for humans to read and write and it is easy for computers to parse and generate. The structure is built up by a collection of name/value pairs and ordered lists of values.

fish processing is defined here as the whole (industrial) process, that starts immediately after the catch, of making high quality food products of the fish.

hydraulics is a power system that uses pressurized hydraulic oil to transmit power. In such systems, hydraulic oil is pressurized by a pump and then transmitted through hoses and pipes to actuators, which can be cylinders, rotating motors, or other types of devices, to produce linear or rotational motion.

IMR is Norway's Institute of Marine Research, one of the biggest marine research institutes in Europe, with about 1,100 employees. Through its research and advice, the IMR seeks to help society to continue exploiting the valuable assets in the sea sustainably. IMR is a neutral knowledge provider, and publicise the research results both in Norway and internationally.

MD is an acronym for Machine Directive. MD is a European Union directive with mandatory specifications in health and safety combined with harmonized standards that machines must comply with.

ML is an acronym for Machine Learning. *IBM*[3] defines *ML* as: "Machine learning is a branch of AI and computer science which focuses on the use of data and algorithms to imitate the way that humans learn, gradually improving its accuracy.".

pelvic fins also called ventral fins, are a set of paired fins located ventrally in the frontal part of the fish abdomen..

pneumatics is a power system that uses pressurized air to transmit power. In such systems, air is pressurized by a compressor and then transmitted through hoses and pipes to actuators, which can be cylinders, rotating motors, or other types of devices, to produce linear or rotational motion.

salmonids is a common name for fish from the Salmonidae family, such as salmon, trout, arctic char. Their flesh is orange/reddish in color, and have a more fatty or oily consistence than the flesh of white fish.

Springfield is the codename for the GPU cluster owned and operated by the UiT Machine Learning Group.

white fish is a fisheries term for species of fish having white or light-coloured flesh, including Atlantic cod (Gadus morhua), whiting (Merluccius bilinearis), haddock (Melanogrammus aeglefinus), hake (Urophycis), pollock (Pollachius), and others.

YOLO is an acronym for You Only Look Once, It is a state-of-the-art, real-time object detection system. There are several official versions of YOLO, but in this project only YOLO version 7 (YOLOv7) has been used, so in this thesis every YOLO reference is to YOLOv7, unless otherwise stated.

List of Abbreviations

AI Artificial Intelligence.
CNN Convolutional Neural Network.
GT Ground Truth.
LF Lower Farside keypoint.
LO Lower Opside keypoint.
OLS Ordinary Least Squares.
RoI Region of Interest.
RoIs Regions of Interest.
UiT UiT The Arctic University of Norway.
UF Upper Farside keypoint.
UO Upper Opside keypoint.
WSL2 Windows Subsystem for Linux, version 2.
YOLO You Only Look Once.
YOLOv7 YOLO version 7.
YOLOv7 std. YOLO version 7, standard model.
YOLOv7 tiny YOLO version 7, tiny model.

Part I

Introduction

/1 Introduction

1.1 About the Norwegian fish processing industry

In Norway, fish is one of the top economic export articles. According to a SINTEF article about the Norwegian seafood industry[1], Norway was the world's second largest exporter of fish and seafood in 2017, with NOK 94.5 billion in export value. That constitutes for 7.9% of the total national export revenue. The authors group the seafood value chain into two sub-chains by fish raw materials, *fisheries* and *aquaculture*, both further divided into *fish processing* and *fish wholesale*. Workspace of this thesis is within *fish processing* in general.

In the SINTEF article, they estimate a total number of 58,123 employees in the total seafood value chain for 2017, an increase of 35.6% from 2004. However, looking specifically at the share of employees for fish processing, it has dropped from 20% to 6% within the aquaculture value chain, while for fisheries it has increased from 21.5% to 27.8%.[i]The authors claim that automation and new technology is the reason for increased productivity and hence the strong reduction of employees within aquaculture. The small increase of employees in fisheries is explained due to a stable relationship between the different players and an increased demand for seafood in the market. This could imply that the fisheries processing facilities are and have been highly automated for a long time, or that new technology to a lesser extent have been implemented here. Another reason is that a usual catch delivered by a fishing vessel along the coast of Norway, consists of several fish species in total, each of varying sizes. In fisheries, the fish processing facilities need to handle all that variation while still

i. FTE numbers from table 6 and 7 in [1].

maintaining efficiency and quality. One of the major challenges within the fish processing industry is that the machines are not able to handle fish of different sizes and shapes in an effective manner. Fish of odd sizes or low volumes, that are not handled well in the machines will have to be manually processed by the employees. In comparison, with *salmonids* in aquaculture, fish shape and size are more consistent and hence it is easier to automate the associated machines as compared to other fish, like for instance cod. That makes it more convenient and less costly to automate and scale the production lines.

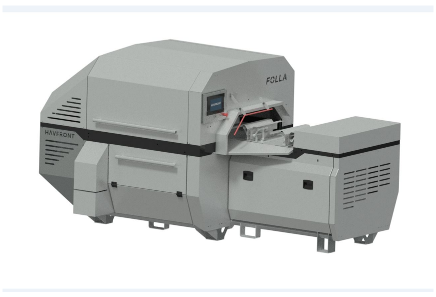
1.2 Machine Learning and Computer Vision in the Norwegian fish industry

ML and *CV* are nowadays finding ever more use cases in the fish industry. For aquaculture, there exists equipment for fish ID control, fish health control, fish lice detection and removal[4], food control and nets inspection, among others. In fisheries onboard boats, there can be equipment like multifrequency echosounders/sonars combined with *ML* for prediction of fish species and sizes, catch control using *CV* for active segregation of fish at the trawl inlet[5]. In fisheries processing facilities, *CV* have been taken in use for *fish grading* and sorting fish, in quality control for estimating level of trapped blood in the fish meat or locating remaining bones in the fish fillets, among others.

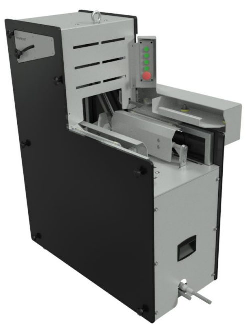
In this project, we are trying to utilize *ML* and *CV* for automatic detection of specific anatomic parts of the fish in order for the machine to automatically adjust its parameters for each individual fish. To the best of our knowledge this is the first project with this scope.

1.3 Company Havfront – Manufacturer of fish processing machines

The collaborating company of this master thesis, Havfront AS, is manufacturing machines for cutting and *gutting* fish. At the moment, Havfront is focusing on the fish processing market in *fisheries* only. Founded 10 years ago and having sales of NOK 10 million (2021), Havfront is a relatively small and young company. Currently, they have two different machines in production, Loppa and Folla. Both machines are designed for performing gutting and head removal, the main difference is in size and technology. Loppa[ii]is small in size, intended for smaller boats, handles up to 20 fish per minute of size 1 to 12 kilos, runs on 24VDC + *hydraulics* and adjusting cutting parameters is done manually. Folla[iii] is much larger, intended for onshore processing facilities, handles 18 to 23 fish per minute of size 1 to 20 kilos, runs on 230/440VAC + *pneumatics* + fresh water supply, and has automatic adjustment of the cutting parameters. This master thesis is a part of a project of adding *CV* to Folla in order to improve accuracy of the cut for fish gutting.



(a) Folla



(b) Loppa

Figure 1.1: Illustrations of the Folla and Loppa machines. Note: The images are differently scaled, in reality Folla is considerably larger than Loppa.

1.3.1 Development funding

Havfront is member of *Cod Cluster*[6], and participating in the workgroup for fish processing where they seek to optimize fish processing in order to increase the utilization of fish and profitability. The project of adding *CV* to Folla is partly funded by *FHF (Norwegian Seafood Research Fund)*[7].

ii. https://www.havfront.no/maskiner/loppa100/
iii. https://www.havfront.no/maskiner/folla/

2 Background

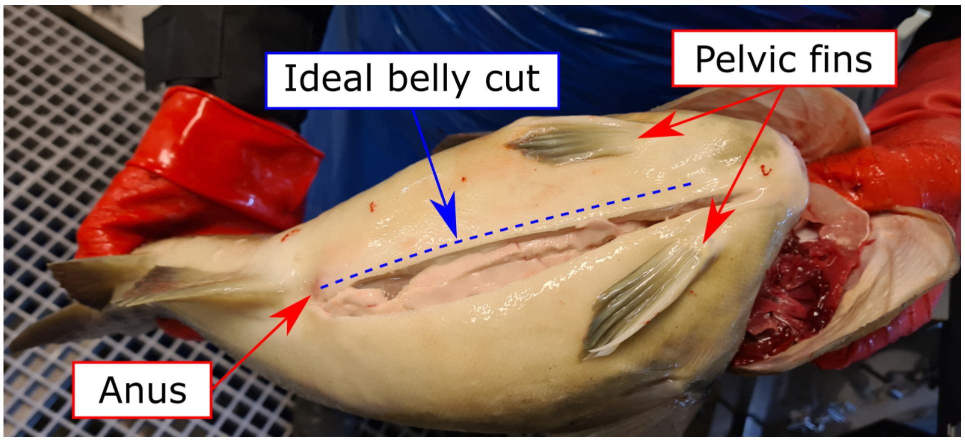


Figure 2.1: Illustration of an ideal belly cut for gutting of fish to customer's request, shown with dashed line in blue. The starting point is centered between the pelvic fins, and the cut should end at or very close to the anus opening, but must not go past the anus and cause damage the meat. Note that the actual cut, although a little side-shifted from the blue line, is still acceptable.

2.1 Problem

One of Havfront's fish farming customers wants the machine to cut open the belly only from around the midpoint between the pelvic fins to the anus, see illustration in fig. 2.1. Originally, Folla is designed for cutting open the belly from the throat to the anus, using a mechanical depth pin (located immediately below the rotating knife) to end the cut at the correct position, but for this specialized belly cut where the knife has to start cutting from in between the anus and pelvic fins, shown in fig. 2.2, the depth pin need to be removed. Then, the machine does not have a robust method to determine the cutting length, and a number of belly cuts gets too short or, even worse, goes beyond the anus opening and cuts into the flesh. The latter reduces the food quality and shelf life. Havfront's experience is that 5-10% of the belly cuts are either too long or too short, and they want to improve the precision in order to lower the number of unacceptable cuts. Hence, the challenge is to come up with a better solution to estimate the starting and stopping points for gutting each individual fish that enters the Folla machine.

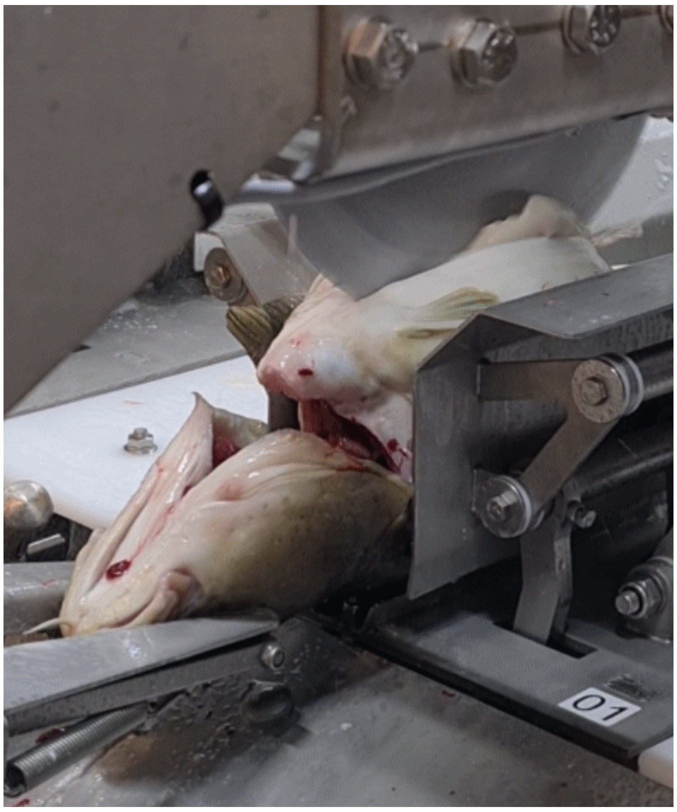


Figure 2.2: Showing gutting process in Folla, the rotating knife opening the fish' belly.

2.2 Desired outcome

Besides improved cutting precision, there is also a wish for increased efficiency, i.e. that the proposed solution would also open for a higher processing rate (more fish per minute) than of today. The two Havfront machines are being used both onboard small and large fishing vessels, and at onshore processing facilities, for both wild catch and fish farming. At many locations they have their own specification for how the fish should be gutted. In addition to the specific problem mentioned above, some want the cut at an angle, some need different cut length for different fish species, and yet another wants a side-shifted cut because a machine further down the processing line then gives better output. Havfront has not done any cost-benefit calculations specifically for a precision improving solution, as their customers are not sharing how much more they get paid for better quality fish. Only by being out in the market, discussing with customers and "wrestling" with competing machine builders, one gets a sense of what they are willing to pay for efficient and precise machinery. But being able to offer higher precision and speed as options, will give Havfront an advantage towards competitors and make Folla even more attractive for demanding customers.

The aim for the cutting length precision is that the cut should end (or start) as close to the anus opening as possible, and not extend past it. Havfront's customers have not dictated a strict set of limits, but Havfront would like to get the cut within 5 millimeters from the opening. In the opposite end, at the pelvic fins the requirement is less strict. Up to 15 millimeters from target along the centerline of the fish, and roughly ±5 millimeters sideways offset can be acceptable, depending on the length of the fish.

2.3 Proposed solution

The figure 2.3, attempts to illustrate the proposed solution, which is a *Computer Vision* (*CV*) system with automatic object detection of specific anatomy parts on the fish. *CV* involves camera and a *Machine Learning* (*ML*) algorithm that have been trained to recognize the anatomy details of interest. Details about the scene, extracted from the video feed by the algorithm, will give the machine controller the information needed to precisely adjust the cutting blade to each fish individually, independent of it's size or body shape.

For this solution, there are two points on the fish that are important: 1. the center point between the pelvic fins, and 2. the anus opening. In addition, in order to make estimates of the positions and the distance between the two points on the fish, at least two fixed objects of the background (with known positions and measures) are needed for quality control and calibration of the camera.

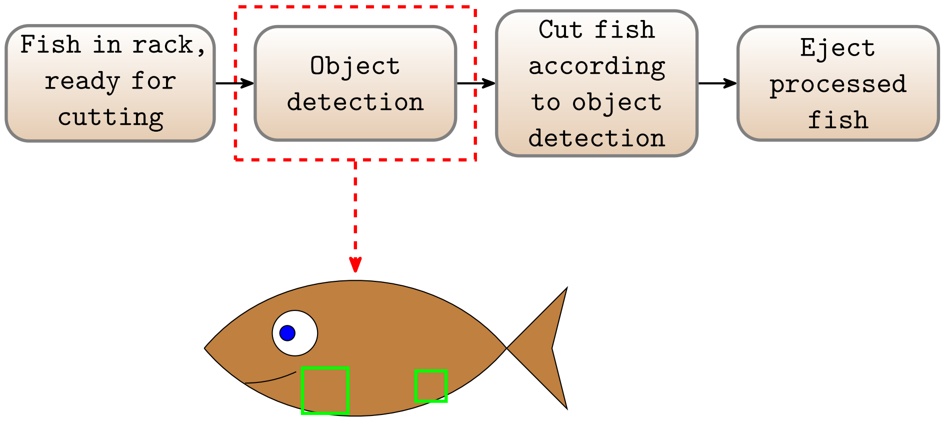

Fish in rack, ready for cutting
Object detection
Cut fish according to object detection
Eject processed fish

Figure 2.3: Illustration of the gutting process in the Folla machine. Emphasized with the dashed line in red, automated object detection with computer vision, is the proposed solution for improving the cutting process. The green rectangles on the fish figure are meant to visualize the anatomy parts of interest. The illustration originates from my feasibility study report[8].

2.4 Project overview

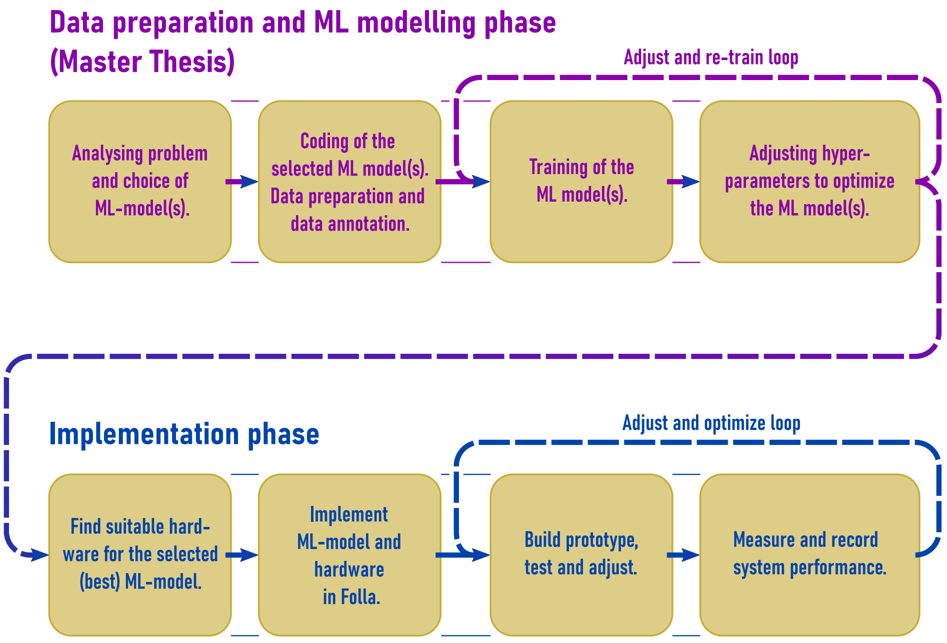

Data preparation and ML modelling phase (Master Thesis)
Adjust and re-train loop
Analysing problem and choice of ML-model(s).
Coding of the selected ML model(s). Data preparation and data annotation.
Training of the ML model(s).
Adjusting hyper-parameters to optimize the ML model(s).
Implementation phase
Adjust and optimize loop
Find suitable hard-ware for the selected (best) ML-model.
Implement ML-model and hardware in Folla.
Build prototype, test and adjust.
Measure and record system performance.

Figure 2.4: Workflow of Master Thesis / Folla Computer Vision project.

In this project, aim is to develop and train some object detection algorithms ready for further fine-tuning, testing and implementation in Folla. The plan is to develop some *ML* models for doing object and keypoint detection on details of the outer anatomy of the fish entering the Folla machine. The project workflow is illustrated in fig. 2.4. As a starting point, two state-of-the-art *ML* algorithms that can do both object and keypoint detection were selected, namely *YOLOv7*[9] and *Detectron2*[10]. A working model proposed in this project would enable further experimentation to be performed during the development, implementation and testing stage at Havfront's premises.

The experimental phase of the project starts with selection of some ML models that could be suitable for the project, followed by data preparation for the different models. The most important data preparation is annotation of the dataset, and the selected models need different sets of annotations. The annotation process is covered in chapter 3. Chapter 4 deals with the *ML* models for this project. The writing order of the models has been attempted to remain the same throughout the document. The experiments and results are found in chapter 5 and 6, respectively. The thesis is rounded off with discussion and conclusion in chapter 7 and 8. Results in the form of data tables and the models' config files are included in the appendices.

2.4.1 ChatGPT – Declaration of use in this project

Yes, *ChatGPT* has been used to solve problems during this project. Not to generate bodies of text for this report, but as an efficient tool for solving programming problems like *ML* configurations and tuning, efficient interaction with *JSON* files, Linux CLI commands and regex search patterns, Python classes, functions and plotting, LaTeXmacros and mathematical expressions. And also, not to forget, the epigraph poem.

Part II

Dataset and ML models

/3 Dataset

3.1 Dataset source

The dataset was collected (during the pre-project[8] for this Master Thesis) at Namdal Seafood, a processing facility for farmed cod. In total, the dataset consist of almost 106 minutes (11 videos or 4506 frames) with recordings of 207 fish individuals[iv]. The videos have numbered names from 17.1 to 17.6 and 18.1 to 18.5. In this project, video 17.1 was used for annotation and 18.5 for prediction.

An ordinary action camera (GoPro Hero8) was used to record fish entering the Folla machine. The camera was mounted on the sidewall looking down at fish being transported on a conveyor, lying upside down in a rack, just before entering the processing zone. Although the viewing angle and lighting condition was not ideal for getting low distortion and good contrast imaging, it was the most convenient camera position to record fish in the conveyor racks. Ideally, the camera should have been mounted inside the machine, directly over or in close vicinity of the gutting position, but space limitations and need to cover for splashing debris from the rotating knife made that an impossible option for this project. Figure 3.1 shows a typical frame from the dataset. The improper lighting condition caused poor contrast between the conveyor rack and the background panel, especially on the left side. For every fish in the dataset, for both training and test images, the ideal cutting length for gutting was measured by hand and recorded (fig. 3.2).

iv. The only fish species recorded on video was cod.

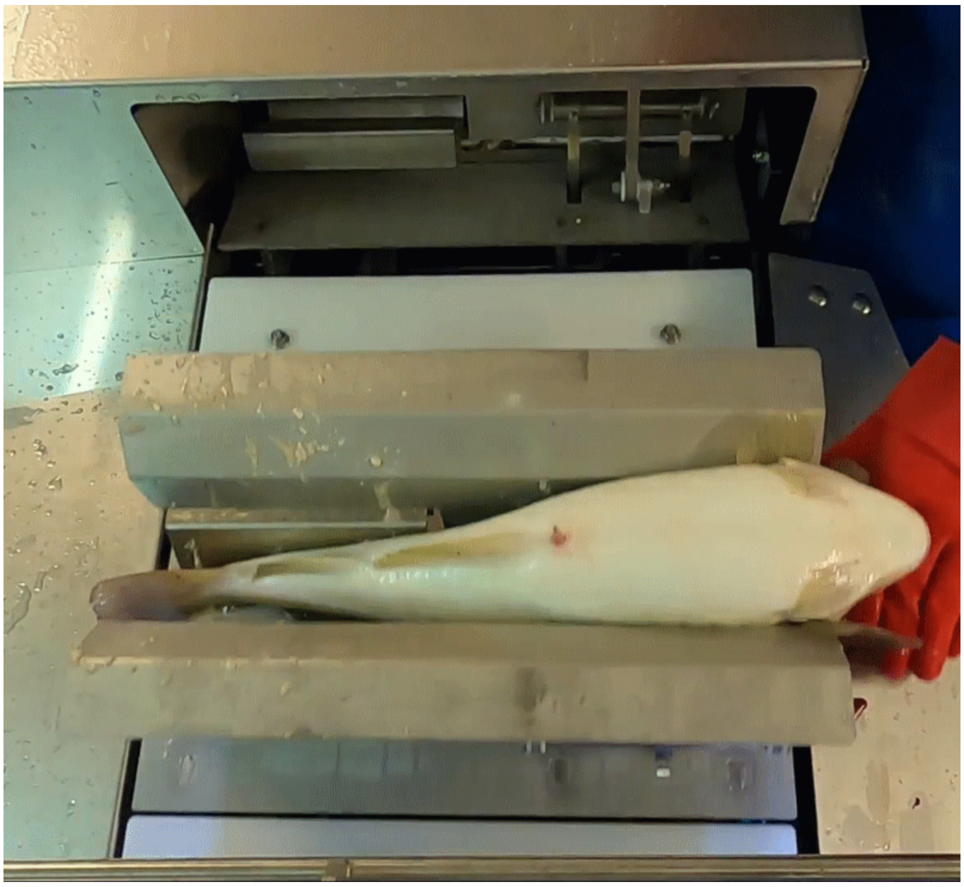


Figure 3.1: Typical image from the dataset.

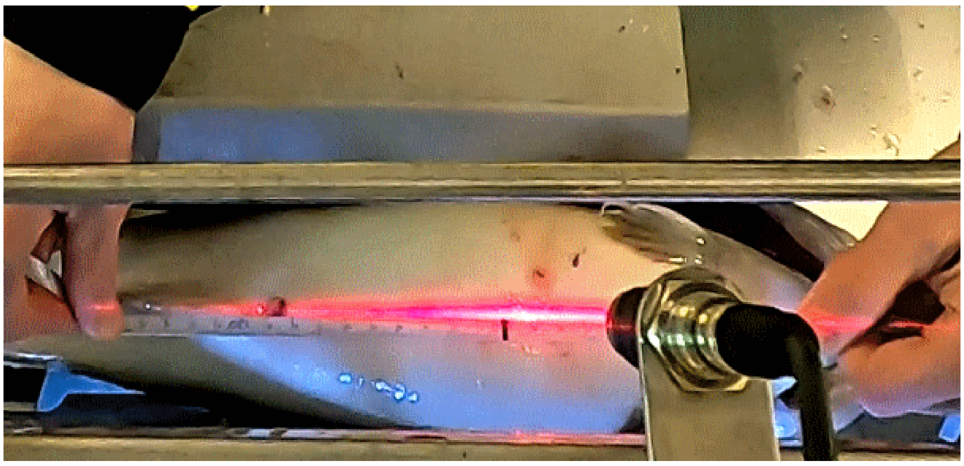


Figure 3.2: Measuring the cut length for gutting each fish individually.

3.2 Data preparation

Before annotation, all the frames capturing a fish were extracted from the dataset videos. As recorded with a wide-angle camera, a lot of unimportant surroundings were included in the scene, so the surroundings were cropped out. The final image size (in pixels) became 1178x1080 after cropping.

3.3 Objects of interest

For this particular project, the anus opening and the pelvic fins were the obvious candidates for annotations on the fish. Since plan was to estimate the ideal cutting distance, one would also need to locate the position for some fixed points on the machine, with known distance between them. The idea is that the estimated cutting distance should become more precise and invariant to an eventual change of camera. The outer edges, left and right on the two top plates of the rack, was chosen for this purpose. They are relatively close to the fish and their relative positions to the fish are more or less stationary when moving over the frame/scene. In addition, for possible future use as safety measure, the *ML* model should also be able to detect a human hand. The names for the rack keypoints describes their relative position in the image. Position 'Opside' is towards the right in the scene, where the machine *operator* is standing. Position 'Farside' is towards the left in the scene, *far* from the operator. 'Upper' is towards the top of the image, and 'Lower' is then towards the bottom of the image. Further in this thesis, the rack keypoints are denounced *Upper Farside keypoint (UF)*, *Upper Opside keypoint (UO)*, *Lower Farside keypoint (LF)* and *Lower Opside keypoint (LO)*.

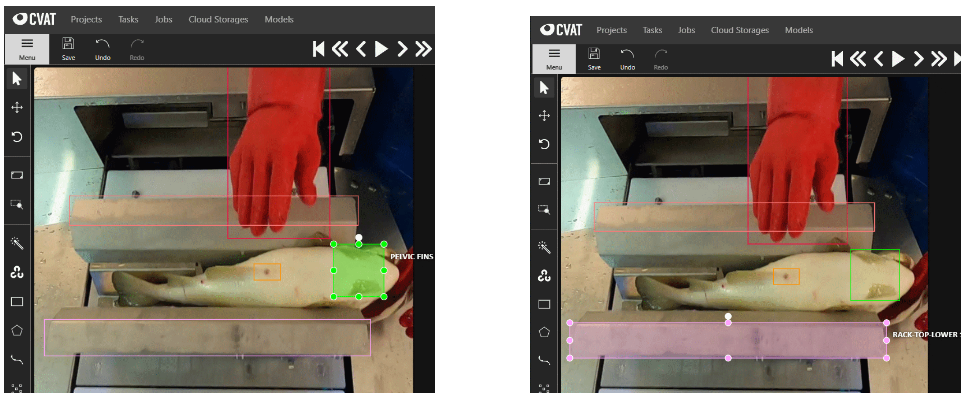
Please note that the idea behind using positional rack and fish keypoints is to use the known length of the rack to make an estimate of the real world length of the optimal cutting line between the pelvic fins and the anus of the fish.

3.3.1 Annotating training data

Out of the complete dataset, only video *17.1* with 624 images of 12 different fish were annotated for this project. The limited time available did not allow for more, as different *data annotation* types were needed for the different *ML* models. Two different annotation tools were used to label the training images, *CVAT* and *COCO-annotator*. *CVAT* was the initial choice due to simple implementation and efficient workflow, but for this project it was a little cumbersome to make keypoint annotations, where *COCO-annotator* was found to make the job more easy and in less time. Both *CVAT* and *COCO-annotator* are web-based image annotation tools, and can be run on a local computer with a Docker image.

Not all the annotation data are presented in this report. Tables given in appendix C contains annotation data and corresponding statistics from the annotations of fish number two and five from video *17.1*, respectively. These are thought to represent the annotations well in general.

Annotations for Yolo



(a) CVAT - Annotating pelvic fins on fish. **(b)** CVAT - Annotating top plate of rack.

Figure 3.3: Annotation example screenshots from CVAT.

The annotations for *YOLOv7* was done with *CVAT*. Different annotation shapes were tried, such as rectangles, polylines, polygons, keypoints. For the actual *YOLOv7* model, polylines and polygons did not work, and when trying to use keypoint annotations, it came for a day that it is not straight forward to implement a *custom keypoint model*[v] in *YOLOv7*. Searching the Internet revealed that other people has also been looking for ways of doing custom keypoint models in *YOLOv7*, but no hints/solutions/tutorials were found at the time of writing this report. *YOLOv7* can be used for human pose estimation as per *COCO* 2020 Keypoint Detection Task, detecting 17 keypoints on a human body, but to make a custom *YOLOv7* model with less number of keypoints, and for more than one class, one would need to get well under the hood of the *YOLOv7* algorithm. Since tweaking the existing *YOLOv7* code (from the source on GitHub[11]) to achieve a better estimate would require a lot of development time, an alternative approach was opted for with *Detectron2*. Thus, only rectangular bounding box annotations were made for the *YOLOv7* model. However, a possible future implementation of custom keypoint detection might not turn out be so difficult and time consuming as anticipated here.

v. Custom model here means a model that is built for a different task than the standard

For the anus opening, the bounding box should be centered on the opening, and be wide enough so that a small part of the anal fin behind be present inside the rectangle. Idea of including part of the anal fin is to prevent false detections from scattered blood stains or other debris on the fish that incorrectly could be recognized as an anus opening. For the pelvic fins, the midpoint of the bounding box should be centered on the centerline between the two fins. The ideal starting point for the cutting process should be on, or very close to the midpoint of the left edge of the rectangle. See example in figure 3.3a. The neighboring figure 3.3b shows an annotation example of the conveyor rack. In the first annotation round, only one of the long side edges of the rack tops was covered by the bounding box, but after training the model was not able to detect the rack tops, so the annotations had to be changed so that the bounding boxes were completely surrounding the rack tops.

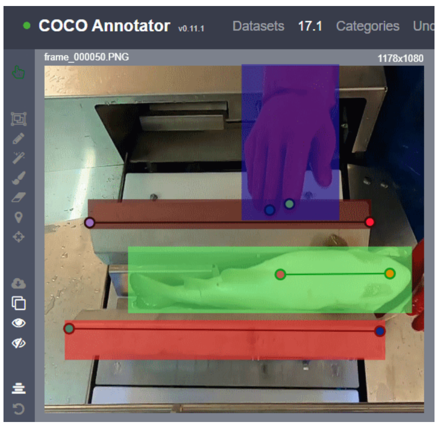
YOLOv7 has it's own annotation format, and for bounding boxes *YOLOv7* is using coordinates relative to the image size (fractional number between 0 and 1), and is referencing the box by it's center point coordinates, height and width. The *CVAT* format for a bounding box is referencing it in absolute coordinates (in pixels) by it's top left corner and bottom right corner. Yet another format is the "standard" *COCO-format*, which is using the top left corner, height and width in absolute coordinates to reference the same bounding box. Hence, it is very important to choose the correct format when exporting the final annotations. *YOLOv7* also requires the annotations in ordinary text files, one text file with annotations per image stored in a separate "labels" folder. The corresponding images must be in a neighboring "images" folder. In addition, the filenames need to be equal for both images and labels(except for the filetype suffix).

Tables C.1 and C.2 in appendix C.1 contain annotation data relevant for the suggested solution. The ideal cutting length for the actual fish is located in the rightmost column. In the neighboring column to the left, is the estimated cutting length to be compared with the ideal cutting length. The estimated cutting length is computed by dividing the annotated cutting distance [pixels] by the pixel-per-millimeter scale [pix./mm]. As the fish, from one image to another, propagates over the scene, the scale values are slowly increasing due to the perspective of the camera. The scale is computed for the centerline between the lower and upper rack tops, which is approximately also the centerline of the fish. The scale computation is done by first finding the rack top width at the centerline. This is simply done by adding the widths of lower and upper rack top and dividing by two. Although not 100% representative for the rack top width at the centerline, the deviation from the true centerline is negligible. The centerline width is then divided by the true measure of the rack top width in millimeters to get the pixel scale. The true rack top width is 463 millimeters. The bottom part of the table contains statistics about the

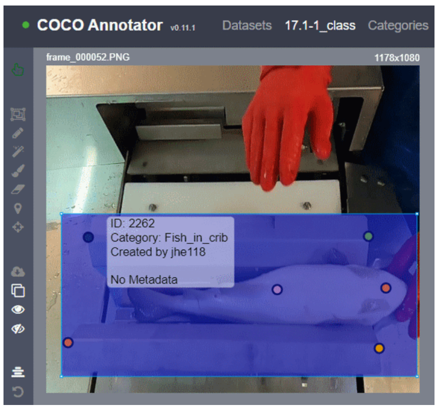
COCO challenges that many of these algorithms are built for and evaluated against.

values in each column. This to be able to say something about the annotation quality. The variation and standard deviation for the width of the rack tops seems relatively large (in pixels), this is the perspective giving variable values along the scene, and these statistics can be ignored. The important statistics are those about the cut length. are just under 1 millimeter, and that implies the annotations have consistent size and position relative to the target. So, the annotations are quite accurate, but they also seem to constantly miss the target by around 5 millimeters. Explanation for this is that the width of bounding boxes are always *outside* of the object (rack tops), hence the pixel scale value will become slightly too large, and subsequently the cut length will always be a bit short of the target value. It is possible to correct for this just by including an offset in the calculations.

Annotations for Detectron2



(a) COCO-Annotator - 4 classes with 2 keypoints.



(b) COCO-Annotator - 1 class with 6 keypoints.

Figure 3.4: Annotation example screenshots from COCO-Annotator.

While custom keypoints was not so simple to implement in *YOLOv7*, it was relatively easy to customize *Detectron2* for any number of keypoints simply by modifying some settings in the configuration. *Detectron2* supports the *COCO-format*, and as the keypoint annotations for *Detectron2* were done with *COCO-annotator* which only exports in *COCO-format*, one did not have to think about the correct annotation format conversion/export. However, *Detectron2* has a limitation regarding custom keypoints. If there are more than one class having keypoint annotations, the number of keypoints has to be equal for all classes.

During the keypoint experiments, one model evolved into three different *Detectron2* models, each needing their own set of annotations. The first keypoint model had 4 classes with 2 keypoints each, as can be seen in figure 3.4a. From here on called *Det2-4cls-2kpts*. The classes and keypoints were given logical and descriptive names:

1. "Cod"
   - Keypoints: ("Anus", "Pelvic fins")
2. "Rack top, Lower"
   - Keypoints: ("Farside", "Opside")
3. "Rack top, Upper"
   - Keypoints: ("Farside", "Opside")
4. "Hands"
   - Keypoints: ("Index finger", "Middle finger")

The second model have 1 class with 6 keypoints, as shown in figure 3.4b, from here on called *Det2-1cls-6kpts*:

1. "Fish_in_rack"
   - Keypoints: ("Rack-L_farside", "Rack-L_opside", "Rack-U_farside", "Rack-U_opside", "Anus", "Pelvic fins")

Finally, the third and last *Detectron2* model got 3 classes with 2 keypoints each, from here on called *Det2-3cls-2kpts*:

1. "Cod"
   - Keypoints: ("Anus", "Pelvic fins")
2. "Rack top"
   - Keypoints: ("Farside", "Opside")
3. "Hands"
   - Keypoints: ("Index finger", "Middle finger")

In appendix C.2 and C.3 are tables showing relevant annotation data for the *Det2-4cls-2kpts* and *Det2-1cls-6kpts* models. The column headers and the statistics part are same as for the *YOLOv7* annotation tables (described in 3.3.1), so no need to give another explanation of the content. The *Det2-3cls-2kpts* model was abandoned late in the project as it proved impossible to differ between the keypoints of the lower and upper rack tops, and the model also failed to detect rack top keypoints, so all data from this model has been taken out of this report.

Just like for the *YOLOv7* annotations, the variance and standard deviation seems quite large for the rack widths, but again the corresponding statistics for the cut lengths are acceptably smaller, roughly between one and two millimeters only, for both *Det2-4cls-2kpts* and *Det2-1cls-6kpts*. When comparing the annotated cut length with the ideal cut length, one can see that it is roughly 7 to 8 millimeters shorter than the target for fish #2 with 150 millimeters, but just a tiny fraction below the target of 180 millimeters for fish #5. It is not absolutely clear why, but the measurements were done with a flexible measuring band, so one explanation could be that the band was following along the circumference of the fish belly, and therefore measured a longer distance than the direct horizontal distance as seen by the camera. For farmed cod in this case, small/short fish tend to have a more "balloon-like" belly curvature than the larger/longer fish, and thus the difference in estimated and measured length would be larger for the smaller fish.

4
ML models

4.1 State-of-the-art

Computer vision and machine learning have made significant progress in recent years, with many applications in areas such as object detection, image classification, and face recognition. This section is an overview of the related research in the field of computer vision and machine learning that is relevant for the fish industry.

Object detection is a fundamental problem in computer vision, which involves identifying the presence and location of objects in an image. Over the years, several approaches have been proposed to address this problem, including sliding window-based methods, region-based methods, and anchor-based methods. One of the most popular anchor-based methods is *YOLO*, which was first introduced in 2015[12]. Since then, several versions of YOLO have been proposed, which is known for their high accuracy and fast inference speed.

Another popular approach for object detection is two-stage detection, which involves generating region proposals followed by classification and bounding box regression. One of the most widely used two-stage detectors is Faster R-CNN, which was introduced in 2015. Since then, several variations of Faster R-CNN have been proposed, including Mask R-CNN, which adds a mask prediction branch to the original Faster R-CNN. Facebook's *Detectron2*[10] is also a two-stage detector, with ability to perform detection tasks such as bounding-box detection, instance and semantic segmentation, and keypoint detection.

Keypoint detection is another important problem in computer vision, which involves identifying specific points or landmarks on an object. One of the most popular methods for keypoint detection is the DeepPose model[13], which was introduced in 2014. Since then, several variations of DeepPose have been proposed, including the convolutional pose machines (CPMs)[14], which use a sequence of convolutional networks to refine the initial predictions. Recently, there has been a growing interest in using deep learning techniques within fisheries, aquaculture and agriculture. Several approaches have been proposed for detecting fish in underwater images and videos, including deep learning-based methods. Listed below are a few studies that have been relevant for this project.

- A peduncle detection method of tomato for autonomous harvesting[15]
- Multi-level feature fusion for fruit bearing branch keypoint detection[16]
- Improving fish from catch to the consumer[17]
- Robust automatic net damage detection and tracking on real aquaculture environment using computer vision[18]
- Deep Semisupervised Semantic Segmentation in Multifrequency Echosounder Data[19]
- A visual detection method for nighttime litchi fruits and fruiting stems[20]
- Vegetable Size Measurement Based on Stereo Camera and Keypoints Detection[21]
- Yolov4-tiny with wing convolution layer for detecting fish body part[22]

Since both the object detection models mentioned above have well developed source code[11][10], and relatively easy to take in use, they were chosen for the *ML* experiments in this project were we compared keypoint based *Detectron2* models with region based single shot detectors such: *YOLOv7* models. Our initial assumption was that a keypoint based model could be better suited for our task.

4.2 YOLO

YOLOv7[9] is the latest version[vi] of the YOLO series and is considered one of the fastest object detection models available. Research on *YOLOv7* has focused on improving its accuracy and efficiency, including the use of feature pyramids, anchor boxes, and multi-scale training. *YOLOv7* is a one-stage object detection model that directly predicts the class and location of objects in the image. *YOLOv7* takes an image as input and passes it through a series of convolutional and max pooling layers to extract features. For feature extraction, *YOLOv7* uses a series of convolutional and max pooling layers to extract features from the input image. Then the extracted features are passed through several prediction layers that make predictions about the presence of objects in the image and their locations. Next, *YOLOv7* uses anchor boxes to make predictions about the size and shape of objects in the image. To eliminate overlapping bounding boxes and obtain the final detections, *YOLOv7* applies Non-Maximum Suppression (NMS) on the predictions. The *YOLOv7* model is trained using a multi-scale training approach and uses a custom loss function that takes into account the accuracy of bounding box predictions, objectness predictions, and classification predictions.

4.2.1 Choice of YOLO models

There are different sized *YOLOv7* models to chose from, ranging from roughly 151 million down to 6 million parameters. The larger ones from 100 million parameters are meant for running on high-end cloud GPUs, the small ones between 6 to 70 million parameters are meant for consumer (normal) GPUs and edge GPUs. For this project the two models with the least number of parameters were selected, *YOLOv7 std.* with 37 million parameters and *YOLOv7 tiny* with 6 million parameters. These are mentioned further in this document just as *YOLOv7 std.* and *YOLOv7 tiny*. Plan is to run the final *CV* system on an embedded computer with an edge GPU and one would like the predictions to be reasonably fast, so the final, recommended model should not be too big.

For the two *YOLOv7* models, their standard configuration was used, only adapted for number of classes and class names. The configuration for both models are found in appendix A. For training, the model reduced the input images to 640x640 pixels.

4.3 Detectron2

Detectron2 is a popular open-source framework for computer vision and machine learning developed by Facebook AI. It is used for various computer vision tasks including object detection, instance segmentation, and keypoint detection. Research on Detectron2 has focused on improving its performance and efficiency, as well as exploring its use for various computer vision tasks.

Detectron2 is a two-stage object detection model that uses a backbone network, such as ResNet or FPN, to extract features from the input image, and a *Region of Interest (RoI)* head predicts the class and location of objects in the image. *Detectron2* is trained using a multi-task loss function that takes into account the loss from multiple heads and the accuracy of multiple predictions.

vi. A bird twittered that a YOLO version 8 has been released recently, but at the time of writing there has not been time to verify this.

4.3.1 Setup of Detectron2

The source code for *Detectron2* is made flexible for users to customize for own projects, but it takes some effort to get your custom code to work properly. For this project, functions like file/folder handling, trainer, augmentation and evaluation were modularized.

Pre-trained weights for ResNet-50 were used for training to keep the number of parameters around same level as the *YOLOv7 std.* model. The configuration for both models are found in appendix B. For training, the model reduced the input images to 640x571 pixels (original aspect ratio).

4.4 Computing resources for training

Mainly free, open source software has been used in this project. The most important ones being VSCode and Anaconda with Spyder for programming the *ML* algorithms in Python, and a LaTeXcombination of MikTex, TexMaker and VSCode for writing this thesis. All running on a consumer grade, Windows 10 laptop, without an *ML* compatible GPU from Nvidia.

4.4.1 WSL2 and Docker

The two web-based *data annotation* programs were both installed locally on the Windows laptop with Docker images on *Windows Subsystem for Linux, version 2 (WSL2)*. They could then be opened in the standard internet browser of the computer and have direct access to the file system, without the need for transferring a lot of image files over the internet.

4.4.2 Google Colab

The very first training of the *YOLOv7 std.* model was done online with Google Colab Pro (paid subscription). The power and speed of Google's high-end GPUs made the training very fast, but due to a combination of cost and cumbersome, time consuming data transfer of many large *ML* files, this option was abandoned in favor of a local hardware alternative at the *UiT The Arctic University of Norway (UiT)*.

4.4.3 Local GPU cluster

The *ML* group at *UiT* has a GPU cluster called *Springfield*[23], that is put available for graduate students. All but one training session and all the predictions were run on *Springfield*. Getting access to and learning how to use *Springfield* took a few days, but then it was easy and straightforward to get a good setup for folder structure and a decent work flow.

Part III

Experiments and Results

/5 Experiments

5.1 YOLOv7 training

The*YOLOv7 std.* model was trained for 100 epochs and the *YOLOv7 tiny* model for 200 epochs. Pretrained weights from the *COCO* dataset (provided with the *YOLOv7* source code[11]) were used to get a "flying" start for both models.

The first training of the *YOLOv7 std.* model was not successful as it was not able to detect the rack tops during validation. The problem seemed to be that the height/width ratio of the bounding boxes was very small, only covering one of the long edges of the rack tops, so the annotated features probably became too small for the model to make a good feature map. After increasing height of the rack top annotations, so that whole of the lower/upper rack top was inside the bounding box, the model performed better and was able to detect all of the five classes after training. The confusion matrices in figures 5.1 and 5.2 gives a good visualization of the improvement from first to second model. The confusion matrix and loss/performance curves for the training are shown in figure 5.3 and 5.4, respectively. The *precision* and *recall* curves peaks close to 1, as does the *mAP* curves.

YOLOv7 tiny was quickly set up same way as *YOLOv7 std.*, only adapted the standard config files with number of classes and class names before training. The confusion matrix and evaluation curves for the training are shown in figure 5.5 and 5.6, respectively. Like for *YOLOv7 std.*, the performance is peaking close to 1. The *val Classification* loss curve is a bit unstable with an increasing trend, but none of the other loss curves have any particular signs of overfitting, so no changes in config or number of epochs done.

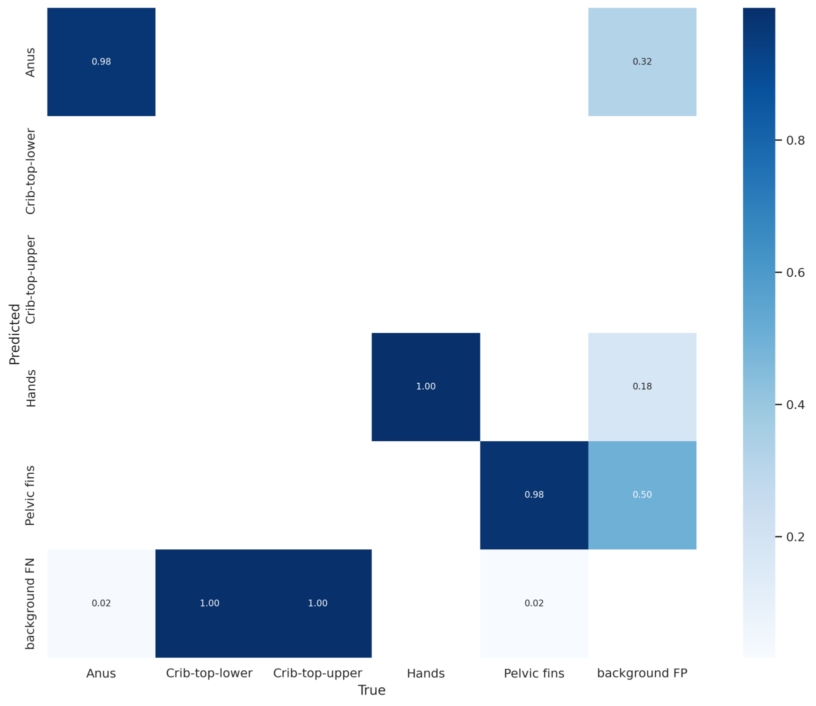


Figure 5.1: *YOLOv7 std.* confusion matrix from initial test training - with slim (long and thin) rack top annotations.

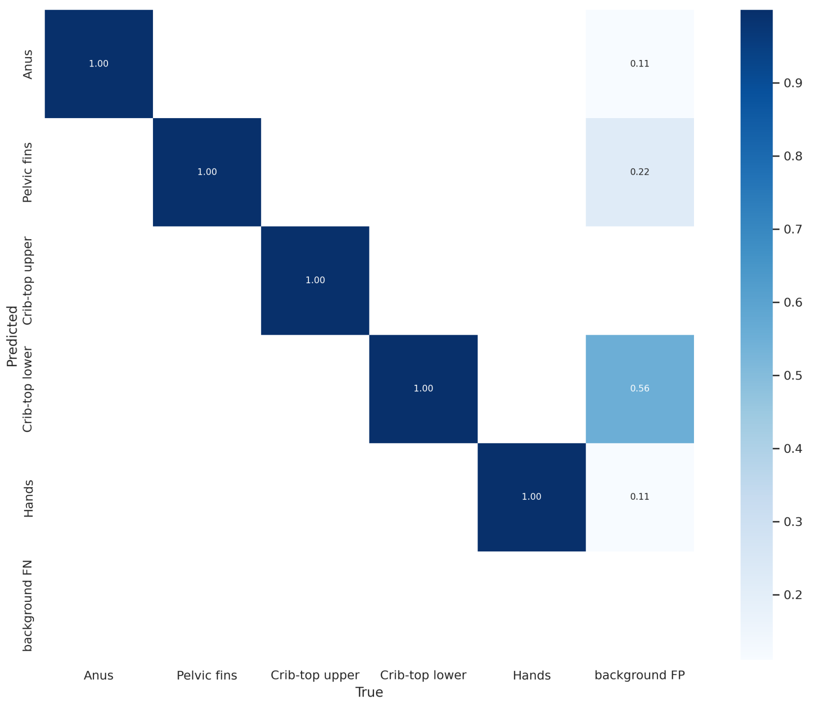


Figure 5.2: *YOLOv7 std.* confusion matrix after enlargement of the rack top annotations.

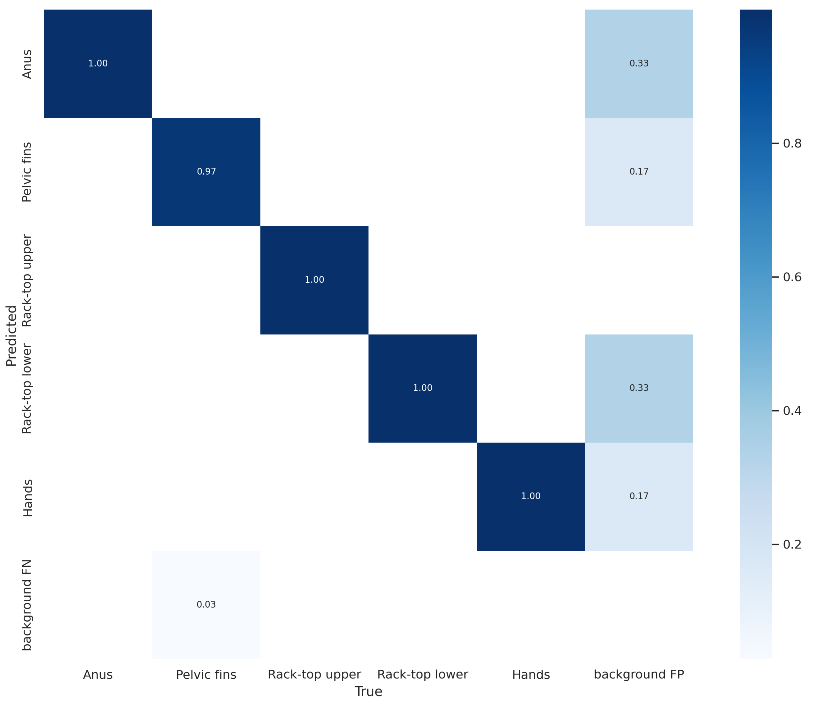


Figure 5.3: Confusion matrix from training of *YOLOv7 std.*.

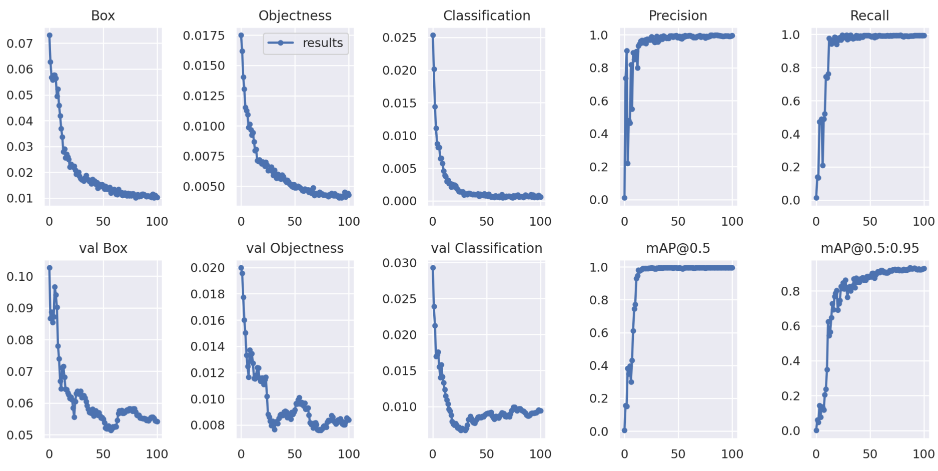


Figure 5.4: Loss and performance curves from training of *YOLOv7 std.*.

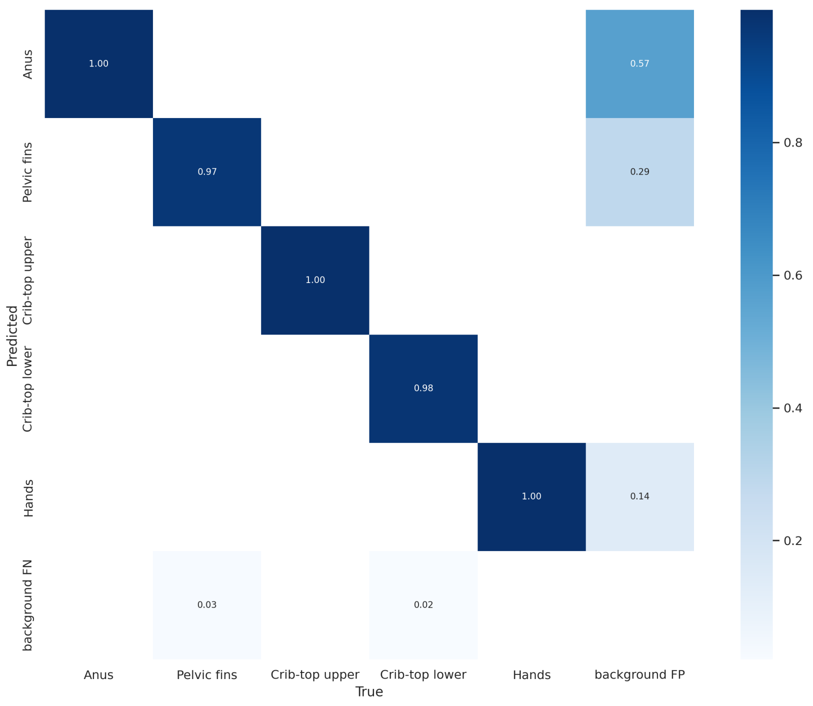


Figure 5.5: Confusion matrix from training of *YOLOv7 tiny*.

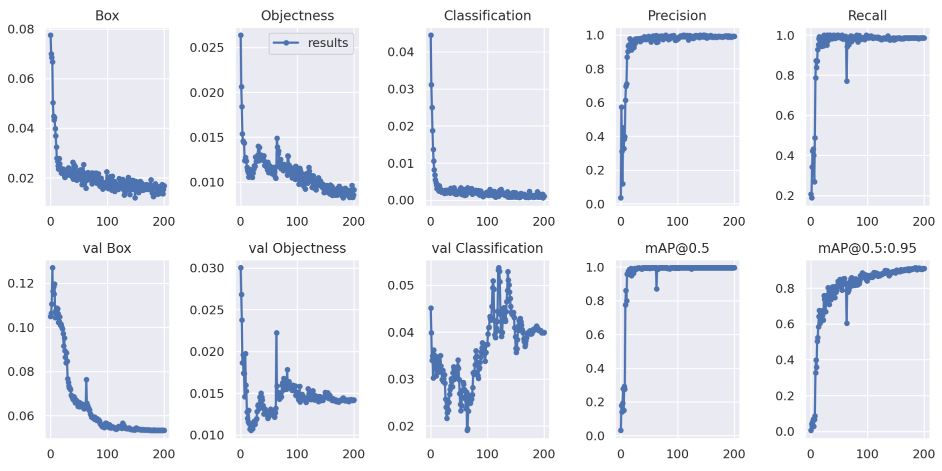


Figure 5.6: Loss and performance curves from training of *YOLOv7 tiny*.

5.2 Detectron2 training

The idea to develop a keypoints model is that keypoint detection could possible be more precise for the purpose of accurately estimating position of and distance between the keypoints than would be for a more "floating" bounding box detection. Three different *Detectron2* keypoint models were used for the experiments. Annotating images is a time consuming task and for this project, in order to achieve a first working model, we had to use quite a lot of time on annotation and setup of the model programming codes.

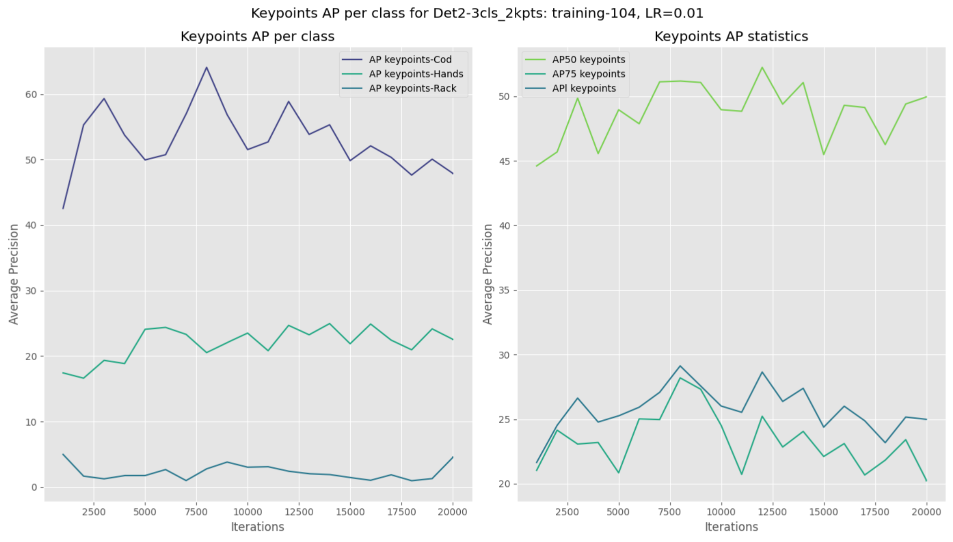
To start with, a few short training runs were made to find a suitable learning rate range for the experiments. For *Det2-1cls-6kpts* and *Det2-4cls-2kpts* the range (0.005 - 0.015) was ok, but for *Det2-3cls-2kpts* $\text{lr} = 0.015$ was close to the upper limit as giving occasional early stops in training due to infinite values.

As the standard *Detectron2* source code does not do augmentations on the input images, another experiment was to add augmented images of the dataset. At first, a few different augmentations and transformations from the Python *Albumentation*[24] package were added, like *Spatter*, *RandomFog* and *RandomRain*. Idea was to make the model more robust by imitating debris on the camera lens, like blood stains or water condensation and water drops. However, this gave a performance drop in object detection under normal conditions for all models. In a second augmentation attempt, in order to improve the general performance, made only color transformations and image enhancement (also from the *Albumentation* package) like *ChannelShuffle*, *Sharpen*, *FancyPCA*[25], *CLAHE*[vii].

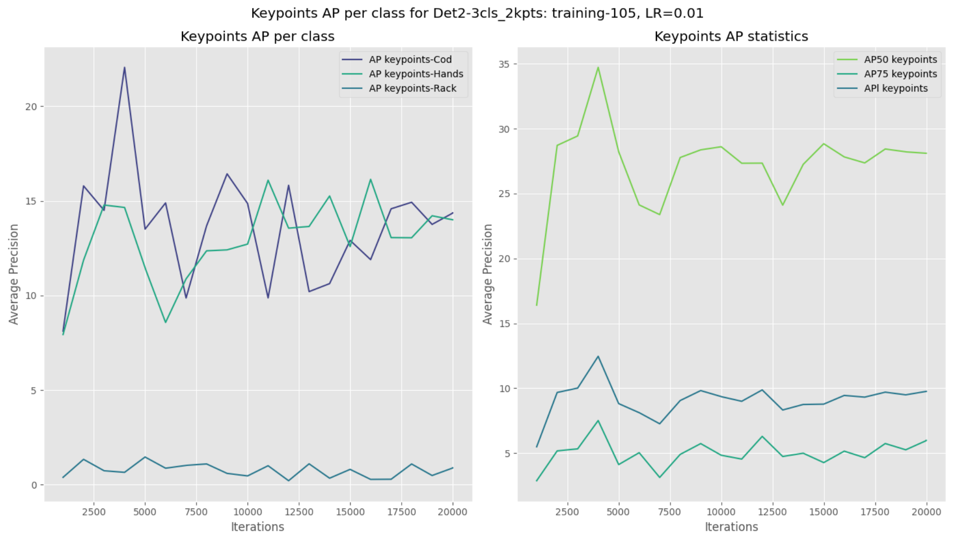
Of all the three *Detectron2* models, *Det2-3cls-2kpts* had the lowest performance. The average precision for the rack keypoints was never better than 5%, and adding color- and contrast-enhanced images made it even worse, as can be seen in figure 5.7. Most likely, due to having both the lower and upper rack top as one class, made the model confused. Not a problem for the bounding box detection, but since the keypoints are positional related, the keypoints positional order flips between the lower and upper rack top due to their symmetry. This duality caused the model to fail, and it also became impossible to compute prediction statistics, so all work on this model stopped here, and no results from this model is presented in this report.

Of the remaining *Detectron2* models, *Det2-1cls-6kpts* had clearly the best keypoint precision curves, with *AP* with roughly 70% and *mAP* close to 100%. With *Det2-4cls-2kpts*, *AP* for the rack top keypoints was peaking at only 10%, and peaking at 40% for the cod keypoints. The respective curves are put next to each other in figure 5.8 for comparison.

vii. Contrast Limited Adaptive Histogram Equalization

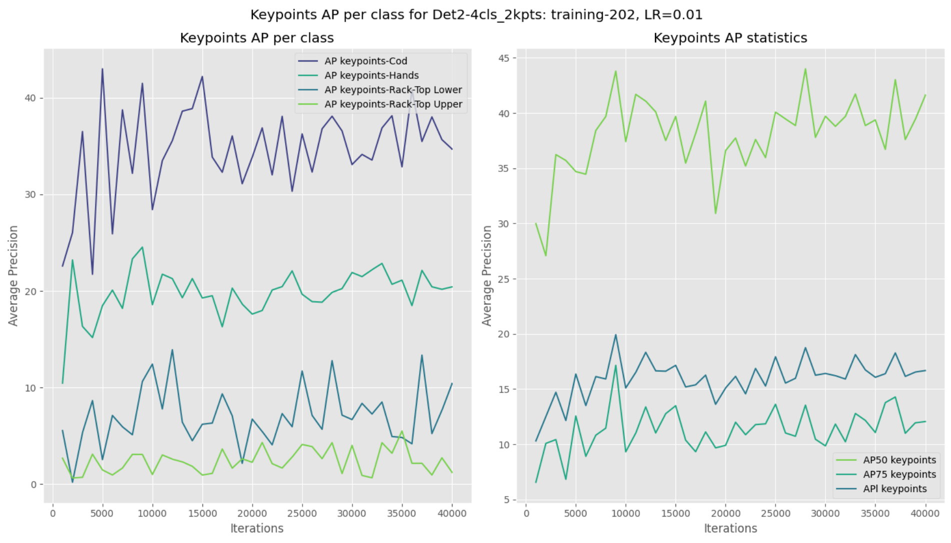


(a) Keypoint precision curves *without* extra augmentations.

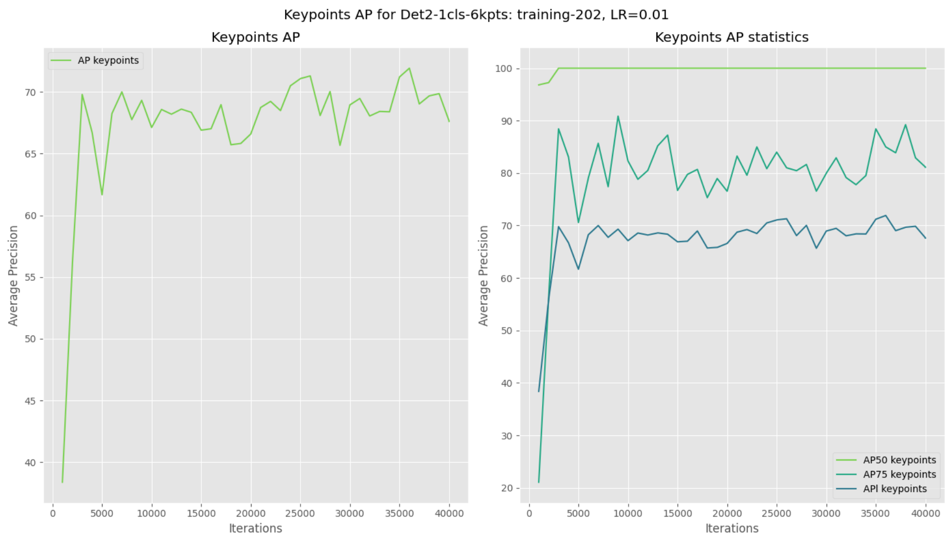


(b) Keypoint precision curves *with* extra augmentations.

Figure 5.7: Keypoint precision curves for Det2-3cls-2kpts. Adding color- and contrast-augmented images made the model's performance worse.



(a) Keypoint precision curves for *Det2-4cls-2kpts*.



(b) Keypoint precision curves for *Det2-1cls-6kpts*.

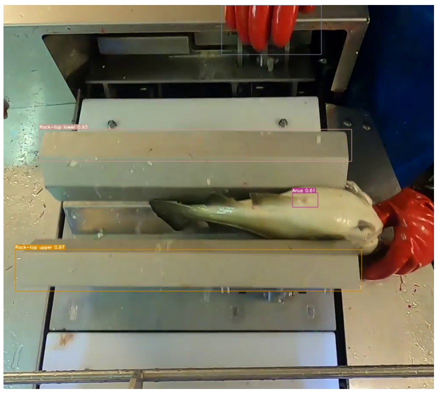
Figure 5.8: Keypoint precision curves for *Det2-4cls-2kpts* and *Det2-1cls-6kpts*.

/6 Results

6.1 YOLOv7 results

Although the *YOLOv7* models was only run with the standard configurations, they show very good performance. The predictions are precise and stable, definitely not bad for non-tweaked algorithms. As one can see in the subsequent prediction data tables and regression plots, it is clear that the *YOLOv7 std.* model has the absolute lowest variation and standard deviation of all the *ML* models in this project. *YOLOv7 tiny* being second best, and even outperforms all the other models on the difficult first fish in the prediction dataset, missing only one detection of the pelvic fins. This first fish is quite small and has an awkward posture, like it is on the brim of sliding out of the rack, therefore the algorithms seem to have trouble making good detections on that one. In figure 6.1, is shown the predictions done by *YOLOv7 std.* and *YOLOv7 tiny* on the same image, for which *YOLOv7 tiny* were able to detect all *Regions of Interest (RoIs)*, while *YOLOv7 std.* missed the pelvic fins. For all the frames of fish #1, *YOLOv7 std.* did miss in total seven detections of the pelvic fins and so failed to compute a cut length for the image-ids 12 to 18.

In the predictions data tables D.1 through D.4, are the predictions in numbers and the corresponding statistics for four selected fish sequences. Again, like for the annotation data tables, the variance and standard deviation of the rack top widths are not so interesting. The important numbers are for how precise the estimated cut length will be. For *YOLOv7 std.*, the standard deviation for the estimated cut length is just under 2 millimeters for all sequences, except for the first, difficult fish where it jumps to 4 millimeters. *YOLOv7 tiny* estimated cut length has a standard deviation of just over 5 millimeters for the first fish and between 1 and 3 millimeters for the other fish sequences.


(a) Predictions on fish #1 - *YOLOv7 std.*

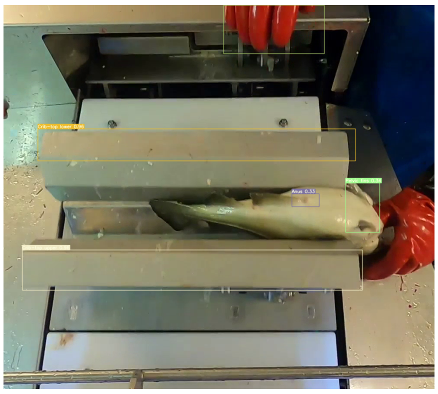
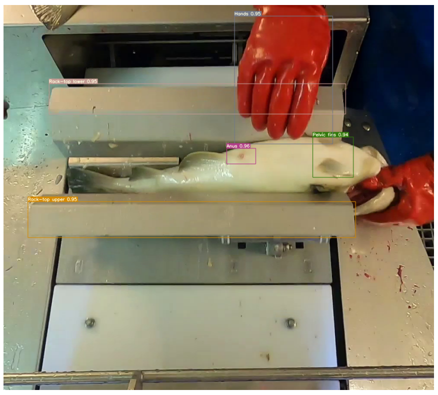

(b) Predictions on fish #1 - *YOLOv7 tiny*

Figure 6.1: Comparing object detection performance on image-id #13 of fish #1. Corresponding data tables D.1 and D.5 for (a) and (b) respectively.


(a) Predictions on fish #3 - *YOLOv7 std.*

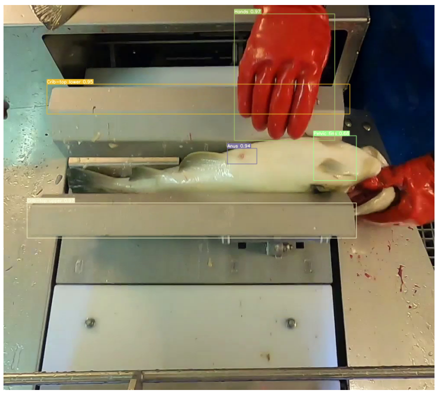
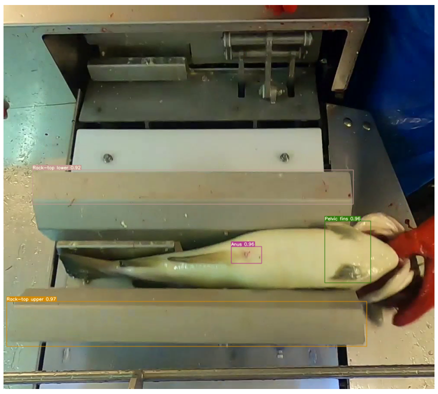

(b) Predictions on fish #3 - *YOLOv7 tiny*

Figure 6.2: Comparing object detection performance on image-id #48 of fish #3. Corresponding data tables D.2 and D.6 for (a) and (b) respectively.


(a) Predictions on fish #5 - *YOLOv7 std.*

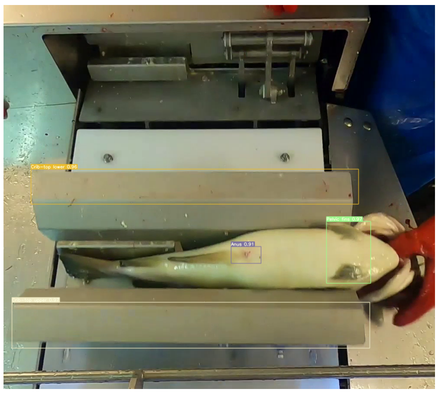
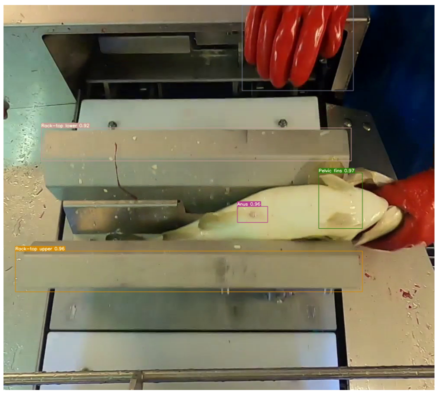

(b) Predictions on fish #5 - *YOLOv7* tiny

Figure 6.3: Comparing object detection performance on image-id #106 of fish #5. Corresponding data tables D.3 and D.7 for (a) and (b) respectively.


(a) Predictions on fish #8 - *YOLOv7 std.*

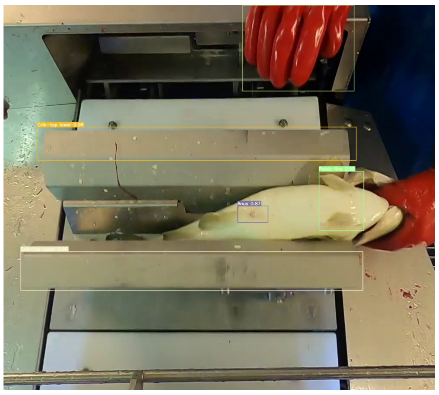

(b) Predictions on fish #8 - *YOLOv7 tiny*

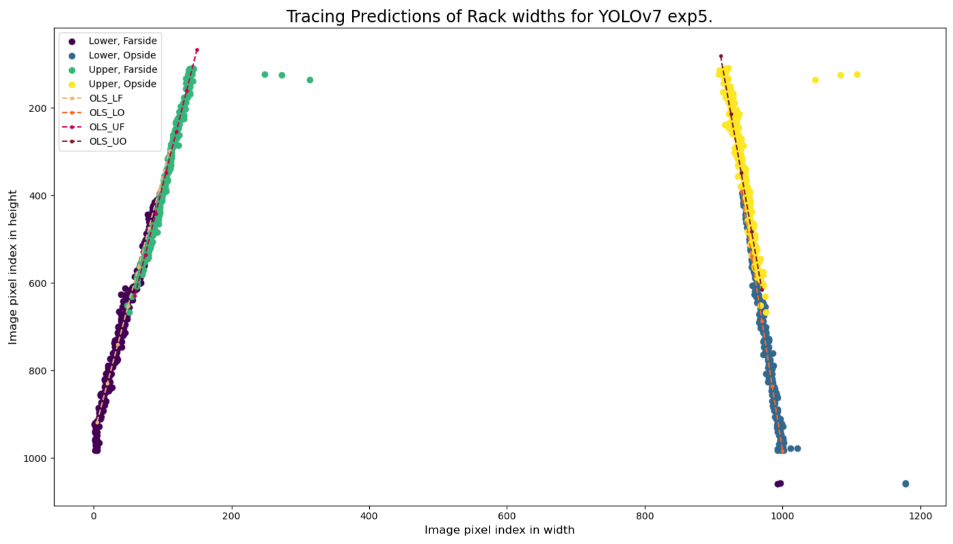
Figure 6.4: Comparing object detection performance on image-id #173 of fish #8. Corresponding data tables D.4 and D.8 for (a) and (b) respectively.

6.1.1 Linear regression of the YOLOv7 models' predictions

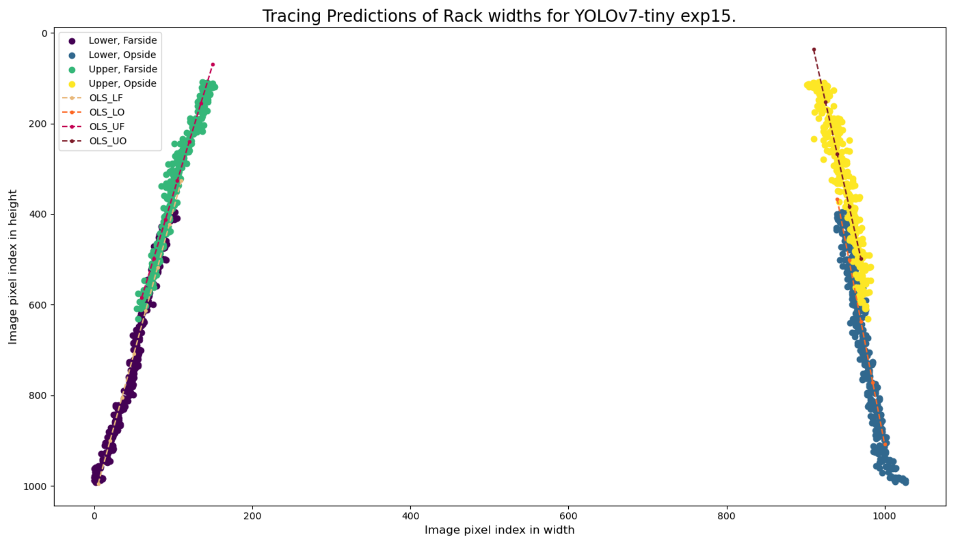
Another quality check of the model's performance, is to extract some important data and perform regression on them to visually present the results. Here, it is natural to use the rack width data, as they are definitely moving linearly and they have a rigid, fixed size. The fish has a soft and flexible body, and although it's fixed in the rack, it might move around a bit, so the points on the fish will be less fit for this purpose. Figure 6.5 shows plots of the *Ordinary Least Squares (OLS)* regression lines fitted to the rack top keypoints. Both models show relatively steady predictions. The *YOLOv7 std.* model have some odd outliers at the scene extremities, but the variation is quite low. The *YOLOv7 tiny* model has no outliers, but the variance is somewhat larger than for *YOLOv7 std.*. Looking through all predictions, the outliers come from six false positives with confidence values ranging from 0.29 to 0.62. The confidence threshold was set as low as 0.25, an upping to 0.35 would have removed half of the false detections. The remaining three comes from the lower left corner of the scene, for when the conveyor comes to a certain position, the structure looks just like the corner of a lower rack top. A more tight cropping of the scene would have removed this situation, but was impossible to see beforehand. With camera mounted in a more appropriate position, this exact problem is not likely to be an issue in the future.

In the regression tables 6.1, 6.2, 6.3 and 6.4 are the cumulated statistics for each of the midpoints[viii]of the rack's side edges, as they move from top to bottom of the scene for every fish in the predictions dataset.

viii. These midpoints are being used as "rack keypoints" so the *YOLOv7* models' predictions can be comparable to the two *Detectron2* models.



(a) Linear regression of the *YOLOv7 std.* predictions on the racks. Corresponding statistics tables 6.1 and 6.2



(b) Linear regression of the *YOLOv7 tiny* predictions on the racks. Corresponding statistics tables 6.3 and 6.4

Figure 6.5: Plots of the rack top predictions together with corresponding regression lines for both *YOLOv7 std.* and *YOLOv7 tiny* model.

Table 6.1: OLS coefficients statistics from regression of rack keypoints from predictions of *YOLOv7 std.*, cumulated per fish-scene for whole of dataset 18.5.

Keypoint id	OLS coeffs. mean b0	OLS coeffs. mean b1	OLS coeffs. median b0	OLS coeffs. median b1
UF	983.9	-5.973	1.004×10^3	-6.247
UO	-7.521×10^3	8.360	-8.038×10^3	8.921
LF	915.9	-5.225	947.3	-5.905
LO	-8.196×10^3	9.141	-8.914×10^3	9.897

Keypoint id	OLS coeffs. var b0	OLS coeffs. var b1	OLS coeffs. std b0	OLS coeffs. std b1
UF	9.975×10^3	1.061	99.87	1.030
UO	4.931×10^6	5.533	2.221×10^3	2.352
LF	1.038×10^4	4.054	101.9	2.013
LO	4.974×10^6	5.325	2.230×10^3	2.308

Table 6.2: OLS residuals statistics from regression of rack keypoints from predictions of *YOLOv7 std.*, cumulated per fish-scene for whole of dataset 18.5.

Keypoint id	Cumulated OLS residuals mean	median	var	std
UF	-1.642×10^{-15}	-1.553	4.832×10^3	41.41
UO	-3.290×10^{-13}	-1.291	2.619×10^3	34.67
LF	1.156×10^{-14}	0.9265	792.5	18.93
LO	-4.210×10^{-13}	-1.235	2.215×10^3	34.58

Table 6.3: OLS coefficients statistics from regression of rack keypoints from predictions of *YOLOv7 tiny*, cumulated per fish-scene for whole of dataset 18.5.

Keypoint id	OLS coeffs. mean b0	OLS coeffs. mean b1	OLS coeffs. median b0	OLS coeffs. median b1
UF	937.8	−5.790	928.0	−5.727
UO	-7.015×10^3	7.748	-6.972×10^3	7.701
LF	1.032×10^3	−6.457	1.027×10^3	−6.365
LO	-8.044×10^3	8.963	-8.112×10^3	9.020
Keypoint id	**OLS coeffs. var b0**	**OLS coeffs. var b1**	**OLS coeffs. std b0**	**OLS coeffs. std b1**
UF	1.156×10^3	1.079	34.00	0.3285
UO	1.334×10^6	1.421	1.155×10^3	1.192
LF	2.862×10^2	1.052	16.92	0.3243
LO	4.662×10^5	4.942	682.8	0.7030

Table 6.4: OLS residuals statistics from regression of rack keypoints from predictions of *YOLOv7 tiny*, cumulated per fish-scene for whole of dataset 18.5.

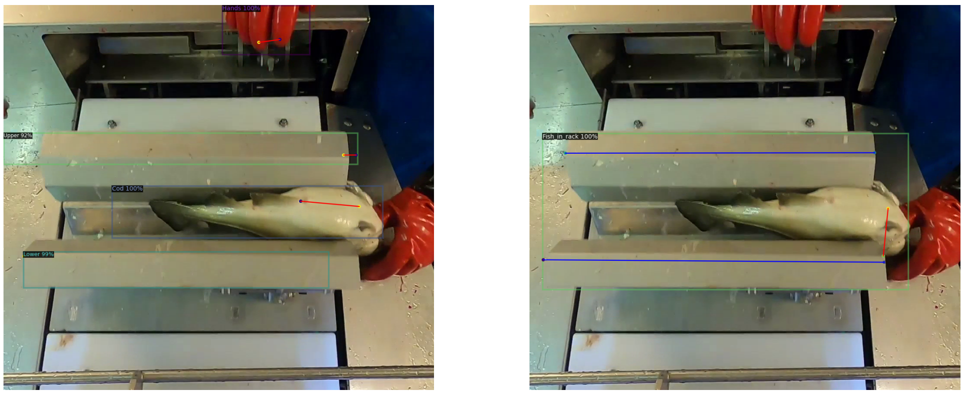
Keypoint id	Cumulated OLS residuals mean	median	var	std
UF	5.189×10^{-14}	2.709	750.7	26.45
UO	1.498×10^{-13}	−0.5961	1.739×10^3	41.14
LF	9.978×10^{-14}	−2.679	682.2	25.86
LO	1.286×10^{-13}	2.200	1.855×10^3	42.13

6.2 Detectron2 results

The *Det2-4cls-2kpts* model has big problems detecting the rack keypoints, resulting in none or completely incorrect cut length estimations. The prediction data tables in appendix **??** presents the predictions in numbers and statistics.

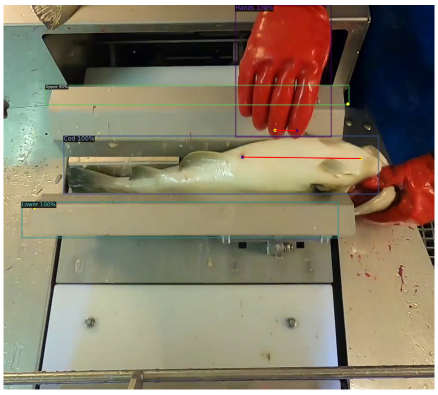
On the other hand, *Det2-1cls-6kpts* is giving much better numbers, in appendix D.4 the data tables D.13 through D.16, one can see that the variation and standard deviation of the pixel scale values are on par with *YOLOv7*. However, *Det2-1cls-6kpts* is a bit more unstable than *YOLOv7* in general. Like *YOLOv7* for the image-ids 12 to 18, it failed to compute a adequate cut length on the difficult fish #1. On figure 6.6 one can see that *Det2-4cls-2kpts* fails to correctly detect the rack keypoints, and *Det2-1cls-6kpts* fails to detect the keypoints for anus and pelvic fins.

In the predictions data tables D.9 through D.12 for *Det2-4cls-2kpts* and D.13 through D.16 for *Det2-1cls-6kpts*, are the predictions in numbers and the corresponding statistics for the same four selected fish sequences as for the *YOLOv7* models. For *Det2-4cls-2kpts*, the standard deviation for the estimated cut length is not giving reasonable values for any sequences. This model fails to detect the rack keypoints, and therefore is not able to estimate the cut length properly. *Det2-1cls-6kpts* gives stable values close to the target cut length, and the estimated cut lengths have standard deviations varies from 2 to 12 millimeters, and goes up to just over 23 millimeters for the first fish.

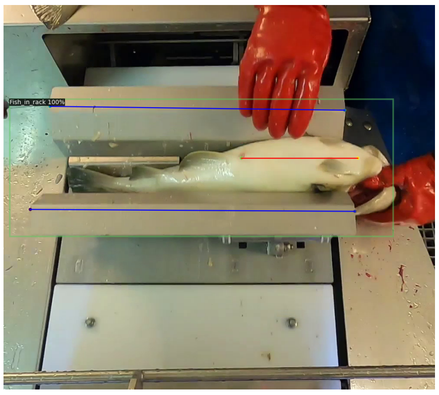


(a) Predictions on fish #1 - *Det2-4cls-2kpts* **(b)** Predictions on fish #1 - *Det2-1cls-6kpts*

Figure 6.6: Comparing object detection performance on image-id #13 of fish #1. Corresponding data tables D.9 and D.13 for (a) and (b) respectively.

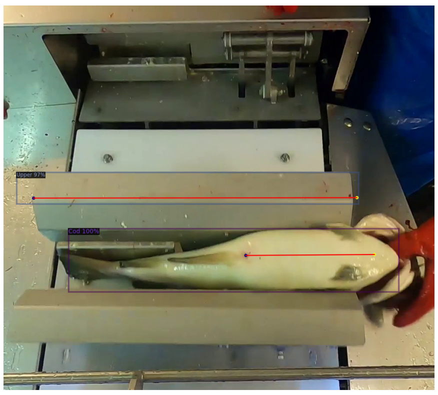


(a) Predictions on fish #3 - *Det2-4cls-2kpts*.

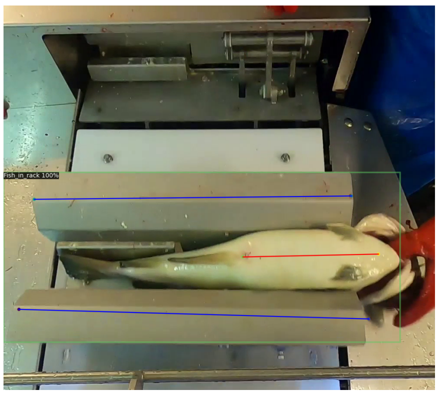


(b) Predictions on fish #3 - *Det2-1cls-6kpts*.

Figure 6.7: Comparing object detection performance on image-id #48 of fish #3. Corresponding data tables D.10 and D.14 for (a) and (b) respectively.



(a) Predictions on fish #5 - *Det2-4cls-2kpts*.

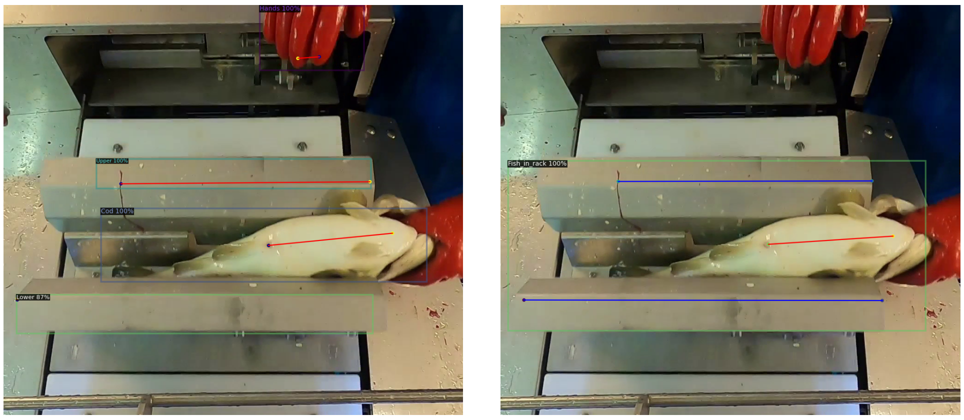


(b) Predictions on fish #5 -*Det2-1cls-6kpts*.

Figure 6.8: Comparing object detection performance on image-id #106 of fish #5. Corresponding data tables D.11 and D.15 for (a) and (b) respectively.

6.2.1 Linear regression of the Detectron2 models' predictions

Like done for *YOLOv7*, figure 6.10 shows plots of the *OLS* regression lines fitted to the rack top keypoints. The regression lines for *Det2-4cls-2kpts* The predicted points are more or less chaotically spread around the scene, and hence the regression lines are going anywhere but the direction they are expected to do. The *Det2-1cls-6kpts* model does much better, only the lower rack keypoint on the operator's side that are heavily influenced by outliers. The regression lines are drawn up using the median of the fitted parameters in order to be less affected by the outlier predictions. Towards the center of the plots, one notices a series of stand-alone *UF* keypoints. Taking a look at figure 6.9, it is easy to



(a) Predictions on fish #8 - *Det2-4cls-2kpts*.   **(b)** Predictions on fish #8 - *Det2-1cls-6kpts*.

Figure 6.9: Comparing object detection performance on image-id #175 of fish #8. Corresponding tables D.12 and D.16 for (a) and (b) respectively.

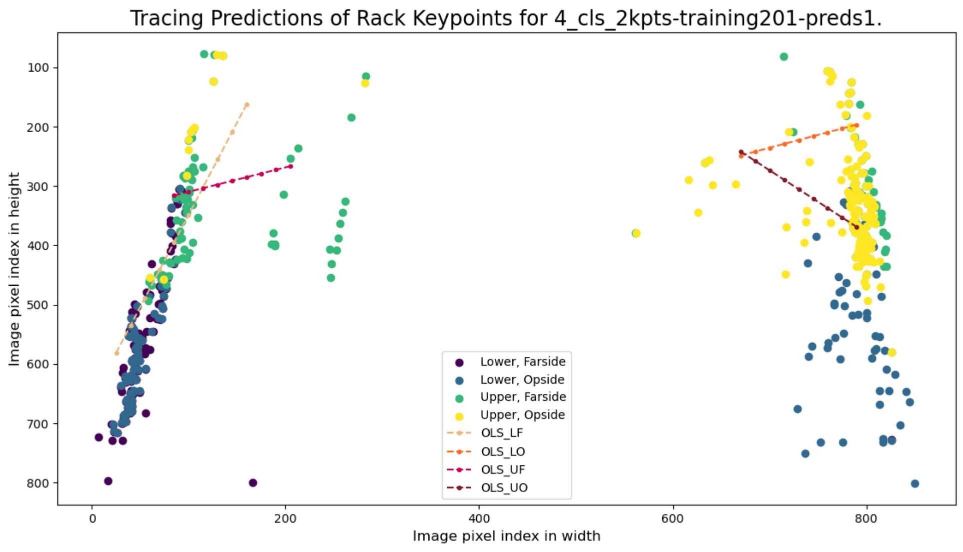
spot the reason, a thick line of blood residue across the upper rack is being wrongly detected as the *UF* keypoint by both models.

In data tables 6.5, 6.6, 6.7 and 6.8 are the cumulated statistics for each of the rack keypoints, as they move from top to bottom of the scene for every fish in the predictions dataset.

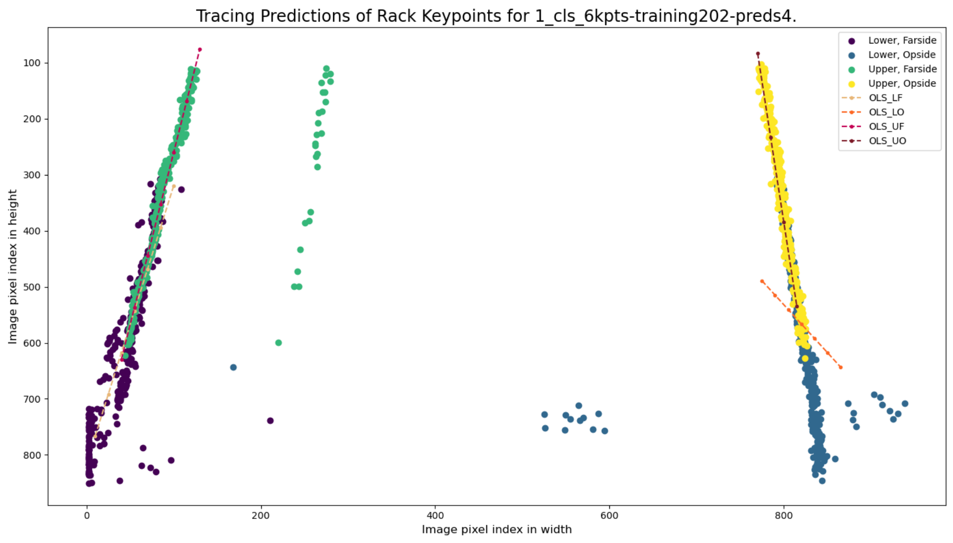
Table 6.5: OLS coefficients statistics from regression of rack keypoints from predictions of *Det2-4cls-2kpts*, cumulated per fish-scene for whole of dataset 18.5.

Keypoint id	OLS coeffs. mean b0	OLS coeffs. mean b1	OLS coeffs. median b0	OLS coeffs. median b1
UF	351.0	−0.4122	364.1	−0.026 74
UO	−469.1	1.061	66.74	0.3534
LF	659.3	−3.109	779.2	−4.076
LO	539.5	−0.4342	585.9	−0.1620

Keypoint id	OLS coeffs. var b0	OLS coeffs. var b1	OLS coeffs. std b0	OLS coeffs. std b1
UF	4.839×10^4	2.750	220.0	1.658
UO	1.783×10^6	2.856	1.335×10^3	1.690
LF	8.973×10^4	1.184	299.5	3.442
LO	6.788×10^4	2.615	260.5	1.617



(a) Linear regression of the *Det2-4cls-2kpts* predictions on the rack keypoints.



(b) Linear regression of the *Det2-1cls-6kpts* predictions on the rack keypoints.

Figure 6.10: Plots of the rack top predictions together with corresponding regression lines for both *Det2-4cls-2kpts* and *Det2-1cls-6kpts* model.

Table 6.6: OLS residuals statistics from regression of rack keypoints from predictions of *Det2-4cls-2kpts*, cumulated per fish-scene for whole of dataset 18.5.

Keypoint	Cumulated OLS residuals			
id	mean	median	var	std
UF	4.436×10^{-14}	0.057 13	2.896×10^{3}	44.58
UO	-2.254×10^{-13}	1.491	2.545×10^{3}	41.33
LF	1.858×10^{-13}	1.444	1.180×10^{3}	30.15
LO	3.792×10^{-14}	1.075	2.950×10^{3}	45.02

Table 6.7: OLS coefficients statistics from regression of rack keypoints from predictions of *Det2-1cls-6kpts*, cumulated per fish-scene for whole of dataset 18.5.

Keypoint	OLS coeffs. mean		OLS coeffs. median	
id	b0	b1	b0	b1
UF	830.7	−5.531	875.5	−6.151
UO	-7.580×10^{3}	9.936	-7.644×10^{3}	10.04
LF	806.0	4.813	815.9	−4.962
LO	-1.923×10^{3}	3.044	−834.7	1.708

Keypoint	OLS coeffs. var		OLS coeffs. std	
id	b0	b1	b0	b1
UF	1.401×10^{4}	2.775	118.4	1.666
UO	1.719×10^{5}	0.2717	414.6	0.5213
LF	1.277×10^{3}	0.7318	35.74	0.8554
LO	8.761×10^{6}	12.98	2.960×10^{3}	3.603

Table 6.8: OLS residuals statistics from regression of rack keypoints from predictions of *Det2-1cls-6kpts*, cumulated per fish-scene for whole of dataset 18.5.

Keypoint	Cumulated OLS residuals			
id	mean	median	var	std
UF	3.076×10^{-14}	−0.094 56	1.844×10^{3}	29.17
UO	-2.299×10^{-13}	−1.099	528.3	22.55
LF	6.115×10^{-14}	−6.408	4.615×10^{3}	62.34
LO	-3.128×10^{-14}	2.208	1.435×10^{4}	111.7

Part IV

Discussion and Conclusion

/7 Discussion

7.1 Proposed solution

From the results of the experiments, there is not a clear winner of the four remaining *ML* models, but there is one clear loser. *Det2-4cls-2kpts* is falling behind on all measures. It has the absolute lowest numbers of acceptable cut length estimations (in millimeters) because of problems detecting the rack keypoints, even though it's estimated belly line (in pixels) doesn't actually deviate much from the other models. Since, in these models, the computation of the cut length is dependent on proper detection of the rack keypoints, *Det2-4cls-2kpts* cannot work as intended. If, however, the camera had been looking vertically down on the conveyor racks, there wouldn't have been much of a perspective to account for in the scene, and a fixed pixel scale could have been used to compute the cut length in millimeters. Then, *Det2-4cls-2kpts* would probably been able to do make some decent cut length estimations. However, one weakness of setting a fixed pixel scale, is that the cut length estimation becomes wrong if (sometime in the future) the camera is being replaced by one with a higher resolution, or if the distance between camera and rack/fish changes. A dynamically computed pixel scale is invariant of camera resolution or changes in distance. In the end, it is also about trusting the predictions, and *Det2-4cls-2kpts* seems too inaccurate to be trusted for this project.

Taking a closer look at the spread in the models' estimated cut length, it is timely to ask what the acceptable limits are. Thus, if the algorithm is very good at accurately detecting the anus opening and the width of the rack tops (or trained to recognize any other background object with fixed and known measures), then it will satisfy the requirement to be used in a *CV* system for Folla. Unfortunately maybe, that extracting the the anus detections data was not done before deadline for this thesis, as it would also be good to know how well these models performs on that. Should be done in the continuation of this project.

The next *Detectron2* model, *Det2-1cls-6kpts*, is much more trustworthy in it's predictions of the rack keypoints. The standard deviation in estimated cut length, for the selected fish in chapter 6, lies in general between 2.5 mm to 11.9 mm, increasing up to 23.5 mm for the small and oddly positioned fish #1. However, in figure 6.6b one can see an issue with this model; the anus keypoint detection is completely wrong. According to [21] and [26], keypoint detection networks using low-level feature maps and a small training dataset, can have problems detecting small objects and keypoints. It seems reasonable to believe that this is exactly what is being observed here with *Det2-4cls-2kpts* and *Det2-1cls-6kpts*. Again, with the camera mounted inside the Folla machine, looking vertically down on the conveyor, and retraining the model with a expanded dataset, the problems with small and odd fish would much likely be reduced to a negligible level. And occasional odd cut length estimations are possible to average out over a few detections, and also excluding estimations located outside a specified range of coordinates or having too large of an angle. Since this model need extra work to improve the accuracy, while still having some extra uncertainty attached, it cannot be recommended for use in the Folla machine at the moment.

Predictions with the two *YOLOv7* models are definitely more stable than the *Detectron2* models. However, they are not directly comparable the way it has been done here in this project. Thinking about it now at the end of working on this thesis, also the bounding boxes of the *Detectron2* models should have been analyzed. Then it would have been possible to do a direct comparison, but then again only for *Det2-4cls-2kpts* which detects the rack tops equally as the *YOLOv7* models. Anyway, the important part here is to obtain accurate and repeatable predictions within the desired limits for the cutting the fish belly, so if done with bounding boxes or keypoints doesn't really matter. Here in this project, both the two *YOLOv7* models have shown ability to do exactly that. Figure 6.5 shows that *YOLOv7 tiny* has the "keypoints" a little bit more spread out, which should be expected given the lightweight network size. The results from this project shows that both the *YOLOv7* models can be recommended for use in a *CV* system for Folla.

7.2 Future work

There was hope that this Master thesis would also manage to explore the implementation of *CV* in the Folla machine, but the time spent for extra annotations and experimenting with keypoint detections, pushed this over to future work. Roughly speaking, the remaining work can be listed like this:

- Determine detection speed (FPS) of the recommended *ML* models, and what speed that is needed for the machine to process the fish fast enough. Then select the best overall model for the *CV*.
- Find and select a suitable embedded computer for running the CV system. NVIDIA's Jetson seems like a good option.
- Fine-tune, shrink the numbers of parameters and deploy the model.
- Find the best way of communication/interaction between CV system and the PLC in Folla.
- Find suitable location for camera in Folla.
- Obtain new, purpose-built dataset with camera mounted in Folla.
- Annotate new dataset.
- Retrain and fine-tune the selected *ML* model.
- Install and test embedded CV system.
- Make a plan for how the CV system should handle unknown situations or unexpected events, and how the operator could be able to resolve such problems.

7.2.1 Future development

The suggested solution is only monitoring the conveyor racks in 2D, so there might come to situations where 3D vision is needed for making proper cuts. Then one would better see the profile of the belly. 3D vision will make it possible to maneuver the knife more precisely and adapt the cutting depth from start to end, causing less damage to the intestines with consumer value e.g., liver and roe. Perhaps also an evaluation of the cut can be made in posteriori, and thereby perform some kind of calibration of the knife positioning system or other measures.

8 Conclusion

This master project was inspired by challenges faced by commercial fisheries in the north of Norway of controlling food quality and food safety. In this thesis, four different *ML* models' ability to do object and keypoint detection on specific anatomy parts of fish, has been studied. With the aim of recommending a suitable model to be part of a *CV* system for an industrial fish gutting machine that cuts open the fish belly between the pelvic fins and the anus. Requirement that the rotating knife shall not cut into the flesh behind the anus opening, and cut should end (or start) maximum 5 millimeters from the anus opening. Likewise, at the pelvic fins, the cut shall start (or end) 15 millimeters from target along the centerline of the fish, and a sideways offset of roughly ±5 millimeters can be acceptable, depending on the length of the fish.

The experiments were performed with two *YOLOv7* and two *Detectron2* models, *YOLOv7* for object detection with bounding boxes, and *Detectron2* for keypoint detections. The results showed that only one of the *Detectron2* models was able to do keypoint detection repeatedly, but the achieved accuracy was not good enough. Both the *YOLOv7* models were able to meet the cut length requirements and are both got recommended for use in the suggested *CV* solution.

More work still remains before one of the *YOLOv7* models can be taken in use, such as determining the object detection speed, finding a suitable embedded computer with GPU to run the *CV* system on, determining the best way of communication between the PLC in Folla and the *CV* system and finding a suitable location for a camera inside the Folla machine.

Part V

Appendix

A
Configuration for Yolov7

A.1 YOLOv7 config

```
#------------ From 'data.yaml' ----------------------
# dataset paths
train: ../data/DataSet-17.1/train/images  # 436/624 images (70 %)
val: ../data/DataSet-17.1/val/images     # 125/624 images (20 %)
test: ../data/DataSet-17.1/test/images    # 63/624 images (10 %)

# number of classes
nc: 5

# class names
names: ['Anus', 'Pelvic␣fins', 'Rack-top␣upper',
        'Rack-top␣lower', 'Hands']


#------------ From 'hyp.scratch.yaml' ---------------------
lr0: 0.01  # initial learning rate (SGD=1E-2, Adam=1E-3)
lrf: 0.1  # final OneCycleLR learning rate (lr0 * lrf)
momentum: 0.937  # SGD momentum/Adam beta1
weight_decay: 0.0005  # optimizer weight decay 5e-4
warmup_epochs: 3.0  # warmup epochs (fractions ok)
warmup_momentum: 0.8  # warmup initial momentum
warmup_bias_lr: 0.1  # warmup initial bias lr
box: 0.05  # box loss gain
cls: 0.3  # cls loss gain
cls_pw: 1.0  # cls BCELoss positive_weight
obj: 0.7  # obj loss gain (scale with pixels)
obj_pw: 1.0  # obj BCELoss positive_weight
iou_t: 0.20  # IoU training threshold
anchor_t: 4.0 # anchor-multiple threshold
# anchors: 3  # anchors per output layer (0 to ignore)
fl_gamma: 0.0 # focal loss gamma (efficientDet default gamma=1.5)
```

```
hsv_h: 0.015  # image HSV-Hue augmentation (fraction)
hsv_s: 0.7  # image HSV-Saturation augmentation (fraction)
hsv_v: 0.4  # image HSV-Value augmentation (fraction)
degrees: 0.0  # image rotation (+/- deg)
translate: 0.2  # image translation (+/- fraction)
scale: 0.5  # image scale (+/- gain)
shear: 0.0  # image shear (+/- deg)
perspective: 0.0 # image perspective (+/- fraction),range 0-0.001
flipud: 0.0  # image flip up-down (probability)
fliplr: 0.0  # image flip left-right (probability)
mosaic: 1.0  # image mosaic (probability)
mixup: 0.0   # image mixup (probability)
copy_paste: 0.0  # image copy paste (probability)
paste_in: 0.0 # image paste (prob.), use 0 for faster training
loss_ota: 1 # use ComputeLossOTA, use 0 for faster training


#------------ From 'hyp.yaml' ---------------------
lr0: 0.01
lrf: 0.1
momentum: 0.937
weight_decay: 0.0005
warmup_epochs: 3.0
warmup_momentum: 0.8
warmup_bias_lr: 0.1
box: 0.05
cls: 0.3
cls_pw: 1.0
obj: 0.7
obj_pw: 1.0
iou_t: 0.2
anchor_t: 4.0
fl_gamma: 0.0
hsv_h: 0.015
hsv_s: 0.7
hsv_v: 0.4
degrees: 0.0
translate: 0.2
scale: 0.5
shear: 0.0
perspective: 0.0
flipud: 0.0
fliplr: 0.0
mosaic: 1.0
mixup: 0.0
copy_paste: 0.0
paste_in: 0.0
loss_ota: 1


#------------ From 'yolov7.yaml' ---------------------
# parameters
nc: 5  # number of classes
depth_multiple: 1.0  # model depth multiple
```

```
width_multiple: 1.0  # layer channel multiple

# anchors
anchors:
  - [12,16, 19,36, 40,28]  # P3/8
  - [36,75, 76,55, 72,146]  # P4/16
  - [142,110, 192,243, 459,401]  # P5/32
```

A.2 YOLOv7-tiny config

```
#------------ From 'data.yaml' ----------------------
# dataset paths
train: ../data/DataSet-17.1/train/images  # 436/624 images (70 %)
val: ../data/DataSet-17.1/val/images     # 125/624 images (20 %)
test: ../data/DataSet-17.1/test/images    # 63/624 images (10 %)

# number of classes
nc: 5

# class names
names: ['Anus', 'Pelvic␣fins', 'Rack-top␣upper',
        'Rack-top␣lower', 'Hands']


#------------ From 'hyp.scratch.tiny.yaml' ---------------------
lr0: 0.01  # initial learning rate (SGD=1E-2, Adam=1E-3)
lrf: 0.01  # final OneCycleLR learning rate (lr0 * lrf)
momentum: 0.937  # SGD momentum/Adam beta1
weight_decay: 0.0005  # optimizer weight decay 5e-4
warmup_epochs: 3.0  # warmup epochs (fractions ok)
warmup_momentum: 0.8  # warmup initial momentum
warmup_bias_lr: 0.1  # warmup initial bias lr
box: 0.05  # box loss gain
cls: 0.5  # cls loss gain
cls_pw: 1.0  # cls BCELoss positive_weight
obj: 1.0  # obj loss gain (scale with pixels)
obj_pw: 1.0  # obj BCELoss positive_weight
iou_t: 0.20  # IoU training threshold
anchor_t: 4.0  # anchor-multiple threshold
# anchors: 3  # anchors per output layer (0 to ignore)
fl_gamma: 0.0 # focal loss gamma (efficientDet default gamma=1.5)
hsv_h: 0.015  # image HSV-Hue augmentation (fraction)
hsv_s: 0.7  # image HSV-Saturation augmentation (fraction)
hsv_v: 0.4  # image HSV-Value augmentation (fraction)
degrees: 0.0  # image rotation (+/- deg)
translate: 0.1  # image translation (+/- fraction)
scale: 0.5  # image scale (+/- gain)
shear: 0.0  # image shear (+/- deg)
perspective: 0.0 # image perspective (+/- fraction),range 0-0.001
flipud: 0.0  # image flip up-down (probability)
```

```
fliplr: 0.0  # image flip left-right (probability)
mosaic: 1.0  # image mosaic (probability)
mixup: 0.05  # image mixup (probability)
copy_paste: 0.0  # image copy paste (probability)
paste_in: 0.05 # image paste (prob.), use 0 for faster training
loss_ota: 1 # use ComputeLossOTA, use 0 for faster training



#------------ From 'hyp.yaml' ---------------------
lr0: 0.01
lrf: 0.01
momentum: 0.937
weight_decay: 0.0005
warmup_epochs: 3.0
warmup_momentum: 0.8
warmup_bias_lr: 0.1
box: 0.05
cls: 0.5
cls_pw: 1.0
obj: 1.0
obj_pw: 1.0
iou_t: 0.2
anchor_t: 4.0
fl_gamma: 0.0
hsv_h: 0.015
hsv_s: 0.7
hsv_v: 0.4
degrees: 0.0
translate: 0.1
scale: 0.5
shear: 0.0
perspective: 0.0
flipud: 0.0
fliplr: 0.0
mosaic: 1.0
mixup: 0.05
copy_paste: 0.0
paste_in: 0.05
loss_ota: 1


#------------ From 'yolov7.yaml' ---------------------
# parameters
nc: 5  # number of classes
depth_multiple: 1.0  # model depth multiple
width_multiple: 1.0  # layer channel multiple

# anchors
anchors:
  - [46,25, 88,74, 68,99]         #  P3/8.
  - [182,108, 461,48, 148,1779]   # P4/16.
  - [180,157, 529,67, 177,218]    # P5/32.
```

/B Configuration for Detectron2

B.1 4cls-2kpts config

```
# Initialize dataset configuration.
self.cfg = get_cfg()
self.cfg.INPUT.FORMAT = "RGB"
self.cfg.OUTPUT_DIR = self.output_path
self.cfg.merge_from_file(
    'configs/COCO-Keypoints/keypoint_rcnn_R_50_FPN_1x.yaml')
self.cfg.MODEL.WEIGHTS = model_zoo.get_checkpoint_url(
    'COCO-Keypoints/keypoint_rcnn_R_50_FPN_1x.yaml')
self.cfg.MODEL.DEVICE = 'cuda' # if no Nvidia GPU use: 'cpu'.
# Mean of norm. RGB values of 'images-17.1/frame_000050.PNG'.
self.cfg.MODEL.PIXEL_MEAN = [0.34446954, 0.41759595, 0.48091757]
# St.dev. of norm. RGB values of 'images-17.1/frame_000050.PNG'.
self.cfg.MODEL.PIXEL_STD = [0.1987732 , 0.23223531, 0.22960765]
self.cfg.MODEL.RETINANET.NUM_CLASSES = 1 #
self.cfg.DATASETS.TRAIN = ('train',)
self.cfg.DATASETS.TEST = ('test',)
self.cfg.DATALOADER.NUM_WORKERS = 2
self.cfg.SOLVER.IMS_PER_BATCH = 2
self.cfg.SOLVER.BASE_LR = 0.01   #
self.cfg.SOLVER.MAX_ITER = 40000 #
self.cfg.SOLVER.GAMMA = 0.9 # LR multiplied by GAMMA at STEPS.
self.cfg.SOLVER.STEPS = (1100,1300,1500,1700,2000,2300,2700,
                         3100,3500,3900,4300,4800,5300,5800,
                         6300,7000,7800,8700) #
self.cfg.SOLVER.CHECKPOINT_PERIOD = 1000
self.cfg.TEST.EVAL_PERIOD = 1000
self.cfg.MODEL.KEYPOINT_ON = True
self.cfg.MODEL.thing_classes = ['Cod','Lower','Upper','Hands']
self.cfg.MODEL.keypoint_names = ["farSide", "opSide"]
self.cfg.MODEL.keypoint_flip_map = [("opSide","farSide")]
self.cfg.MODEL.keypoint_connection_rules = [
                    ("farSide", "opSide", (255,0,0))] # (R,G,B)
# IOU overlap ratios [IOU_THRESHOLD].
# Overlap threshold for an RoI to be considered background
# if < IOU_THRESHOLD and foreground if >= IOU_THRESHOLD.
self.cfg.MODEL.ROI_HEADS.IOU_THRESHOLDS = [0.5]
self.cfg.MODEL.ROI_HEADS.BATCH_SIZE_PER_IMAGE = 512
self.cfg.MODEL.ROI_HEADS.NUM_CLASSES = 4
self.cfg.MODEL.ROI_KEYPOINT_HEAD.NUM_KEYPOINTS = 2
self.cfg.MODEL.ROI_BOX_HEAD.NORM = "GN" #
self.cfg.TEST.KEYPOINT_OKS_SIGMAS = [.01, .01]
```

B.2 1cls-6kpts config

```
# Initialize dataset configuration.
self.cfg = get_cfg()
self.cfg.INPUT.FORMAT = "RGB"
self.cfg.OUTPUT_DIR = self.output_path
self.cfg.merge_from_file(
    'configs/COCO-Keypoints/keypoint_rcnn_R_50_FPN_1x.yaml')
self.cfg.MODEL.WEIGHTS = model_zoo.get_checkpoint_url(
    'COCO-Keypoints/keypoint_rcnn_R_50_FPN_1x.yaml')
self.cfg.MODEL.DEVICE = 'cuda' # if no Nvidia GPU use: 'cpu'.
# Mean of norm. RGB values of 'images-17.1/frame_000050.PNG'.
self.cfg.MODEL.PIXEL_MEAN = [0.34446954, 0.41759595, 0.48091757]
# St.dev. of norm. RGB values of 'images-17.1/frame_000050.PNG'.
self.cfg.MODEL.PIXEL_STD = [0.1987732 , 0.23223531, 0.22960765]
self.cfg.MODEL.RETINANET.NUM_CLASSES = 1 #
self.cfg.DATASETS.TRAIN = ('train',)
self.cfg.DATASETS.TEST = ('test',)
self.cfg.DATALOADER.NUM_WORKERS = 2
self.cfg.SOLVER.IMS_PER_BATCH = 2
self.cfg.SOLVER.BASE_LR = 0.01   #
self.cfg.SOLVER.MAX_ITER = 40000 #
self.cfg.SOLVER.GAMMA = 0.9 # LR multiplied by GAMMA at STEPS.
self.cfg.SOLVER.STEPS = (1100,1300,1500,1700,2000,2300,2700,
                         3100,3500,3900,4300,4800,5300,5800,
                         6300,7000,7800,8700) #
self.cfg.SOLVER.CHECKPOINT_PERIOD = 1000
self.cfg.TEST.EVAL_PERIOD = 1000
self.cfg.MODEL.KEYPOINT_ON = True
self.cfg.MODEL.thing_classes = ['Fish_in_rack']
self.cfg.MODEL.keypoint_names = [
                            'Rack-L_farside','Rack-L_opside',
                            'Rack-U_opside','Rack-U_farside',
                            'Anus','Pelvic␣fins']
self.cfg.MODEL.keypoint_flip_map = [
                            ('Rack-L_opside','Rack-L_farside'),
                            ('Rack-U_farside', 'Rack-U_opside'),
                            ('Pelvic␣fins', 'Anus')]
self.cfg.MODEL.keypoint_connection_rules = [
                  ('Rack-L_farside','Rack-L_opside', (0,0,255)),
                  ('Rack-U_opside','Rack-U_farside', (0,0,255)),
                  ("Anus", "Pelvic␣fins", (255,0,0))] # (R,G,B)
# IOU overlap ratios [IOU_THRESHOLD].
# Overlap threshold for an RoI to be considered background
# if < IOU_THRESHOLD and foreground if >= IOU_THRESHOLD.
self.cfg.MODEL.ROI_HEADS.IOU_THRESHOLDS = [0.5]
self.cfg.MODEL.ROI_HEADS.BATCH_SIZE_PER_IMAGE = 512
self.cfg.MODEL.ROI_HEADS.NUM_CLASSES = 1
self.cfg.MODEL.ROI_KEYPOINT_HEAD.NUM_KEYPOINTS = 6
self.cfg.MODEL.ROI_BOX_HEAD.NORM = "GN" #
self.cfg.TEST.KEYPOINT_OKS_SIGMAS = [.01, .01,
                                     .01, .01,
                                     .01, .01]
```

/C
Annotation data tables

C.1 YOLOv7 annotation tables

Table C.1: Data from annotations to YOLOv7, for image-ids 87 to 103 (fish #2 from 17.1), all of the same fish with an ideal cut length of 150mm: Two leftmost columns shows the annotated width of lower and upper rack top, "Belly cut" is the computed Euclidean distance in pixels between the pelvic fins and anus. The "Pixel scale" column contains the estimated image scale (at current fish position in frame) in pixels per millimeter. Next column to the right is the estimated cut length in millimeters computed from the "Belly cut" and "Pixel scale" columns. For reference, in the rightmost column, is the ideal cut length (measured by hand during collection of dataset)

Image id	Lower rack width [pixels]	Upper rack width [pixels]	Belly cut [pixels]	Pixel scale [pix./mm]	Annot. cut length [mm]	Ideal cut length [mm]
87	884.0	799.0	NaN	1.8175	NaN	150.0
88	900.0	807.0	267.9	1.8434	145.3	150.0
89	900.0	815.0	269.2	1.8521	145.3	150.0
90	909.0	818.0	269.4	1.8650	144.4	150.0
91	918.0	827.0	274.0	1.8844	145.4	150.0
92	928.0	833.0	273.0	1.9017	143.5	150.0
93	931.0	838.0	276.9	1.9104	144.9	150.0
94	940.0	846.0	277.7	1.9287	144.0	150.0
95	950.0	854.0	280.8	1.9482	144.1	150.0
96	952.0	860.0	283.7	1.9568	145.0	150.0
97	972.0	871.0	286.8	1.9903	144.1	150.0
98	976.0	876.0	286.8	2.0000	143.4	150.0
99	982.0	884.0	293.6	2.0151	145.7	150.0
100	993.0	893.0	296.4	2.0367	145.5	150.0
101	999.0	902.0	300.5	2.0529	146.4	150.0
102	1017.0	911.0	304.5	2.0821	146.3	150.0
103	1024.0	915.0	305.4	2.0940	145.8	150.0
count	17	17	16	17	16	17
mean	951.5	855.8	284.2	1.9517	145.0	150.0
var	1.836×10^3	1.374×10^3	162.0	0.0074	0.8630	0.0
std	42.85	37.07	12.73	0.0863	0.9290	0.0
min	884.0	799.0	267.9	1.8175	143.4	150.0
25%	918.0	827.0	273.7	1.8844	144.1	150.0
50%	950.0	854.0	282.2	1.9482	145.2	150.0
75%	982.0	884.0	294.3	2.0151	145.6	150.0
max	1.024×10^3	915.0	305.4	2.0940	146.4	150.0

Table C.2: Data from annotations to YOLOv7, for image-ids 257 to 271 (fish #5 from 17.1), all of the same fish with an ideal cut length of 180mm: Two leftmost columns shows the annotated width of lower and upper rack top, "Belly cut" is the computed Euclidean distance in pixels between the pelvic fins and anus. The "Pixel scale" column contains the estimated image scale (at current fish position in frame) in pixels per millimeter. Next column to the right is the estimated cut length in millimeters computed from the "Belly cut" and "Pixel scale" columns. For reference, in the rightmost column, is the ideal cut length (measured by hand during collection of dataset).

Image id	Lower rack width [pixels]	Upper rack width [pixels]	Belly cut [pixels]	Pixel scale [pix./mm]	Annot. cut length [mm]	Ideal cut length [mm]
257	899.0	813.0	326.0	1.8488	176.4	180.0
258	905.0	817.0	328.4	1.8596	176.6	180.0
259	915.0	823.0	333.7	1.8769	177.8	180.0
260	924.0	831.0	333.6	1.8952	176.0	180.0
261	931.0	835.0	336.5	1.9071	176.5	180.0
262	939.0	843.0	339.3	1.9244	176.3	180.0
263	951.0	850.0	341.2	1.9449	175.4	180.0
264	951.0	857.0	345.1	1.9525	176.7	180.0
265	956.0	863.0	348.2	1.9644	177.3	180.0
266	976.0	868.0	348.1	1.9914	174.8	180.0
267	977.0	876.0	351.1	2.0011	175.5	180.0
268	992.0	885.0	352.9	2.0270	174.1	180.0
269	1000.0	893.0	359.6	2.0443	175.9	180.0
270	1010.0	902.0	365.7	2.0648	177.1	180.0
271	1019.0	910.0	365.5	2.0832	175.5	180.0
count	15	15	15	15	15	15
mean	956.3	857.7	345.0	1.9590	176.1	180.0
var	1.463×10^3	965.8	156.5	0.0056	0.9351	0.0
std	38.24	31.08	12.51	0.0748	0.9670	0.0
min	899.0	813.0	326.0	1.8488	174.1	180.0
25%	927.5	833.0	335.1	1.9012	175.5	180.0
50%	951.0	857.0	345.1	1.9525	176.3	180.0
75%	984.5	880.5	352.0	2.0140	176.7	180.0
max	1.019×10^3	910.0	365.7	2.0832	177.8	180.0

C.2 Detectron2 4cls-2kpts annotation tables

Table C.3: Data from annotations for Det2-4cls-2kpts, for image-ids 87 to 103 (fish #2 from 17.1), all of the same fish with an ideal cut length of 150mm:
Three left-most columns contains Euclidean pixel distances for lower and upper rack top and between the pelvic fins and anus ("Belly cut"). The "Pixel scale" column contains the estimated image scale (at current fish position in frame) in pixels per millimeter, and in column is the estimated cut length in millimeters computed from the "Belly cut" and "Pixel scale" columns. For reference, in the rightmost column, is the ideal cut length (measured by hand during collection of dataset).

Image id	Lower rack width [pixels]	Upper rack width [pixels]	Belly cut [pixels]	Pixel scale [pix./mm]	Annot. cut length [mm]	Ideal cut length [mm]
87	862.0	793.0	255.0	1.7873	142.7	150.0
88	870.0	800.0	255.1	1.8035	141.4	150.0
89	875.0	807.0	259.2	1.8164	142.7	150.0
90	886.0	813.0	261.2	1.8348	142.4	150.0
91	894.0	819.0	267.2	1.8499	144.4	150.0
92	900.0	827.0	269.1	1.8650	144.3	150.0
93	909.0	832.0	271.1	1.8801	144.2	150.0
94	915.0	839.0	275.1	1.8942	145.2	150.0
95	924.0	846.0	273.1	1.9115	142.9	150.0
96	931.0	852.0	279.1	1.9255	145.0	150.0
97	938.0	860.0	281.1	1.9417	144.8	150.0
98	948.0	869.0	281.1	1.9622	143.2	150.0
99	957.1	875.0	285.1	1.9785	144.1	150.0
100	967.0	885.0	284.0	2.0000	142.0	150.0
101	974.1	891.0	286.0	2.0141	142.0	150.0
102	983.1	898.0	291.0	2.0315	143.3	150.0
103	994.2	907.0	294.0	2.0531	143.2	150.0
count	17	17	17	17	17	17
mean	925.2	847.8	274.6	1.9147	143.4	150.0
var	1.691×10^3	1.274×10^3	147.4	0.0069	1.2844	0.0
std	41.13	35.69	12.14	0.0829	1.1333	0.0
min	862.0	793.0	255.0	1.7873	141.4	150.0
25%	894.0	819.0	267.2	1.8499	142.7	150.0
50%	924.0	846.0	275.1	1.9115	143.2	150.0
75%	957.1	875.0	284.0	1.9785	144.3	150.0
max	994.2	907.0	294.0	2.0531	145.2	150.0

Table C.4: Data from annotations for Det2-4cls-2kpts, for image-ids 257 to 271 (fish #5 from 17.1), all of the same fish with an ideal cut length of 180mm: Three left-most columns contains Euclidean pixel distances for lower and upper rack top and between the pelvic fins and anus ("Belly cut"). The "Pixel scale" column contains the estimated image scale (at current fish position in frame) in pixels per millimeter, and in column is the estimated cut length in millimeters computed from the "Belly cut" and "Pixel scale" columns. For reference, in the rightmost column, is the ideal cut length (measured by hand during collection of dataset).

Image id	Lower rack width [pixels]	Upper rack width [pixels]	Belly cut [pixels]	Pixel scale [pix./mm]	Annot. cut length [mm]	Ideal cut length [mm]
257	879.0	803.0	323.6	1.8164	178.2	180.0
258	886.0	811.0	327.0	1.8326	178.4	180.0
259	892.0	817.0	336.3	1.8456	182.2	180.0
260	903.0	820.0	332.4	1.8607	178.7	180.0
261	908.0	830.0	339.0	1.8769	180.6	180.0
262	914.0	835.0	337.9	1.8888	178.9	180.0
263	926.0	842.0	342.4	1.9093	179.3	180.0
264	932.0	848.0	346.6	1.9223	180.3	180.0
265	938.0	856.0	345.9	1.9374	178.5	180.0
266	948.0	863.0	346.8	1.9557	177.3	180.0
267	959.0	871.0	352.6	1.9763	178.4	180.0
268	965.0	879.0	354.6	1.9914	178.1	180.0
269	976.1	888.0	359.5	2.0130	178.6	180.0
270	981.1	895.0	364.4	2.0260	179.8	180.0
271	996.1	903.0	361.6	2.0509	176.3	180.0
count	15	15	15	15	15	15
mean	933.6	850.7	344.7	1.9269	178.9	180.0
var	1.346×10^3	1.012×10^3	151.7	0.0055	2.0277	0.0
std	36.68	31.81	12.32	0.0739	1.4240	0.0
min	879.0	803.0	323.6	1.8164	176.3	180.0
25%	905.5	825.0	337.1	1.8688	178.3	180.0
50%	932.0	848.0	345.9	1.9223	178.6	180.0
75%	962.0	875.0	353.6	1.9838	179.6	180.0
max	996.1	903.0	364.4	2.0509	182.2	180.0

C.3 Detectron2 1cls-6kpts annotation tables

Table C.5: Data from annotations to model Det2-1cls-6kpts, for image-ids 87 to 103 (fish #2 from 17.1), all of the same fish with an ideal cut length of 150mm: Three left-most columns contains Euclidean pixel distances for lower and upper rack top and between the pelvic fins and anus ('Belly'). The 'Scale' column contains the estimated image scale (at current fish position in frame) in pixels per millimeter, and in column is the estimated cut length in millimeters computed from the 'Belly' and 'Scale' columns. For reference, in the rightmost column, is the ideal cut length (measured by hand during collection of dataset).

Image id	Lower rack width [pixels]	Upper rack width [pixels]	Belly cut [pixels]	Pixel scale [pix./mm]	Annot. cut length [mm]	Ideal cut length [mm]
87	862.0	795.1	254.0	1.7895	141.9	150.0
88	865.0	800.0	258.0	1.7981	143.5	150.0
89	873.0	806.1	260.2	1.8132	143.5	150.0
90	885.0	814.0	258.3	1.8348	140.8	150.0
91	892.0	819.0	260.1	1.8478	140.8	150.0
92	898.0	827.0	265.0	1.8629	142.3	150.0
93	908.0	833.0	266.0	1.8802	141.5	150.0
94	914.0	838.0	267.0	1.8921	141.1	150.0
95	922.1	845.0	271.0	1.9083	142.0	150.0
96	929.1	852.0	269.1	1.9234	139.9	150.0
97	936.1	861.0	274.1	1.9408	141.2	150.0
98	946.1	868.0	276.0	1.9591	140.9	150.0
99	954.1	875.0	281.1	1.9753	142.3	150.0
100	965.1	883.0	281.0	1.9958	140.8	150.0
101	974.1	890.0	285.0	2.0130	141.6	150.0
102	982.2	897.0	285.0	2.0294	140.4	150.0
103	995.1	907.0	291.0	2.0541	141.7	150.0
count	17	17	17	17.000	17	17
mean	923.6	847.7	270.7	1.9128	141.5	150.0
var	1.731×10^3	1.239×10^3	122.7	0.0069	0.9631	0.0
std	41.61	35.20	11.08	0.0829	0.9814	0.0
min	862.0	795.1	254.0	1.7895	139.9	150.0
25%	892.0	819.0	260.2	1.8478	140.8	150.0
50%	922.1	845.0	269.1	1.9083	141.5	150.0
75%	954.1	875.0	281.0	1.9753	142.0	150.0
max	995.1	907.0	291.0	2.0541	143.5	150.0

Table C.6: Data from annotations for Det2-1cls-6kpts, for image-ids 257 to 271 (fish #5 from 17.1), all of the same fish with an ideal cut length of 180mm: Three left-most columns contains Euclidean pixel distances for lower and upper rack top and between the pelvic fins and anus ("Belly cut"). The "Pixel scale" column contains the estimated image scale (at current fish position in frame) in pixels per millimeter, and in column is the estimated cut length in millimeters computed from the "Belly cut" and "Pixel scale" columns. For reference, in the rightmost column, is the ideal cut length (measured by hand during collection of dataset).

Image id	Lower rack width [pixels]	Upper rack width [pixels]	Belly cut [pixels]	Pixel scale [pix./mm]	Annot. cut length [mm]	Ideal cut length [mm]
257	880.0	805.0	320.3	1.8197	176.0	180.0
258	889.0	811.0	327.8	1.8359	178.6	180.0
259	893.0	816.0	331.9	1.8456	179.8	180.0
260	903.0	823.0	331.9	1.8640	178.1	180.0
261	910.0	829.0	336.0	1.8780	178.9	180.0
262	914.0	835.0	340.8	1.8888	180.4	180.0
263	929.1	842.0	346.7	1.9126	181.3	180.0
264	933.1	850.0	348.8	1.9255	181.2	180.0
265	939.1	856.0	351.7	1.9385	181.4	180.0
266	950.1	863.0	353.7	1.9580	180.7	180.0
267	959.1	872.0	356.6	1.9774	180.3	180.0
268	966.1	880.0	361.5	1.9936	181.3	180.0
269	978.1	888.0	361.4	2.0152	179.3	180.0
270	983.1	893.0	363.6	2.0260	179.4	180.0
271	996.1	903.0	369.4	2.0509	180.1	180.0
count	15	15	15	15.000	15	15
mean	934.9	851.1	346.8	1.9286	179.8	180.0
var	1.337×10^3	988.2	217.9	0.0054	2.1678	0.0
std	36.56	31.44	14.76	0.0734	1.4723	0.0
min	880.0	805.0	320.3	1.8197	176.0	180.0
25%	906.5	826.0	334.0	1.8710	179.1	180.0
50%	933.1	850.0	348.8	1.9255	180.1	180.0
75%	962.6	876.0	359.0	1.9855	180.9	180.0
max	996.1	903.0	369.4	2.0509	181.4	180.0

/D

Prediction data tables

D.1 YOLOv7 std. prediction data tables

Table D.1: Data from predictions of *YOLOv7 std.*, for image-ids 7 to 21 (fish #1 from preds.), all of the same fish with an ideal cut length of 125mm:
Two leftmost columns shows the predicted width of lower and upper rack-top, "Belly est. line" is the predicted Euclidean distance in pixels between the pelvic fins and anus. The "Pixel scale" column contains the estimated image scale (at current fish position in frame) in pixels per millimeter. Next column to the right is the estimated cut length in millimeters computed from the "Belly est. line" and "Pixel scale" columns. For reference, in the rightmost column, is the ideal cut length (measured by hand during collection of dataset).

Image id	Lower rack width [pixels]	Upper rack width [pixels]	Belly est. line [pixels]	Pixel scale [pix./mm]	Est. cut length [mm]	Ideal cut length [mm]
7	884	799	217.1	1.8175	119.4	125
8	894	811	220.0	1.8413	119.5	125
9	896	821	219.0	1.8542	118.1	125
10	911	826	217.0	1.8758	115.7	125
11	922	827	212.2	1.8888	112.3	125
12	929	837	NaN	1.9071	NaN	125
13	935	844	NaN	1.9212	NaN	125
14	947	853	NaN	1.9438	NaN	125
15	955	855	NaN	1.9546	NaN	125
16	963	858	NaN	1.9665	NaN	125
17	962	868	NaN	1.9762	NaN	125
18	973	875	NaN	1.9957	NaN	125
19	984	887	220.2	2.0205	109.0	125
20	993	894	227.0	2.0378	111.4	125
21	1007	901	230.0	2.0605	111.6	125
count	15	15	8	15	8	15
mean	943.7	850.4	220.3	1.9374	114.6	125
var	1.422×10^3	944.5	32.62	0.0054	16.68	0
std	37.71	30.73	5.712	0.0737	4.084	0
min	884	799	212.2	1.8175	109.0	125
25%	916.5	826.5	217.1	1.8823	111.6	125
50%	947	853	219.5	1.9438	114.0	125
75%	968	871.5	221.9	1.9860	118.4	125
max	1007	901	230.0	2.0605	119.5	125

Table D.2: Data from predictions of *YOLOv7 std.*, for image-ids 47 to 61 (fish #3 from preds.), all of the same fish with an ideal cut length of 175mm:
Two leftmost columns shows the predicted width of lower and upper rack-top, "Belly est. line" is the predicted Euclidean distance in pixels between the pelvic fins and anus. The "Pixel scale" column contains the estimated image scale (at current fish position in frame) in pixels per millimeter. Next column to the right is the estimated cut length in millimeters computed from the "Belly est. line" and "Pixel scale" columns. For reference, in the rightmost column, is the ideal cut length (measured by hand during collection of dataset). Visualization of the predictions in figure 6.8.

Image id	Lower rack width [pixels]	Upper rack width [pixels]	Belly est. line [pixels]	Pixel scale [pix./mm]	Est. cut length [mm]	Ideal cut length [mm]
47	885	804	305.0	1.8240	167.2	175
48	894	792	304.0	1.8207	167.0	175
49	896	811	307.0	1.8434	166.5	175
50	911	820	309.0	1.8693	165.3	175
51	920	821	311.0	1.8801	165.4	175
52	931	831	310.0	1.9028	162.9	175
53	931	838	310.0	1.9104	162.3	175
54	943	843	315.0	1.9287	163.3	175
55	948	852	315.0	1.9438	162.1	175
56	961	857	322.0	1.9633	164.0	175
57	968	862	322.1	1.9762	163.0	175
58	975	869	326.1	1.9914	163.8	175
59	985	875	328.1	2.0086	163.3	175
60	993	881	330.2	2.0238	163.1	175
61	1015	892	332.1	2.0594	161.3	175
count	15	15	15	15	15	15
mean	943.7	843.2	316.4	1.9297	164.0	175
var	1.518×10^3	893.6	90.72	0.0055	3.386	0
std	38.96	29.89	9.525	0.0741	1.840	0
min	885	792	304.0	1.8207	161.3	175
25%	915.5	820.5	309.5	1.8747	162.9	175
50%	943	843	315.0	1.9287	163.3	175
75%	971.5	865.5	324.1	1.9838	165.4	175
max	1015	892	332.1	2.0594	167.2	175

Table D.3: Data from predictions of *YOLOv7 std.*, for image-ids 94 to 110 (fish #5 from preds.), all of the same fish with an ideal cut length of 180mm:
Two leftmost columns shows the predicted width of lower and upper rack-top, "Belly est. line" is the predicted Euclidean distance in pixels between the pelvic fins and anus. The "Pixel scale" column contains the estimated image scale (at current fish position in frame) in pixels per millimeter. Next column to the right is the estimated cut length in millimeters computed from the "Belly est. line" and "Pixel scale" columns. For reference, in the rightmost column, is the ideal cut length (measured by hand during collection of dataset). Visualization of the predictions in figure **??**.

Image id	Lower rack width [pixels]	Upper rack width [pixels]	Belly est. line [pixels]	Pixel scale [pix./mm]	Est. cut length [mm]	Ideal cut length [mm]
94	878	792	309.3	1.8035	171.5	180
95	879	800	312.3	1.8132	172.2	180
96	887	799	318.6	1.8207	175	180
97	900	810	319.8	1.8467	173.2	180
98	904	821	320.5	1.8629	172.0	180
99	922	821	325.4	1.8823	172.9	180
100	927	827	324.2	1.8942	171.1	180
101	933	837	326.2	1.9114	170.6	180
102	943	844	332.3	1.9298	172.2	180
103	946	851	332.3	1.9406	171.2	180
104	960	855	334.3	1.9600	170.5	180
105	966	861	337.2	1.9730	170.9	180
106	975	866	338.2	1.9881	170.1	180
107	984	876	339.1	2.0086	168.8	180
108	988	884	343.1	2.0216	169.7	180
109	992	891	349.1	2.0335	171.7	180
110	NaN	898	352.1	NaN	NaN	180
count	16	17	17	16	16	17
mean	936.5	843.1	330.2	1.9181	171.5	180
var	1.527×10^3	1.118×10^3	149.5	0.0057	2.158	0
std	39.08	33.44	12.23	0.0758	1.469	0
min	878	792	309.3	1.8035	168.8	180
25%	903	821	320.5	1.8588	170.6	180
50%	938	844	332.3	1.9206	171.4	180
75%	968.3	866	338.2	1.9768	172.2	180
max	992	898	352.1	2.0335	175.0	180

Table D.4: Data from predictions of *YOLOv7 std.*, for image-ids 167 to 181 (fish #8 from preds.), all of the same fish with an ideal cut length of 165mm:
Two leftmost columns shows the predicted width of lower and upper rack-top, "Belly est. line" is the predicted Euclidean distance in pixels between the pelvic fins and anus. The "Pixel scale" column contains the estimated image scale (at current fish position in frame) in pixels per millimeter. Next column to the right is the estimated cut length in millimeters computed from the "Belly est. line" and "Pixel scale" columns. For reference, in the rightmost column, is the ideal cut length (measured by hand during collection of dataset). Visualization of the predictions in figure 6.9.

Image id	Lower rack width [pixels]	Upper rack width [pixels]	Belly est. line [pixels]	Pixel scale [pix./mm]	Est. cut length [mm]	Ideal cut length [mm]
167	887	800	285.2	1.8218	156.5	165
168	897	799	287.4	1.8315	156.9	165
169	901	812	288.7	1.8499	156.0	165
170	924	820	295.4	1.8834	156.9	165
171	933	829	295.4	1.9028	155.3	165
172	926	833	298.7	1.8996	157.3	165
173	941	838	301.2	1.9212	156.8	165
174	948	846	302.4	1.9374	156.1	165
175	953	851	310.0	1.9482	159.1	165
176	966	857	313.3	1.9687	159.1	165
177	972	865	311.5	1.9838	157.0	165
178	976	872	309.7	1.9957	155.2	165
179	985	883	307.0	2.0173	152.2	165
180	994	889	311.9	2.0335	153.4	165
181	995	897	314.6	2.0432	154.0	165
count	15	15	15	15	15	15
mean	946.5	846.1	302.2	1.9359	156.1	165
var	1.231×10^3	977.2	99.86	0.0051	3.646	0
std	35.09	31.26	9.993	0.0714	1.910	0
min	887	799	285.2	1.8218	152.2	165
25%	925	824.5	295.4	1.8915	155.2	165
50%	948	846	302.4	1.9374	156.5	165
75%	974	868.5	310.8	1.9897	157.0	165
max	995	897	314.6	2.0432	159.1	165

D.2 YOLOv7 tiny prediction data tables

Table D.5: Data from predictions of *YOLOv7 tiny*, for image-ids 7 to 21 (fish #1 from preds.), all of the same fish with an ideal cut length of 125mm: Two leftmost columns shows the predicted width of lower and upper rack-top, "Belly est. line" is the predicted Euclidean distance in pixels between the pelvic fins and anus. The "Pixel scale" column contains the estimated image scale (at current fish position in frame) in pixels per millimeter. Next column to the right is the estimated cut length in millimeters computed from the "Belly est. line" and "Pixel scale" columns. For reference, in the rightmost column, is the ideal cut length (measured by hand during collection of dataset).

Image id	Lower rack width [pixels]	Upper rack width [pixels]	Belly est. line [pixels]	Pixel scale [pix./mm]	Est. cut length [mm]	Ideal cut length [mm]
7	881	799	217.0	1.8143	119.6	125
8	890	810	219.1	1.8359	119.4	125
9	891	830	215.0	1.8585	115.7	125
10	908	827	220.0	1.8737	117.4	125
11	913	838	214.0	1.8909	113.2	125
12	926	862	206.2	1.9309	106.8	125
13	922	862	203.6	1.9266	105.7	125
14	929	868	204.1	1.9406	105.2	125
15	943	874	NaN	1.9622	NaN	125
16	955	875	207.6	1.9762	105.1	125
17	971	878	211.4	1.9968	105.9	125
18	974	883	213.3	2.0054	106.4	125
19	988	897	223.0	2.0356	109.6	125
20	995	902	223.0	2.0486	108.9	125
21	1015	905	227.0	2.0734	109.5	125
count	15	15	14	15	14	15
mean	940.1	860.7	214.6	1.9446	110.6	125
var	1.720×10^3	1.085×10^3	55.02	0.0063	29.41	0
std	41.47	32.94	7.418	0.0793	5.423	0
min	881	799	203.6	1.8143	105.1	125
25%	910.5	834	208.6	1.8823	106.0	125
50%	929	868	214.5	1.9406	109.2	125
75%	972.5	880.5	219.8	2.0011	115.1	125
max	1015	905	227.0	2.0734	119.6	125

Table D.6: Data from predictions of *YOLOv7 tiny*, for image-ids 47 to 61 (fish #3 from preds.), all of the same fish with an ideal cut length of 175mm:
Two leftmost columns shows the predicted width of lower and upper rack-top, "Belly est. line" is the predicted Euclidean distance in pixels between the pelvic fins and anus. The "Pixel scale" column contains the estimated image scale (at current fish position in frame) in pixels per millimeter. Next column to the right is the estimated cut length in millimeters computed from the "Belly est. line" and "Pixel scale" columns. For reference, in the rightmost column, is the ideal cut length (measured by hand during collection of dataset).

Image id	Lower rack width [pixels]	Upper rack width [pixels]	Belly est. line [pixels]	Pixel scale [pix./mm]	Est. cut length [mm]	Ideal cut length [mm]
47	891	814	311.0	1.8413	168.9	175
48	890	811	310.0	1.8369	168.8	175
49	900	827	312.0	1.8650	167.3	175
50	902	830	313.1	1.8704	167.4	175
51	901	835	314.0	1.8747	167.5	175
52	911	861	312.0	1.9136	163.1	175
53	913	852	314.0	1.9060	164.7	175
54	928	851	316.0	1.9212	164.5	175
55	931	857	317.0	1.9309	164.2	175
56	942	863	326.0	1.9492	167.2	175
57	953	877	327.0	1.9762	165.5	175
58	959	881	329.0	1.9870	165.6	175
59	962	891	335.0	2.0011	167.4	175
60	984	891	337.0	2.0248	166.4	175
61	999	907	340.0	2.0583	165.2	175
count	15	15	15	15	15	15
mean	931.1	856.5	320.9	1.9305	166.2	175
var	1.178×10^3	855.4	108.8	0.0046	2.995	0
std	34.33	29.25	10.43	0.0679	1.731	0
min	890	811	310.0	1.8369	163.1	175
25%	901.5	832.5	312.6	1.8726	165.0	175
50%	928	857	316.0	1.9212	166.4	175
75%	956	879	328.0	1.9816	167.4	175
max	999	907	340.0	2.0583	168.9	175

Table D.7: Data from predictions of *YOLOv7 tiny*, for image-ids 94 to 110 (fish #5 from preds.), all of the same fish with an ideal cut length of 180mm:
Two leftmost columns shows the predicted width of lower and upper rack-top, "Belly est. line" is the predicted Euclidean distance in pixels between the pelvic fins and anus. The "Pixel scale" column contains the estimated image scale (at current fish position in frame) in pixels per millimeter. Next column to the right is the estimated cut length in millimeters computed from the "Belly est. line" and "Pixel scale" columns. For reference, in the rightmost column, is the ideal cut length (measured by hand during collection of dataset).

Image id	Lower rack width [pixels]	Upper rack width [pixels]	Belly est. line [pixels]	Pixel scale [pix./mm]	Est. cut length [mm]	Ideal cut length [mm]
94	884	812	308.3	1.8315	168.3	180
95	895	820	314.4	1.8521	169.7	180
96	889	825	317.5	1.8510	171.5	180
97	903	838	319.8	1.8801	170.1	180
98	903	835	322.5	1.8769	171.8	180
99	908	850	323.3	1.8985	170.3	180
100	914	845	325.2	1.8996	171.2	180
101	915	855	326.1	1.9114	170.6	180
102	927	856	327.2	1.9255	169.9	180
103	939	862	334.3	1.9449	171.9	180
104	935	875	335.1	1.9546	171.4	180
105	954	871	336.1	1.9708	170.5	180
106	959	879	340.1	1.9849	171.4	180
107	976	894	340.1	2.0194	168.4	180
108	982	894	345.1	2.0259	170.3	180
109	993	897	350.1	2.0410	171.5	180
110	1024	911	350.1	2.0896	167.5	180
count	17	17	17	17	17	17
mean	935.3	859.9	330.3	1.9387	170.4	180
var	1.633×10^3	849.8	152.5	0.0056	1.676	0
std	40.41	29.15	12.35	0.0745	1.295	0
min	884	812	308.3	1.8315	167.5	180
25%	903	838	322.5	1.8801	169.9	180
50%	927	856	327.2	1.9255	170.5	180
75%	959	879	340.1	1.9849	171.4	180
max	1024	911	350.1	2.0896	171.9	180

Table D.8: Data from predictions of *YOLOv7 tiny*, for image-ids 167 to 181 (fish #8 from preds.), all of the same fish with an ideal cut length of 165mm: Two leftmost columns shows the predicted width of lower and upper rack-top, "Belly est. line" is the predicted Euclidean distance in pixels between the pelvic fins and anus. The "Pixel scale" column contains the estimated image scale (at current fish position in frame) in pixels per millimeter. Next column to the right is the estimated cut length in millimeters computed from the "Belly est. line" and "Pixel scale" columns. For reference, in the rightmost column, is the ideal cut length (measured by hand during collection of dataset).

Image id	Lower rack width [pixels]	Upper rack width [pixels]	Belly est. line [pixels]	Pixel scale [pix./mm]	Est. cut length [mm]	Ideal cut length [mm]
167	895	821	282.4	1.8531	152.4	165
168	900	845	289.2	1.8844	153.5	165
169	905	835	288.5	1.8790	153.5	165
170	914	857	292.4	1.9125	152.9	165
171	920	861	295.3	1.9233	153.6	165
172	923	871	295.5	1.9374	152.5	165
173	923	875	302.9	1.9417	156.0	165
174	934	870	304.3	1.9482	156.2	165
175	940	870	314.0	1.9546	160.6	165
176	954	870	314.0	1.9698	159.4	165
177	969	888	311.5	2.0054	155.4	165
178	968	890	312.7	2.0065	155.8	165
179	986	896	313.3	2.0324	154.1	165
180	992	913	312.9	2.0572	152.1	165
181	1018	915	315.5	2.0875	151.1	165
count	15	15	15	15	15	15
mean	942.7	871.8	303.0	1.9595	154.6	165
var	1.368×10^3	696.6	130.2	0.0045	7.124	0
std	36.99	26.39	11.41	0.0674	2.669	0
min	895	821	282.4	1.8531	151.1	165
25%	917	859	293.9	1.9179	152.7	165
50%	934	870	304.3	1.9482	153.6	165
75%	968.5	889	313.1	2.0059	155.9	165
max	1018	915	315.5	2.0875	160.69	165

D.3 Prediction tables for Det2-4cls-2kpts

Table D.9: Data from predictions of *Det2-4cls-2kpts* (training201-preds1), for image-ids 7 to 21 (fish #1 from preds.), all of the same fish with an ideal cut length of 125mm: Three left-most columns contains Euclidean pixel distances for lower and upper rack-top and between the pelvic fins and anus ("Belly est. line"). The "Pixel scale" column contains the estimated image scale (at current fish position in frame) in pixels per millimeter, and in column is the estimated cut length in millimeters computed from the "Belly est. line" and "Pixel scale" columns. For reference, in the rightmost column, is the ideal cut length (measured by hand during collection of dataset).

Image id	Lower rack width [pixels]	Upper rack width [pixels]	Belly est. line [pixels]	Pixel scale [pix./mm]	Est. cut length [mm]	Ideal cut length [mm]
6	684.4	NaN	180.7	NaN	NaN	125
7	646.9	NaN	180.9	NaN	NaN	125
8	738.9	NaN	177.7	NaN	NaN	125
9	706.6	NaN	172.2	NaN	NaN	125
10	0.0	NaN	177.3	NaN	NaN	125
11	0.0	NaN	168.7	NaN	NaN	125
12	2.2	694.3	164.0	0.7521	218.0	125
13	0.0	28.9	135.8	0.0312	4.360×10^3	125
14	1.1	0.0	136.0	0.0012	1.142×10^5	125
15	NaN	0.0	1.1	NaN	NaN	125
16	803.9	695.9	169.9	1.6197	104.9	125
17	2.2	28.9	176.8	0.0336	5.270×10^3	125
18	NaN	748.8	177.4	NaN	NaN	125
19	NaN	686.4	181.2	NaN	NaN	125
20	NaN	694.0	185.8	NaN	NaN	125
count	11	9	15	5	5	15
mean	326.0	397.5	159.0	0.4875	2.483×10^4	125
var	1.409×10^5	1.325×10^5	2.136×10^3	0.5007	2.502×10^9	0
std	375.4	364.0	46.22	0.7076	5.002×10^4	0
min	0	0	1.104	0.0012	104.9	125
25%	0.5515	28.85	166.3	0.0312	218.0	125
50%	2.209	686.4	176.8	0.0336	4.360×10^3	125
75%	695.5	694.3	179.2	0.7521	5.269×10^3	125
max	803.9	748.8	185.8	1.6197	1.142×10^5	125

Table D.10: Data from predictions of *Det2-4cls-2kpts* (training201-preds1), for image-ids 47 to 61 (fish #3 from preds.), all of the same fish with an ideal cut length of 175mm: Three left-most columns contains Euclidean pixel distances for lower and upper rack-top and between the pelvic fins and anus ("Belly est. line"). The "Pixel scale" column contains the estimated image scale (at current fish position in frame) in pixels per millimeter, and in column is the estimated cut length in millimeters computed from the "Belly est. line" and "Pixel scale" columns. For reference, in the rightmost column, is the ideal cut length (measured by hand during collection of dataset).

Image id	Lower rack width [pixels]	Upper rack width [pixels]	Belly est. line [pixels]	Pixel scale [pix./mm]	Est. cut length [mm]	Ideal cut length [mm]
45	NaN	224.3	294.4	NaN	NaN	175
46	663.6	NaN	NaN	NaN	NaN	175
47	717.6	NaN	268.6	NaN	NaN	175
48	1.1	0.0	273.0	0.0012	2.279×10^5	175
49	701.3	NaN	264.2	NaN	NaN	175
50	1.1	36.4	271.8	0.0405	6.706×10^3	175
51	0.0	NaN	274.0	NaN	NaN	175
52	715.3	1.1	280.9	0.7736	363.0	175
53	757.5	0.0	274.2	0.8181	335.2	175
54	NaN	0.0	273.1	NaN	NaN	175
55	NaN	0.0	277.4	NaN	NaN	175
56	2.2	603.5	285.2	0.6541	436.0	175
57	4.4	0.0	289.5	0.0047	6.105×10^4	175
58	NaN	1.6	289.6	NaN	NaN	175
59	NaN	729.9	300.6	NaN	NaN	175
count	10	11	14	6	6	15
mean	356.4	145.2	279.7	0.3820	4.947×10^4	175
var	1.403×10^5	7.168×10^4	112.1	0.1643	8.207×10^9	0
std	374.5	267.7	10.59	0.4053	9.059×10^4	0
min	0	0	264.2	0.0012	335.2	175
25%	1.383	0	273.0	0.0137	381.3	175
50%	334.0	1.109	275.8	0.3473	3.571×10^3	175
75%	711.8	130.4	288.4	0.7437	4.747×10^4	175
max	757.5	729.9	300.6	0.8181	2.279×10^5	175

Table D.11: Data from predictions of *Det2-4cls-2kpts* (training201-preds1), for image-ids 94 to 110 (fish #5 from preds.), all of the same fish with an ideal cut length of 180mm: Three left-most columns contains Euclidean pixel distances for lower and upper rack-top and between the pelvic fins and anus ("Belly est. line"). The "Pixel scale" column contains the estimated image scale (at current fish position in frame) in pixels per millimeter, and in column is the estimated cut length in millimeters computed from the "Belly est. line" and "Pixel scale" columns. For reference, in the rightmost column, is the ideal cut length (measured by hand during collection of dataset).

Image id	Lower rack width [pixels]	Upper rack width [pixels]	Belly est. line [pixels]	Pixel scale [pix./mm]	Est. cut length [mm]	Ideal cut length [mm]
90	691.3	0.0	1.6	0.7465	2.1	180
91	698.3	NaN	1.1	NaN	NaN	180
92	1.1	NaN	1.1	NaN	NaN	180
93	713.5	NaN	267.5	NaN	NaN	180
94	732.7	NaN	270.0	NaN	NaN	180
95	692.8	NaN	274.7	NaN	NaN	180
96	703.6	0.0	273.4	0.7598	359.9	180
97	755.4	534.9	275.4	1.3934	197.7	180
98	1.1	0.0	276.6	0.0012	2.314×10^5	180
99	0.0	NaN	280.9	NaN	NaN	180
100	0.0	701.2	283.0	0.7573	373.7	180
101	1.1	0.0	289.5	0.0012	2.438×10^5	180
102	787.7	27.7	293.0	0.8806	332.7	180
103	1.1	0.0	289.5	0.0012	2.434×10^5	180
104	1.1	1.1	289.5	0.0024	1.208×10^5	180
105	1.1	0.0	297.6	0.0012	2.501×10^5	180
106	NaN	747.2	297.2	NaN	NaN	180
count	16	11	17	10	10	17
mean	361.4	182.9	233.0	0.4545	1.091×10^5	180
var	1.392×10^5	9.688×10^4	1.231×10^4	0.2622	1.449×10^{10}	0
std	373.1	311.3	111.0	0.5120	1.204×10^5	0
min	0	0	1.105	0.0012	2.096	180
25%	1.102	0	270.0	0.0012	339.5	180
50%	346.2	0	276.6	0.3745	6.058×10^4	180
75%	706.1	281.3	289.5	0.7592	2.404×10^5	180
max	787.7	747.2	297.6	1.3934	2.501×10^5	180

Table D.12: Data from predictions of *Det2-4cls-2kpts* (training201-preds1), for image-ids 167 to 181 (fish #8 from preds.), all of the same fish with an ideal cut length of 165mm: Three left-most columns contains Euclidean pixel distances for lower and upper rack-top and between the pelvic fins and anus ("Belly est. line"). The "Pixel scale" column contains the estimated image scale (at current fish position in frame) in pixels per millimeter, and in column is the estimated cut length in millimeters computed from the "Belly est. line" and "Pixel scale" columns. For reference, in the rightmost column, is the ideal cut length (measured by hand during collection of dataset).

Image id	Lower rack width [pixels]	Upper rack width [pixels]	Belly est. line [pixels]	Pixel scale [pix./mm]	Est. cut length [mm]	Ideal cut length [mm]
161	NaN	NaN	307.3	NaN	NaN	165
162	1.1	1.1	2.2	0.0024	931.7	165
163	706.6	0.0	1.1	0.7631	1.4	165
164	1.6	20.0	1.1	0.0233	47.4	165
165	724.3	1.6	1.1	0.7838	1.4	165
166	0.0	1.1	247.7	0.0012	2.084×10^5	165
167	738.1	1.1	253.6	0.7982	317.7	165
168	714.0	NaN	252.1	NaN	NaN	165
169	738.3	NaN	254.4	NaN	NaN	165
170	758.5	0.0	258.9	0.8192	316.0	165
171	1.1	0.0	255.9	0.0012	2.155×10^5	165
172	0.0	521.6	269.1	0.5633	477.7	165
173	0.0	529.3	266.1	0.5716	465.7	165
174	1.1	528.8	264.3	0.5722	461.9	165
175	2.2	539.7	269.7	0.5852	460.9	165
count	14	12	15	12	12	15
mean	313.3	178.7	193.6	0.4570	3.561×10^4	165
var	1.403×10^5	6.730×10^4	1.459×10^4	0.1193	6.785×10^9	0
std	374.6	259.4	120.8	0.3454	8.237×10^4	0
min	0	0	1.102	0.0012	1.415	165
25%	1.101	0.8154	124.9	0.0180	248.9	165
50%	1.877	1.333	254.4	0.5719	461.4	165
75%	721.7	523.4	265.2	0.7683	591.2	165
max	758.5	539.7	307.3	0.8192	2.155×10^5	165

D.4 Prediction tables for Det2-1cls-6kpts

Table D.13: Data from predictions of *Det2-1cls-6kpts* (training202-preds4), for image-ids 7 to 21 (fish #1 from preds.), all of the same fish with an ideal cut length of 125mm:
Three left-most columns contains Euclidean pixel distances for lower and upper rack-top and between the pelvic fins and anus ('Belly'). The 'Scale' column contains the estimated image scale (at current fish position in frame) in pixels per millimeter, and in column is the estimated cut length in millimeters computed from the 'Belly' and 'Scale' columns. For reference, in the rightmost column, is the ideal cut length (measured by hand during collection of dataset).

Image id	Lower rack width [pixels]	Upper rack width [pixels]	Belly est. line [pixels]	Pixel scale [pix./mm]	Est. cut length [mm]	Ideal cut length [mm]
7	729.0	675.7	182.0	1.5169	120.0	125
8	727.5	679.9	174.1	1.5198	114.6	125
9	744.8	680.4	173.2	1.5391	112.5	125
10	751.5	696.0	160.0	1.5632	102.3	125
11	750.2	698.0	167.7	1.5639	107.3	125
12	758.9	708.9	154.6	1.5850	97.5	125
13	754.7	711.3	125.8	1.5831	79.5	125
14	788.0	714.7	110.2	1.6228	67.9	125
15	769.7	723.1	43.9	1.6122	27.2	125
16	785.6	725.6	140.2	1.6320	85.9	125
17	805.5	730.1	156.5	1.6583	94.4	125
18	795.4	728.8	160.7	1.6460	97.6	125
19	797.9	737.8	175.9	1.6584	106.1	125
20	831.3	744.8	186.7	1.7021	109.7	125
21	839.3	752.7	186.7	1.7193	108.6	125
count	15	15	15	15	15	15
mean	775.3	713.9	153.2	1.6082	95.40	125
var	1.187×10^3	577.2	1.391×10^3	0.0039	550.2	0.0
std	34.45	24.03	37.29	0.0624	23.46	0.0
min	727.5	675.7	43.87	1.5169	27.21	125
25%	750.9	697.0	147.4	1.5636	90.14	125
50%	769.7	714.7	160.7	1.6122	102.3	125
75%	796.6	729.4	175.0	1.6522	109.1	125
max	839.3	752.7	186.7	1.7193	120.0	125

Table D.14: Data from predictions of *Det2-1cls-6kpts* (training202-preds4), for image-ids 47 to 61 (fish #3 from preds.), all of the same fish with an ideal cut length of 175mm: Three left-most columns contains Euclidean pixel distances for lower and upper rack-top and between the pelvic fins and anus ("Belly est. line"). The "Pixel scale" column contains the estimated image scale (at current fish position in frame) in pixels per millimeter, and in column is the estimated cut length in millimeters computed from the "Belly est. line" and "Pixel scale" columns. For reference, in the rightmost column, is the ideal cut length (measured by hand during collection of dataset).

Image id	Lower rack width [pixels]	Upper rack width [pixels]	Belly est. line [pixels]	Pixel scale [pix./mm]	Est. cut length [mm]	Ideal cut length [mm]
47	731.2	665.8	265.3	1.5086	175.8	175
48	736.5	675.5	266.2	1.5248	174.6	175
49	749.3	680.5	268.6	1.5440	174.0	175
50	747.7	692.3	274.0	1.5550	176.2	175
51	754.7	689.2	271.9	1.5594	174.4	175
52	762.7	706.0	283.1	1.5861	178.5	175
53	753.3	705.6	269.6	1.5754	171.1	175
54	784.3	717.7	278.4	1.6220	171.7	175
55	777.8	714.5	275.2	1.6116	170.8	175
56	767.9	717.9	273.0	1.6045	170.1	175
57	808.7	722.2	282.9	1.6532	171.1	175
58	787.0	723.7	287.5	1.6315	176.2	175
59	821.1	733.5	291.9	1.6789	173.8	175
60	831.3	740.3	298.6	1.6973	175.9	175
61	829.8	752.2	299.5	1.7084	175.3	175
count	15	15	15	15	15	15
mean	776.2	709.1	279.1	1.6040	174.0	175
var	1.111×10^3	608.4	124.9	0.0038	6.186	0
std	33.33	24.67	11.17	0.0617	2.487	0
min	731.2	665.8	265.3	1.5086	170.1	175
25%	751.3	690.8	270.8	1.5572	171.4	175
50%	767.9	714.5	275.2	1.6045	174.4	175
75%	797.9	723.0	285.3	1.6423	175.9	175
max	831.3	752.2	299.5	1.7084	178.5	175

Table D.15: Data from predictions of *Det2-1cls-6kpts* (training202-preds4), for image-ids 94 to 110 (fish #5 from preds.), all of the same fish with an ideal cut length of 180mm: Three left-most columns contains Euclidean pixel distances for lower and upper rack-top and between the pelvic fins and anus ("Belly est. line"). The "Pixel scale" column contains the estimated image scale (at current fish position in frame) in pixels per millimeter, and in column is the estimated cut length in millimeters computed from the "Belly est. line" and "Pixel scale" columns. For reference, in the rightmost column, is the ideal cut length (measured by hand during collection of dataset).

Image id	Lower rack width [pixels]	Upper rack width [pixels]	Belly est. line [pixels]	Pixel scale [pix./mm]	Est. cut length [mm]	Ideal cut length [mm]
94	723.1	667.7	268.7	1.5019	178.9	180
95	726.8	672.4	266.8	1.5109	176.6	180
96	734.4	676.7	270.1	1.5239	177.2	180
97	730.9	683.2	277.8	1.5270	181.9	180
98	758.2	690.5	272.6	1.5645	174.2	180
99	747.5	698.7	287.5	1.5618	184.1	180
100	755.3	706.5	286.2	1.5786	181.3	180
101	763.1	704.3	287.3	1.5846	181.3	180
102	787.0	711.4	298.6	1.6182	184.5	180
103	795.0	716.2	295.4	1.6320	181.0	180
104	774.3	718.8	297.4	1.6123	184.5	180
105	780.3	719.3	286.4	1.6194	176.9	180
106	801.5	725.9	300.9	1.6495	182.4	180
107	808.1	732.4	313.0	1.6636	188.1	180
108	836.9	741.5	315.3	1.7046	185.0	180
109	811.1	749.0	309.6	1.6847	183.8	180
110	834.8	754.9	308.7	1.7168	179.8	180
count	17	17	17	17	17	17
mean	774.6	710.0	290.7	1.6032	181.3	180
var	1.329×10^3	692.2	253.1	0.0045	13.27	0
std	36.45	26.31	15.91	0.0671	3.643	0
min	723.1	667.7	266.8	1.5019	174.2	180
25%	747.5	690.5	277.8	1.5618	178.9	180
50%	774.3	711.4	287.5	1.6123	181.3	180
75%	801.5	725.9	300.9	1.6495	184.1	180
max	836.9	754.9	315.3	1.7168	188.1	180

Table D.16: Data from predictions of *Det2-1cls-6kpts* (training202-preds4), for image-ids 167 to 181 (fish #8 from preds.), all of the same fish with an ideal cut length of 165mm: Three left-most columns contains Euclidean pixel distances for lower and upper rack-top and between the pelvic fins and anus ("Belly est. line"). The "Pixel scale" column contains the estimated image scale (at current fish position in frame) in pixels per millimeter, and in column is the estimated cut length in millimeters computed from the "Belly est. line" and "Pixel scale" columns. For reference, in the rightmost column, is the ideal cut length (measured by hand during collection of dataset).

Image id	Lower rack width [pixels]	Upper rack width [pixels]	Belly est. line [pixels]	Pixel scale [pix./mm]	Est. cut length [mm]	Ideal cut length [mm]
167	723.5	674.7	249.0	1.5099	164.9	165
168	726.3	519.0	261.0	1.3448	194.1	165
169	745.8	523.8	253.3	1.3711	184.8	165
170	741.6	524.0	264.5	1.3668	193.5	165
171	760.5	701.7	259.0	1.5790	164.0	165
172	760.4	707.1	273.3	1.5847	172.5	165
173	781.5	709.3	269.1	1.6099	167.2	165
174	792.7	715.0	276.1	1.6282	169.5	165
175	806.4	726.5	267.3	1.6553	161.5	165
176	775.7	549.3	270.2	1.4309	188.9	165
177	792.2	721.2	271.2	1.6344	165.9	165
178	795.7	732.5	278.7	1.6503	168.8	165
179	828.8	735.6	273.2	1.6894	161.7	165
180	836.5	746.8	272.1	1.7099	159.2	165
181	840.4	571.8	271.1	1.5251	177.7	165
count	15	15	15	15	15	15
mean	780.5	657.2	267.3	1.5526	172.9	165
var	1.427×10^3	8.082×10^3	70.45	0.0149	142.1	0
std	37.78	89.90	8.394	0.1222	11.92	0
min	723.5	519.0	249.0	1.3448	159.2	165
25%	753.1	560.6	262.8	1.4704	164.5	165
50%	781.5	707.1	270.2	1.5847	168.8	165
75%	801.0	723.8	272.7	1.6424	181.3	165
max	840.4	746.8	278.7	1.7099	194.1	165

Bibliography

[1] U. Johansen, H. Bull-Berg, L. H. Vik, A. M. Stokka, R. Richardsen, and U. Winther, "The Norwegian seafood industry – Importance for the national economy," *Marine Policy*, vol. 110, p. 103 561, Dec. 2019, ISSN: 0308-597X. DOI: 10.1016/j.marpol.2019.103561.

[2] *What is Computer Vision? | IBM*, [Online; accessed 17. Mar. 2023], Mar. 2023. [Online]. Available: https://www.ibm.com/topics/computer-vision.

[3] *What is Machine Learning? | IBM*, [Online; accessed 17. Mar. 2023], Mar. 2023. [Online]. Available: https://www.ibm.com/topics/machine-learning.

[4] *Stingray Marine Solutions AS - Bedre dyrehelse. Økt kunnskap. Uten håndtering av fisken*, [Online; accessed 3. Mar. 2023], May 2021. [Online]. Available: https://www.stingray.no.

[5] *Deep Vision - Realtime sampling and analysis of marine life*, [Online; accessed 5. Mar. 2023], Mar. 2023. [Online]. Available: https://www.deepvision.no.

[6] *CodCluster*, [Online; accessed 6. Mar. 2023], Mar. 2023. [Online]. Available: https://www.codcluster.no.

[7] *FHF – Norwegian Seafood Research Fund*, [Online; accessed 6. Mar. 2023], Mar. 2023. [Online]. Available: https://www.fhf.no/fhf/about-fhf-english.

[8] J. Henriksen, *FYS-3740 Project paper in applied physics and mathematics: Using Machine Learning in the fish processing industry — A feasibility study*, [Unpublished, student's project report, Univertsity of Tromsø], Dec. 2021.

[9] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao, "YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors," *arXiv*, Jul. 2022. DOI: 10.48550/arXiv.2207.02696. eprint: 2207.02696. [Online]. Available: https://arxiv.org/abs/2207.02696.

[10] Y. Wu, A. Kirillov, F. Massa, W.-Y. Lo, and R. Girshick, *Detectron2*, https://github.com/facebookresearch/detectron2, 2019.

[11] WongKinYiu, *yolov7*, [Online; accessed 25. Mar. 2023], Mar. 2023. [Online]. Available: https://github.com/WongKinYiu/yolov7.

[12] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," *arXiv*, Jun. 2015. DOI: 10.48550/arXiv.1506.02640. eprint: 1506.02640. [Online]. Available: https://arxiv.org/abs/1506.02640.

[13] A. Toshev and C. Szegedy, "DeepPose: Human Pose Estimation via Deep Neural Networks," *arXiv*, Dec. 2013. DOI: 10.1109/CVPR.2014.214. eprint: 1312.4659. [Online]. Available: https://arxiv.org/abs/1312.4659.

[14] S.-E. Wei, V. Ramakrishna, T. Kanade, and Y. Sheikh, "Convolutional Pose Machines," *arXiv*, Jan. 2016. DOI: 10.48550/arXiv.1602.00134. eprint: 1602.00134. [Online]. Available: https://arxiv.org/abs/1602.00134.

[15] J. Rong, G. Dai, and P. Wang, "A peduncle detection method of tomato for autonomous harvesting," *Complex Intell. Syst.*, vol. 8, no. 4, pp. 2955–2969, Aug. 2022, ISSN: 2198-6053. DOI: 10.1007/s40747-021-00522-7.

[16] Q. Sun, X. Chai, Z. Zeng, G. Zhou, and T. Sun, "Multi-level feature fusion for fruit bearing branch keypoint detection," *Comput. Electron. Agric.*, vol. 191, p. 106 479, Dec. 2021, ISSN: 0168-1699. DOI: 10.1016/j.compag.2021.106479.

[17] T. Tsironi, "Editorial: Improving fish from catch to the consumer: Post harvest handling, processing, packaging, transportation and storage," *Aquaculture and Fisheries*, vol. 8, no. 4, pp. 363–364, Jul. 2023, ISSN: 2468-550X. DOI: 10.1016/j.aaf.2022.12.005.

[18] J. Labra, M. D. Zuniga, J. Rebolledo, *et al.*, "Robust automatic net damage detection and tracking on real aquaculture environment using computer vision," *Aquacult. Eng.*, vol. 101, p. 102 323, May 2023, ISSN: 0144-8609. DOI: 10.1016/j.aquaeng.2023.102323.

[19] C. Choi, M. Kampffmeyer, N. O. Handegard, A.-B. Salberg, and R. Jenssen, "Deep Semisupervised Semantic Segmentation in Multifrequency Echosounder Data," *IEEE J. Oceanic Eng.*, pp. 1–17, Feb. 2023, ISSN: 1558-1691. DOI: 10.1109/JOE.2022.3226214.

[20] C. Liang, J. Xiong, Z. Zheng, *et al.*, "A visual detection method for night-time litchi fruits and fruiting stems," *Comput. Electron. Agric.*, vol. 169, p. 105 192, Feb. 2020, ISSN: 0168-1699. DOI: 10.1016/j.compag.2019.105192.

[21] B. Zheng, G. Sun, Z. Meng, and R. Nan, "Vegetable Size Measurement Based on Stereo Camera and Keypoints Detection," *Sensors*, vol. 22, no. 4, p. 1617, Feb. 2022, ISSN: 1424-8220. DOI: 10.3390/s22041617. [Online]. Available: https://www.mdpi.com/1424-8220/22/4/1617.

[22] E. Prasetyo, N. Suciati, and C. Fatichah, "Yolov4-tiny with wing convolution layer for detecting fish body part," *Comput. Electron. Agric.*, vol. 198, p. 107 023, Jul. 2022, ISSN: 0168-1699. DOI: 10.1016/j.compag.2022.107023.

[23] *Springfield is the codename for the GPU cluster owned and operated by the UiT Machine Learning Group*, [Online; accessed 6. Jun. 2022], Aug. 2022. [Online]. Available: https://uitml.github.io/springfield.

[24] *Albumentations Documentation - Full API Reference*, [Online; accessed 27. Mar. 2023], Mar. 2023. [Online]. Available: https://albumentations.ai/docs/api_reference/full_reference.

[25] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Commun. ACM*, vol. 60, no. 6, pp. 84–90, May 2017, ISSN: 0001-0782. DOI: 10.1145/3065386. [Online]. Available: https://dl.acm.org/doi/10.1145/3065386.

[26] J. Docekal, J. Rozlivek, J. Matas, and M. Hoffmann, "Human keypoint detection for close proximity human-robot interaction," *arXiv*, Jul. 2022. DOI: 10.48550/arXiv.2207.07742. eprint: 2207.07742. [Online]. Available: https://arxiv.org/abs/2207.07742.