MASTER'S THESIS 2022

Surgical Instrument Detection using Deep Learning

Tobias Carlsson, André Svensson

ISSN 1650-2884
LU-CS-EX: 2022-41

DEPARTMENT OF COMPUTER SCIENCE
LTH | LUND UNIVERSITY

EXAMENSARBETE
Datavetenskap

LU-CS-EX: 2022-41

Surgical Instrument Detection using Deep Learning

Kirurgisk instrumentdetektion med djup maskininlärning

Tobias Carlsson, André Svensson

Surgical Instrument Detection using Deep Learning

Tobias Carlsson
tobiasjf.carlsson@gmail.com

André Svensson
andre.svensson314@gmail.com

June 27, 2022

Master's thesis work carried out at
the Department of Computer Science, Lund University.

Supervisors: Maj Stenmark, maj.stenmark@cs.lth.se
Phan Kiet Tran, phan_kiet.tran@med.lu.se

Examiner: Elin Anna Topp, elin_anna.topp@cs.lth.se

Abstract

The purpose of this study was to apply object detection within the field of open-heart surgery. One goal was to retrain state-of-the-art deep neural networks to detect surgical instruments, namely the Heart-Lung tube, forceps, diathermy, scalpel, needle driver, retractor, and saw. The networks used in the evaluation were YOLOv4, YOLOv5, Scaled-YOLOv4, retinaNet, Efficientdet, SSD, and Faster-RCNN. We also studied the possibility of counting instrument changes during a surgery, which could help the surgeons to evaluate their performance. We investigated how incorrect predictions from a network could be used to generate more qualitative data. The data used to train and evaluate the networks were manually retrieved from videos of real-life surgery and annotated during this thesis. The networks were compared with mean average precision as the main metric.

YOLOv5 had the highest mAP score when comparing all networks. Using YOLOv5, we were able to predict instrument switches with promising results. The data generated from incorrect predictions improved the results further. The final model was evaluated on unseen real-life surgical videos, and the mAP_{50} was 88.5 %, which is a promising result for future applications.

Keywords: Artificial intelligence, Computer vision, Object detection, Deep learning, Pediatric heart surgery

Acknowledgements

We wish to express our gratitude to our supervisor, Maj Stenmark, for providing us with guidance, valuable advice and expertise every time we needed it. Maj has provided us with feedback on short notice throughout this thesis and for that we are grateful. Thank you also to Phan Kiet Tran, our medical supervisor, for your enthusiasm, kindness, and expertise as a heart surgeon. We would also like to thank our examiner, Elin Anna Topp for the guidance while examining our thesis. Furthermore, thank you to Jonna Johansson for your support and your valuable ideas. Last but not least, we would like to thank all the people working in the department of pediatric cardiac surgery at Skåne University Hospital for all the help and support they have provided.

Contents

Chapter 1
Introduction

In 2021, 457 pediatric heart surgeries were performed in Sweden [1]. After every surgery, there is an opportunity for the surgeon to improve their skills by receiving feedback. The most important metric is the recovery time of the patient and the quality of life the patient will have after the surgery. However, these metrics will only be available long after the surgery is performed and can therefore not be used as immediate feedback. Several studies have shown that a faster surgery generally leads to a faster recovery and higher quality of life after surgery [[2],[3],[4]]. An additional way that surgeons can evaluate their work is to record and review their own surgeries in order to identify ways to improve. However, the duration of a video is often more than three hours, meaning there may not be enough time for the surgeons to review all the footage. Creating a tool that can be applied on the videos to generate data from the surgeries could potentially be a valuable tool for feedback. In this thesis, we compared seven different object detection networks and identified the best-performing network. Using the highest performing network, we investigated the possibility of creating a tool that can count instrument changes performed by the surgeon. According to Ganni et al.[5], experienced surgeons have smaller average movements and fewer sudden movements if compared to the more novice surgeons. Instrument changes could potentially be a factor and that is why it was explored in this thesis. The networks were trained on images from surgery videos to detect the surgical instruments that were used during surgery.

Object detection can be used for many other applications, for example in robotics. A long term goal could be to train robots to sort instruments or assist during surgery.

1.1 Goals

Our main goal was to evaluate seven state-of-the-art networks retrained on surgical data. The thesis investigated the following three research questions:

- Which object detector network performs best according to the metric mean average precision (mAP)?
- How does the best network handle different instruments?
- How does the best model react to different amounts of training data?

Additionally, we evaluated if the best model can be used to detect instrument changes performed in surgery.

- How well can the model detect instrument changes?

Finally, we evaluated how extending the dataset with new images retrieved from analyzing incorrect detections affects the model.

- How does a model trained on this extended dataset compare to the best model?

1.2 Method

To evaluate the seven object detectors we extracted images from existing open-heart surgery videos. These images were then annotated with the software **labelImg** [6]. Training and validation were performed using these images and annotations. The best performing model was then used to identify surgical instruments in surgery videos.

In order to evaluate how the best model performed on different amounts of training data, the dataset was randomly divided into smaller training datasets. A test set was used to measure the performance for all training datasets.

To evaluate how well the best model could be used as a component in identifying instrument changes, a definition of an instrument change was created. This definition was applied within a python script that takes the output from the model and calculate changes. The evaluation was performed by comparing the output from the script to the ground truth which was created by manually going through a video and annotating the instrument changes.

In order to evaluate how the performance of the best model was affected by adding new images found by analyzing incorrect detections of the best model, the predictions were passed through a script that applied different rules for classifying frames as difficult or easy. A difficult frame would be manually annotated and then added to the dataset. This extended dataset was used to train the model again.

1.3 Contribution

For this master's thesis, Tobias Carlsson and André Svensson worked together and contributed to each part of the final results.

Our work contributed to the field of pediatric heart surgery by showing that an evaluation tool could be developed using an object detector. This benefits medical professionals, specifically pediatric heart surgeons.

1.4 Structure of this thesis

The **Introduction** includes motivations and research goals. The **Background** provides the theoretical background that the work is based on. The chapter contains, amongst others, an introduction to artificial intelligence, convolutional neural networks, object detection and the object detectors used in this thesis. The **Approach** includes the method, experiments and computer setup. **Evaluation** presents all results related to the research questions. **Discussion and Future work** discuss our approach such as network choices. This chapter also contains thoughts on future work which could improve the network performance. **Conclusion** answers the research questions presented in the introduction.

Chapter 2
Background

In this chapter, we introduce the concepts needed to understand the work done in this thesis. The main focus will be on object detection, with an introduction to artificial intelligence, deep learning and convolutional neural networks. We will also present evaluation metrics, which object detection models we have evaluated and related work.

2.1 Artificial Intelligence

Artificial Intelligence, (AI), is a field in computer science focused on making machines perform cognitive tasks, where many approaches have been created over the years. The knowledge-based approach to AI uses formal rules to determine what the system should do in a given situation [7]. These rules are built by humans to infuse the system with knowledge for the environment in which it will operate. Difficulties arise when these rules are hard to define, which is generally the case with tasks that humans perform intuitively, e.g., image recognition. It is not easy to create rules that, for example, detect cars in an image because lighting conditions, occlusion, angle, and multiple other factors will affect the appearance of a car, making it non-trivial to design these rules.

Machine learning is a subfield of AI, in which data about a given task is collected, and then this data is used to train a system, thereby giving it statistical knowledge about the task [7]. To illustrate the approach, assume that we want to design an AI system that can determine if a person has a certain infection. We will need data from multiple persons, both infected and not infected. Each patient will be represented as one data point. A data point consists of multiple features, where a feature is some measurable quantity, some medical test for instance. For every person, we also need to know if they actually have the infection. This type of machine learning, where each data point has a corresponding label, i.e., the actual answer, is known as supervised learning. To train the AI system, mathematical optimization is used [7]. In this example, the task is classification, i.e., classify each person into the correct class,

infected or not infected, see figure 2.1. The choice of features is up to the system designer and will require knowledge of the application domain.

If we instead want an AI that can detect cars in an image, we need to somehow come up with appropriate features that are useful for identifying cars. It is not obvious how to design these features, as mentioned above in the section on knowledge-based systems and formal rules. This issue can be avoided by letting the AI itself find good features. We use the whole image as input and let the image be processed in multiple steps, so-called layers, where each layer performs some computations and sends its results to the next layer. Machine learning algorithms that use multiple layers to extract higher level features are known as deep learning [7].

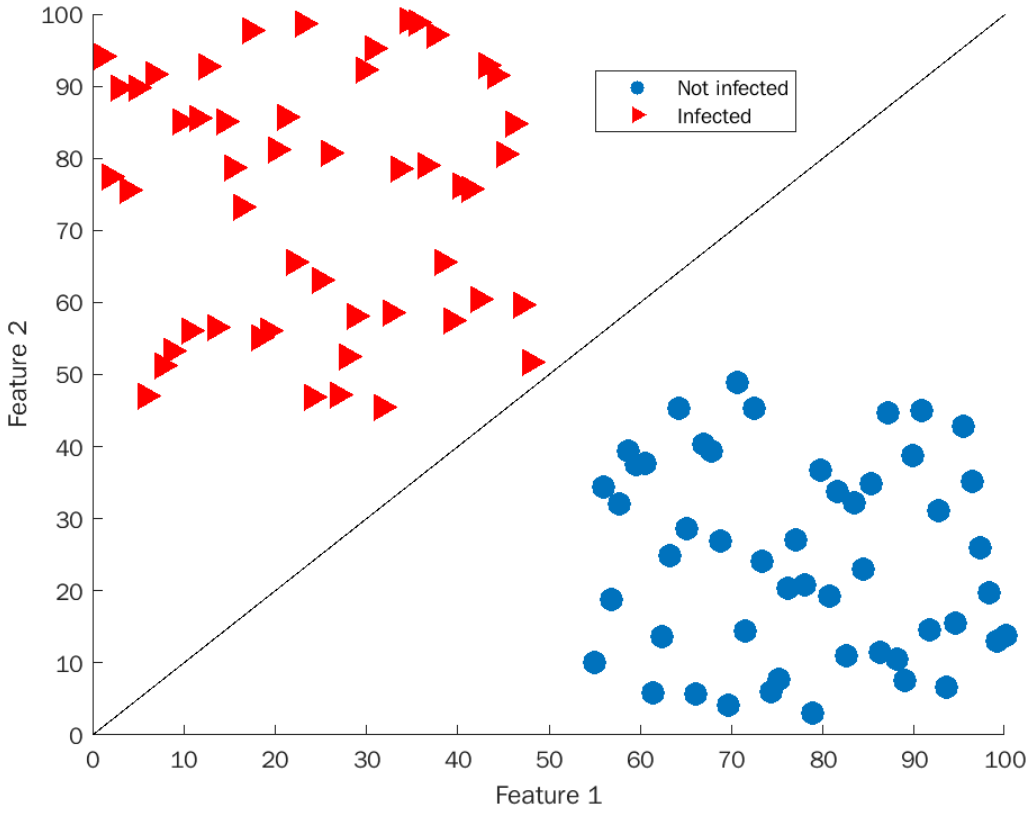


Figure 2.1: Demonstrates a simple classification example. Each blue dot represents a healthy individual and each red triangle represents an infected individual. Only two features are used for each data point. The AI system will use this data in its training, which has the goal of finding a decision boundary that divides the data points into two groups. The black line is the decision boundary which essentially is the AI model. This system will classify all data points below the line as healthy and data points above it will be classified as infected. In this example, the training data could be perfectly separated with a straight line. That does not mean that any new data point from a healthy person cannot end up above the decision boundary, or an infected person ends up below it, just that this particular training data has that property.

2.1.1 Deep learning

Deep learning, see above, is a subfield of machine learning. A deep learning model processes data in multiple layers chained together, where a layer can be seen as a function $\mathbf{f}(\mathbf{x}, \mathbf{w})$, $\mathbf{x}$ is the input and $\mathbf{w}$ are the trainable weight parameters for that layer [7]. A deep learning model with three layers expressed as a function will be written as

$$\mathbf{f}_3\Big(\mathbf{f}_2\big(\mathbf{f}_1(\mathbf{x}, \mathbf{w}_1), \mathbf{w}_2\big), \mathbf{w}_3\Big) \tag{2.1}$$

where $\mathbf{f}_1$ is the input layer, $\mathbf{f}_3$ the output layer and $\mathbf{f}_2$ is a so called hidden layer. The output of one layer is the input to the next. The trainable weight parameters are the values of a deep learning model that gets tuned during training. Hyper-parameters are another set of parameters that influence both the deep learning model itself and also how training is performed. Many deep learning models are named **neural networks** because their original design was inspired by the human brain [7].

Training a deep learning model

If we want a deep learning model that can detect the presence of cars in an image, we do the following: collect many images, ones that contain cars and ones that do not. Each image will be given a corresponding label, indicating if a car is present. The value 0 is used when no cars are present and 1 is used otherwise. These labels are usually created by a human that looks at each image and determines if any car is visible or not. These images will then be divided into three sets named training, validation and test set. The training set is used to perform the actual training of the model, while the validation set is used to evaluate the model during training. When training is complete, the test set is used to get a final evaluation of the model [7].

To train the model, the weight parameters must be initialised. There are multiple ways to initialize the weights, and the use of transfer learning is one of them, see section 2.7. When the weights are initialized we have an actual deep learning model that can take input and generate a prediction, car or no car. The output will be a number between $0 - 1$, where a value close to zero means the model believes there is no car present. The model has not yet been trained so these predictions will not be accurate. So to perform the actual training, the model repeatedly performs predictions on images from the training set. Each prediction is compared to the corresponding label. The error is computed with a loss function $L(\hat{y}, y)$ where $\hat{y}$ is the prediction and y is the label. To reduce this error the weight parameters need to be updated. The method of choice to update the weights is called gradient descent, which computes the gradient of the loss function with respect to every weight $\mathbf{w}_i$ and then updates the weight values along the negative gradient direction [7]. A weight update does not occur after every prediction but after a number of predictions. This number is called the batch size and is a hyper-parameter.

2.2 Convolutional neural network

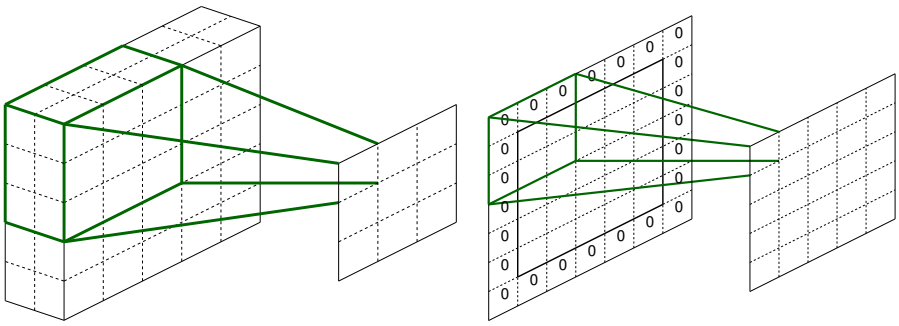
A convolutional neural network (CNN) is a type of neural network well suited for image processing. It contains multiple layers where each layer performs some operation and sends its

output to the next layer. In its simplest form a CNN consists of three kinds of layers, **filters**, **non-linearity (activation)** and **pooling** [8]. The output from a layer is called a feature map. [7].

A CNN takes an image as input. A color image is of size $width \times height \times 3$, where the last dimension represents the color channels red, green, and blue. The feature maps outputted by the layers inside the CNN will be of different sizes depending on where in the network they reside. Layers deeper in the network generate feature maps with decreased width and height while their depth usually increases. Figure 2.4 shows a feature map passing through a filter layer, activation layer and a pooling layer.

Filters

A filter is a small matrix containing trainable weights [8]. A common size is $3 \times 3 \times d$, where d is the depth of the filter, which must match the image (or feature map) the filter is applied to. The filter depth is usually not written as it follows from the image it is operating on. A filter is applied across the entire image, multiplying image values that currently overlap with the filter and summing them into one value. Figure 2.2a shows a 3×3 filter applied to a $5 \times 5 \times 2$ feature map, generating a $3 \times 3 \times 1$ feature map. A filter of the size 3×3 reduces the size of the feature map by two pixels both in width and height. To preserve size, zero padding can be used, as shown in Figure 2.2b. Here the depth of the input feature map is 1, but it works equally well with larger depth. Zero-padding adds zero-valued pixels around the feature map, so when a 3×3 filter is applied, the generated feature map is of the same size as the input feature map. Notice that larger filters would require more zero-padding to preserve feature map size.



(a) **(b)**

Figure 2.2: (a) Shows a 3×3 filter applied to a $5 \times 5 \times 2$ feature map. (b) 3×3 filter applied to a $5 \times 5 \times 1$ feature map with zero-padding

The step size with which a filter is moved across a feature map is called the stride. In Figure 2.2, both filters use a stride of 1. A stride of 2 would in Figure 2.2 generate feature maps of size 2×2 and 3×3 respectively.

A filter-layer in a CNN contains multiple filters, where each filter is responsible for detecting different features. Each filter must output the same sized feature map. If a layer has n filters, their combined output will be a feature map of size $w \times h \times n$, where w and h depend on the input feature map, if zero-padding is used and the chosen stride.

Non-linearity

Following the filters is a non-linearity function (activation function). The Rectified Linear Unit (ReLU) is one example of such a function, $ReLU(x) = max(0, x)$. This function is applied to each of the elements in the feature map, where the variable x is an element from the feature map. The function does not transform the size of the feature map, only the values within it [8].

Pooling

Pooling is used to reduce the size of a feature map [8]. The reduction performed by the pooling layer decreases the computational power needed because the remaining layers then have fewer elements they need to operate on. Figure 2.3 shows the use of max pooling, where a window slides over the input feature map and selects the largest element. The sliding window has a depth of one, meaning the feature map depth is preserved, only width and height are reduced.

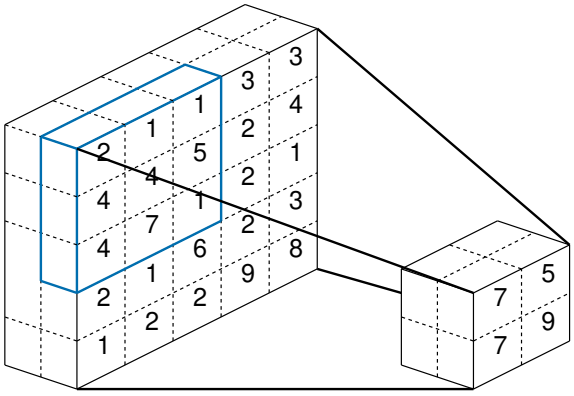


Figure 2.3: Max pooling with a 3×3 window, using stride 2. In this image, only values on depth one are displayed in the feature maps.

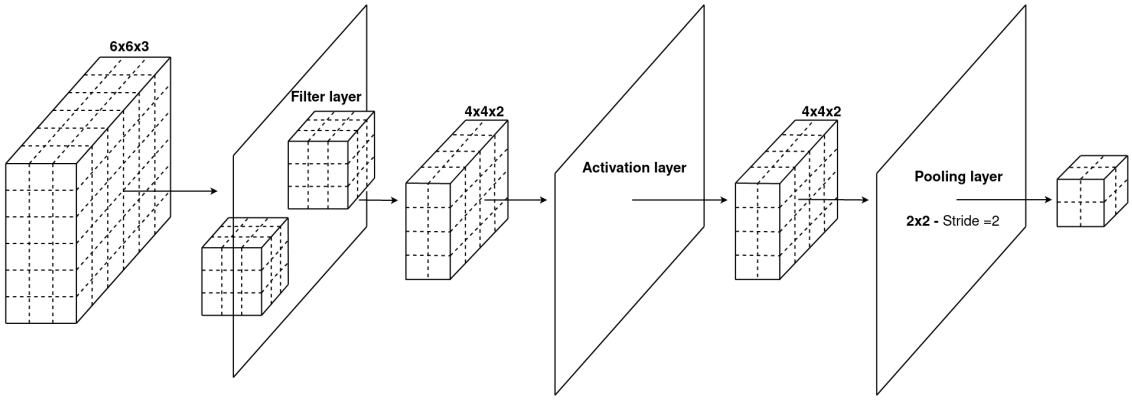


Figure 2.4: Displays a small part of a CNN. To the left, a feature map is entering a filter layer consisting of two 3×3 filters, stride 1, no padding used. Next, an activation layer performs element-wise operations. Lastly, a pooling layer with a window size of 2×2, stride 2. In this case, feature map depth is not increased.

2.3 Object detection

Object detection refers to the task of both detecting and localizing objects in an image, where localization refers to the task of identifying an object's position in an image [9]. In contrast to image classification, where an image only contains objects from one class, object detection deals with images where multiple objects from multiple classes can occur in the same image.

An object detector takes an image as input and outputs a set of bounding boxes. For each bounding box, there is an associated class. A bounding box defines a rectangular region in the image that encapsulates an object. It is defined by four values, commonly they are the box center: x- and y coordinates, and the width and height [10]. Figure 2.5 shows an example of predicted bounding boxes and ground truth bounding boxes, where ground truth bounding boxes are created by a human and used as training data.

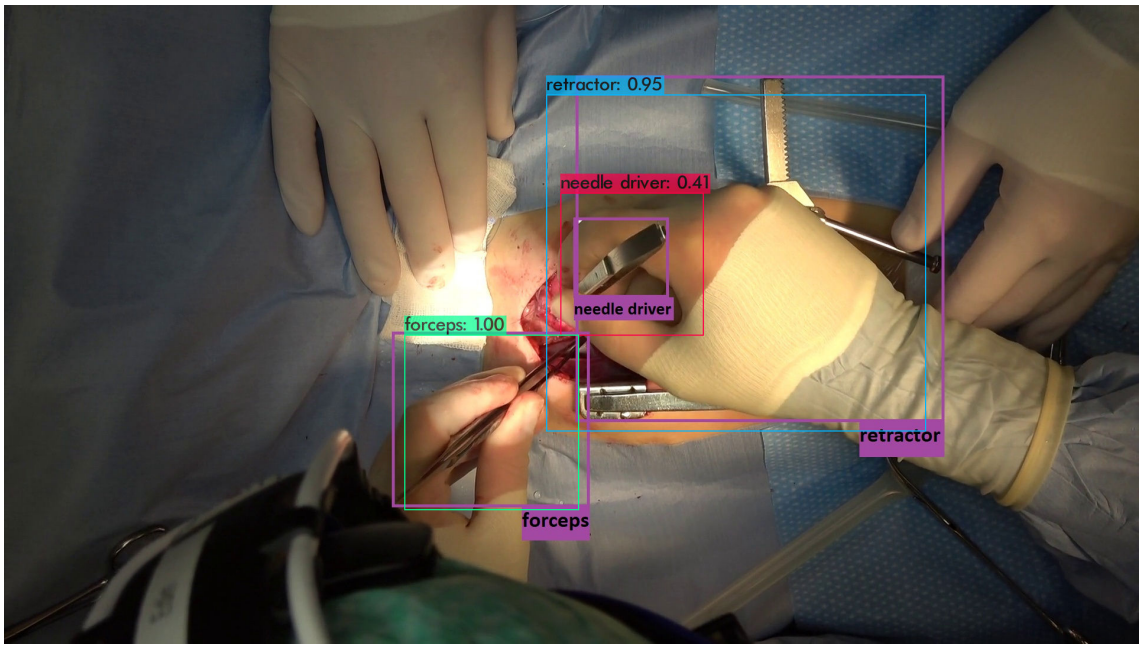


Figure 2.5: Drawn on this image are ground truth bounding boxes in purple and bounding box predictions made from an object detector. The prediction from the object detector also contains class assignments for each bounding box and its confidence in each bounding box.

2.3.1 Model architecture for object detection

Object detection is a difficult task and there exist a plethora of models that have been developed over the years [9]. But in 2014, with the use of deep learning and convolutional neural network, major advancements were made and the best-performing models are now based on deep learning [9]. These deep learning models are commonly divided into two categories, two-stage detectors and one-stage detectors. This classification distinguishes how different models process an image to generate a prediction. A two-stage detector consists of a region proposal step and a classification step. The region proposal tries to identify regions in the image that potentially could contain an object. Then in the classification step, these proposed regions are analyzed and the model tries to determine if they contain an object. A one-stage

detector does not generate any regions, it instead tries to locate and classify objects directly from the image [9].

We will present a generic architecture for an object detector based on the models presented in section 2.8. All models evaluated in this thesis follow this architecture to some extent, although there exist other object detectors based on deep learning that take other approaches [11]. Using the same naming convention as Bochkovskiy et al. [12], an object detector consists of three parts: a **backbone**, consisting of a CNN that extracts image features, a **neck**, that combines various feature maps from the backbone to further refine the features, and finally the **head** that performs predictions on these refined feature maps and outputs bounding boxes with class assignments and confidence scores. Figure 2.6 displays a schematic for the object detector design that is presented here.

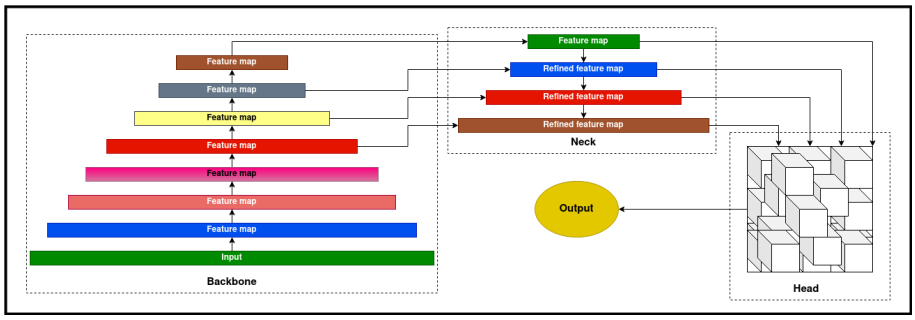


Figure 2.6: Schematic for an object detector. The backbone can be any arbitrary CNN. Four feature maps from the backbone are used as input to the neck, where they are combined and sent to the head for detection. In the head, multiple filters are applied to each feature map to generate all output values.

Backbone

The backbone can be any CNN. As mentioned in section 2.2, a CNN will apply filters to the input image, generating so-called feature maps. More filters will again be applied to these feature maps, generating new feature maps. This process is repeated multiple times, along with other operations, depending on the specific CNN. This will generate a hierarchy of feature maps, from large-sized ones in the early stages to smaller ones (but with greater depth) in the later stages. The smaller ones will be more heavily processed, meaning they contain more useful information for the network, but because their resolution has been reduced, they cannot represent fine-grain details. The larger feature maps retain more details, but because they have not been processed as much, they contain less useful information from the network's perspective. To remedy this, feature maps of different sizes are merged, this happens in the next part of the object detector, the neck.

Neck

A subset of the feature maps generated by the backbone is sent as input to the neck, the subset is determined by the model designer. There exist multiple different strategies to merge feature maps, but many are inspired by Lin et al. [13]. In this paper, they use four feature maps from the backbone. Before any feature maps can be combined they must be transformed to

the same size, and initially, they vary in all three dimensions, width, height and depth. Lin et al. [13] begin by converting all feature maps to be of depth **256**. This can be achieved by applying **256** filters of size $1 \times 1 \times d$ to each of the feature maps, where d denotes the depth of the feature map the filters are applied to. A 1×1 filter applied to a feature map of size $w \times h \times d$ will output a feature map of size $w \times h \times 1$, **256** filters will then together create a feature map of depth **256**. Figure 2.6 demonstrates how the feature maps are merged, the smallest feature map is upsampled to match the width and height of the previous feature map and then their elements are added. This is repeated until all feature maps have been combined. The final feature map and all intermediate ones are then sent to the head for detection.

Head

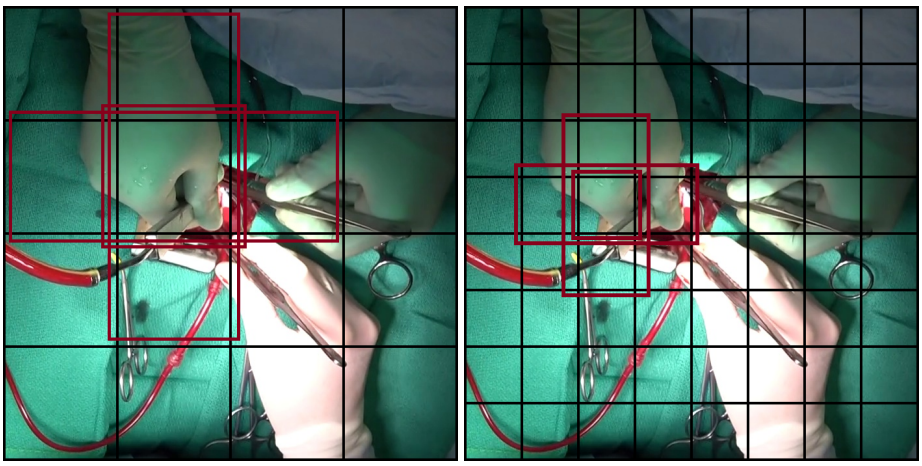
The head receives the refined feature maps and might apply some additional filters on them. Then the final step is to generate predictions. To generate a bounding box prediction, four values are needed because a bounding box is represented with four values, see the beginning of section 2.3. The class assignment for a bounding box is created by an object detector by generating c class probabilities, one for each class the object detector is designed to recognise. The class with the largest probability-value then gets assigned to the bounding box and the probability is used as confidence score.

All models in this thesis make use of anchor boxes in their predictions. An anchor box is a predefined bounding box with a certain area and aspect ratio. These anchor boxes are used as a starting point for the object detector when it generates bounding boxes. It modifies these anchor boxes to generate bounding box predictions. For each anchor box, four offset values are calculated, one for each value that defines a bounding box. The final outputs from the model are these offset values to the anchor boxes, as well as class probabilities for every anchor box. Anchor boxes are directly linked to feature maps. Each pixel in every feature map will be the center for k anchor boxes, where k is chosen by the model designer. Figure 2.7 shows an example where k=3. Pixels in the same feature map get an identical set of anchor boxes. Anchor boxes in low-resolution feature maps usually have larger areas because these feature maps mainly capture information about large objects. Likewise, high-resolution feature maps contain anchor boxes with smaller areas because the higher resolution enables smaller objects to be detected. Using this anchor box setup means they only need small tweaks to make them enclose a nearby object, if any is present.

The total number of anchor boxes across all feature maps becomes $\sum_s s^2 k$, $s \in S$, where S is the set of side lengths of the feature maps (assuming square feature maps). Because pixels in the same feature map have identical anchor boxes, the number of unique anchor boxes– meaning they have the same area and aspect ratio– becomes $N \cdot k$, where N is the number of feature maps.

To generate the output values, multiple filters are applied to the feature maps. Each anchor box needs four offset values, so four filters per anchor box are needed. There are $N \cdot k$ unique anchor boxes, which means there needs to be $4 \cdot N \cdot k$ filters to generate all offset values. This means that anchor boxes with the same area and aspect ratio share the same filters. To generate class predictions, every unique anchor box gets a filter for every class, leading to $N \cdot k \cdot c$ filters, where c is the number of classes. In total $Nk(4 + c)$ filters are used to generate

all output values. Usually, the filters are of size 3×3.



(a) 4×4 feature map **(b)** 8×8 feature map

Figure 2.7: Two differently sized feature maps are overlaid on the input image to demonstrate how the anchor boxes (in red) will appear on the image. Here, three anchor boxes per pixel in the feature maps are used, where only the anchor boxes for one pixel are drawn. Both feature maps use anchor boxes of the same aspect ratios, {1:1, 3:1, 1:3}, but the area for each corresponding anchor box varies between the feature maps. In total there are $3 \cdot 16 + 3 \cdot 64 = 240$ anchor boxes in these two feature maps, and $2 \cdot 3 = 6$ unique anchor boxes.

2.3.2 Training an object detector

To train an object detector, an image dataset with bounding box annotations is needed. During training, ground truth bounding boxes must be assigned to anchor boxes. Each bounding box can be assigned to the anchor box with the greatest overlap, but usually, a bounding box will be assigned to multiple anchor boxes because there will be significant overlap with many of them. There will be a huge amount of anchor boxes not assigned to any bounding box because there are many more anchor boxes than bounding boxes in any given image. During training, an image will be sent to the model as input, it is processed by the network which outputs offset values and class predictions. The loss function will be a sum of the errors between the predicted bounding box coordinates, i.e, anchor boxes with their offsets applied, and the ground truth bounding box coordinates, together with the errors for class probabilities. A distinction in the loss function is made between anchor boxes that were assigned to a bounding box and those that were not. Anchor boxes that have been assigned a ground truth bounding box contribute to the loss function fully, meaning their errors in offsets and class probabilities will be computed. Unassigned anchors will only contribute to their class probabilities because their offset values have no real meaning, there is no object to detect so there are not any values to compare them against.

2.4 Challenges with object detection

Objects of the same class can look very different depending on their angle, scale, lightning conditions and amount of occlusion from other objects. An image classifier only needs to

detect the presence of an object, while an object detector needs to learn how to find its exact position, which is rather difficult when all these factors come into play [9]. Another challenge, arising mainly for models using anchor boxes, is the abundance of anchor boxes with no ground truth assignment. The model presented by Lin et al. [14] contains over a hundred thousand anchor boxes. The errors these anchor boxes generate in the loss function will overshadow the ones that have ground-truth assignments, leading to poor training results as the model cannot focus on detecting actual objects. These challenges must be addressed by a model designer to create a well performing model.

2.5 Evaluation metrics

Metrics are used to measure the performance of a neural network and allow for comparison between different models. Padilla et al. [15] state that a building block for metrics used within object detection is the intersection over union (IOU).

2.5.1 Intersection Over Union (IOU)

IOU calculates the overlap between two bounding boxes, namely a predicted bounding box and a ground truth bounding box. The value is computed by dividing the intersection area with the union area, as can be seen in the figure 2.8. IOU goes from 0, when there is no overlap, to 1 when there is perfect overlap [15].

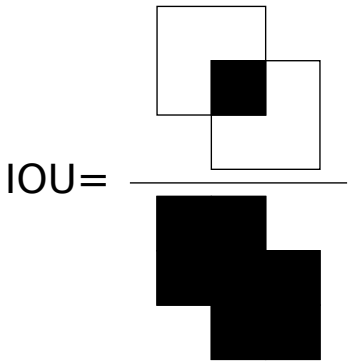


Figure 2.8: The numerator represents the intersection between two bounding boxes and the denominator represents the union of the same two bounding boxes

2.5.2 True positive, False positive, False negative

True positives (TP), false positives (FP) and false negatives (FN) are computed using the IOU value. Only bounding boxes representing the same class are compared. A predicted bounding box is classified as a true positive when its IOU with a ground truth bounding box is greater than a given threshold value. Choosing a threshold value close to 1 means that the two bounding boxes must be very similar to be given a true positive classification. False positive is assigned to a predicted bounding box that does not have an IOU greater than the threshold value with any ground truth bounding box. Objects that are not detected, meaning

their ground truth bounding boxes have no IOU value greater than the threshold value with any predicted bounding box are represented as a false negative [15].

2.5.3 Precision

Precision is a ratio between the number of true positive predictions over the total number of predictions [15]. It is calculated per class. A precision value equal to 1 means the object detector was always correct when predicting for the given class. It does not mean that all instances of the given class was detected since there could be false negatives.

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

2.5.4 Recall

Recall is a ratio between the number of true positive predictions over the total number of ground truth bounding boxes [15]. The sum $TP + FN$ equals the number of ground truth bounding boxes and is therefore independent of how the model has predicted. A recall value equal to 1 means the object detector managed to correctly predict all instances of a given class. It might however also have predicted extra bounding boxes of this class, which does not affect the recall value.

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

2.5.5 F1 Score

The F1 score is the harmonic mean of precision and recall. This can be used advantageously in some situations, specifically when both recall and precision are equally important for the application. F1 can be easier to work with since it only requires the user to observe one metric instead of multiple. The formula for the F1 score is shown below [15].

$$F1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}$$

2.5.6 Mean average precision (mAP)

Precision-Recall curve

For each class, the precision-recall curve shows the relationship between recall and precision. The black solid curve in figure 2.9 displays the precision-recall curve for a class named *scalpel*. This curve is generated based on an object detector's predictions on thousands of images, using a given IOU threshold to assign true positives, false positives and false negatives. To generate the entire precision-recall curve, precision and recall are calculated for different confidence score thresholds, where each bounding box prediction has its own confidence score, see figure 2.5. For example, a confidence threshold of 0.35 means that all predictions with confidence less than 0.35 will be ignored when precision and recall are calculated. Precision and recall values are calculated for all confidence threshold values from 0 up to 1 [15].

A confidence threshold close to one will only include the best predictions, leading to high

precision but low recall. The reason for a low recall is that many correct predictions will be excluded because their confidence scores are to low. Increasing the confidence threshold reduces precision as more predictions are included in the calculation, but recall is then improved. The erratic appearance of the black curve in 2.9 indicates that low confidence predictions can be true positives while higher confidence predictions can be false positives.

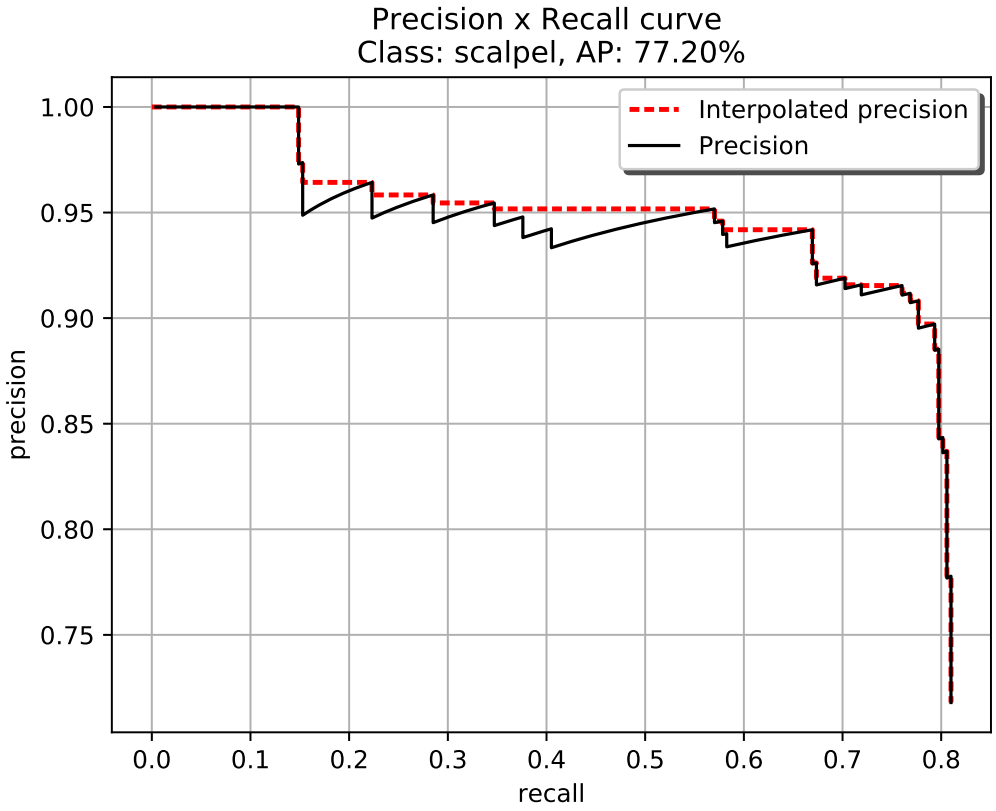


Figure 2.9: The black solid curve is the precision-recall curve of a class named *scalpel*. The red dotted curve is the interpolated precision-recall curve.

Average precision (AP)

Average precision (AP) is a metric that combines precision and recall and is calculated as the area under the interpolated precision-recall curve, shown in figure 2.9. Let $P(r)$ represent the precision-recall curve, where r represents recall and $P(r)$ represents the precision value for the given recall value r. The interpolated curve $P_i(r)$ is given by equation 2.2. The precision at recall r is calculated by taking the maximum precision observed for all recall values above the current recall value. The area under $P_i(r)$ can be calculated in different ways. Equation 2.3 shows the N-point interpolation method which is used by PASCAL VOC [16] and COCO [17] with $N = 11$ and $N = 101$ respectively. Exact calculations of the area are another method.

$$P_i(r) = \max_{x, x \geq r} P(x) \tag{2.2}$$

$$AP = \frac{1}{N} \sum_{r} P_i(r), \quad r = \frac{N-n}{N-1}, n = 1, .., N \tag{2.3}$$

Mean average precision (mAP)

Average precision is calculated per class. To combine the AP scores for every class, mean average precision (mAP) is used which calculates the average over all classes. Equation 2.4 shows the formula, where AP_c represents the average precision for a class and M is the number of classes.

$$mAP = \frac{1}{M} \sum_{c=1}^{M} AP_c \tag{2.4}$$

mAP_{50} and $mAP_{50:95}$

Mean average precision is highly dependent on the IOU threshold, therefore the choice of IOU threshold is normally included when working with mAP. The notation mAP_{50} indicates that an IOU threshold of 0.5 was used. There also exists a metric that computes the average of mAP with different choices of IOU thresholds. $mAP_{50:95}$ represents the average of $mAP_{50}, mAP_{55}, ..., mAP_{95}$ [15].

2.6 Preprocessing and data augmentation

A crucial part of any deep learning project is to have enough data [18]. This data should conform to a certain format, which is handled by a preprocessing stage. For image data, common preprocessing steps are to rescale all pixel values to lie between $[-1, 1]$ or $[0, 1]$ and to resize all images to the same size. [7].

Data augmentation refers to methods that artificially increase the amount of data by manipulating the data that already exists. Data augmentation can be implemented in many different ways. One of the simplest approaches is to create several copies of each training image and apply zooming effects, rotations, or other basic enhancements [18]. Mosaic is another technique, where four images are tiled together [12]. Using these methods to increase the data quantity can improve deep learning models [18].

2.7 Transfer learning

Training a model on data from a given task and then using that model to initiate the training of another task is known as transfer learning [19]. The first training round can simply be used as a weight initialization for the second training. It can also be the main part of the training and the second stage only performs fine-tuning. This is useful when data for the actual task is scarce, but there exists another similar task with much more data [20]. Publicly available object detection models are usually pre-trained on large public datasets such as COCO and PASCAL VOC, see section 2.7.1. It is then up to the user of these models to decide how they

want to apply the transfer learning, either as weight initialization or fine-tune the already trained model.

2.7.1 PASCAL VOC and COCO

Pascal VOC and COCO are two publicly available object detection datasets that can be used to compare different object detection models. They both use their own implementation of the mAP metric, where PASCAL VOC has implemented mAP_{50} with $N = 11$ point interpolation [16] and COCO instead uses $N = 101$ point interpolation [17]. Later versions of PASCAL VOC have switched over from N point interpolation to calculate the exact area under the interpolated precision-recall curve [21].

2.8 Models

When designing an object detection model, there is a multitude of design choices to make and each model brings its own novel ideas in an effort to increase model performance. We will only give a short introduction to them here, more details can be found in their respective publications.

2.8.1 YOLOv4

YOLOv4 (you only look once) is a one-stage detector developed in 2020. It is preceded by YOLOv1-v3. The YOLO family has always focused on creating models capable of performing predictions in real-time, and YOLOv4 is no exception. It introduces some new augmentation methods not present in previous yolo models, among which mosaic is one. Its architecture follows a similar design as described in section 2.3.1.

2.8.2 YOLOv4-scaled

YOLOv4-scaled is a family of models all deriving from yolov4. They were developed in 2020 and implement a scaling scheme, like EfficientDet, but with the addition of also making tweaks to the architecture. YOLOv4-scaled both consist of larger and smaller models than the original YOLOv4.

2.8.3 YOLOv5

YOLOv5 was developed in 2020 shortly after YOLOv4. No publication exists for this model, but the implementation is publicly available on github [22].

2.8.4 retinaNet

retinaNet is a one-stage detector developed in 2017. Its architecture is similar to the one presented in section 2.3.1. retinaNet tries to solve the issue concerning the overwhelming majority of empty anchor boxes affecting the model performance by introducing a loss function called retina loss. This loss function reduces the loss for bounding boxes that are well classified. An accurate prediction leads to low loss, and with retina loss, it is reduced even

further. This should cause the weight updates to be more affected by the inputs that are not classified well instead of being overwhelmed by many small errors from good predictions [14].

2.8.5 Single Shot Multibox Detector - SSD

SSD is a one stage detector developed in 2015. Its architecture has similar elements to the one described in section 2.3.1. It consists of a backbone followed by a few extra layers to create more feature maps. No neck is used in SSD, instead, the filters that usually reside in the head are applied directly to these extra feature maps together with one feature map from the backbone. Anchor boxes are used, in total 8732 [23].

2.8.6 EfficientDet

EfficientDet is a family of one-stage detectors developed in 2019 [24]. The models are produced by employing a scaling scheme over a network architecture that is very similar to the one described in section 2.3.1. The scaling scheme makes it easy to increase the number of parameters to gain better results, at the cost of computational power. There exist 7 models in the family, named EfficientDet d0-d6, d0 is the smallest and d6 is the largest. The model size is increased by scaling the three parameters input image resolution, the number of layers, and the number of filters per layer. The neck in EfficientDet consists of blocks of weighted Bi-directional Feature Pyramid Networks [13].

2.8.7 Faster R-CNN

Faster R-CNN is a two-stage detector developed in 2015 [25]. It precedes the models R-CNN and Fast R-CNN. Because it is a two-stage detector, it differs from the architecture in section 2.3.1, although Faster R-CNN can be extended to become more similar to the approach followed by Lin et al. [13], where a neck is introduced. Faster R-CNN generates region proposals with a CNN they name Region Proposal Network (**RPN**). It takes an image as input and on its final feature map, anchor boxes are defined. Then, similar to how the head operates in section 2.3.1, multiple filters are applied to this feature map to calculate offsets to these anchor boxes. For each anchor box, an objectness score is calculated using a filter, representing the network's confidence for if there is an object there or only background. Faster R-CNN also has a backbone which takes the same input image as RPN. The offset anchor boxes RPN believes contained an object get projected onto the last feature map of the backbone. These regions will be used to generate class predictions and further refinement of the anchor box offsets to generate the final bounding boxes.

2.9 Related work

Deep learning is prevalent in the field of medical image analysis [26], and one of the tools available is CNN based object detection. Multiple open-source medical datasets exist to train these object detectors [27], but to the best of our knowledge, there are no public datasets with images from open-heart surgery and no previous work exploring the use of object detection in the setting of open-heart surgery. However, there exist numerous papers exploring the performance of object detectors like YOLO, SSD and R-CNN, on different medical datasets.

Zhao et al. [28] present an approach using an object detector designed to detect surgical instruments used in robot-assisted surgery. They compare its performance to other detectors, such as Faster R-CNN, YOLOv3 and retinaNet. Surgical instrument detection in robot-assisted surgery is also explored in the publishing from Bamba et al. [29], where the focus lies on detecting different kinds of forceps. Another object detector comparison is made in Tan et al. [30], where retinaNet, Yolov3 and SSD are used to perform pill identification.

Comparison of object detectors is performed on other datasets as well. Srivastava et al. [31] compare the performance of three models, YOLOv3, SSD and Faster R-CNN using the COCO dataset. In Sanchez et al. [32], a multitude of object detectors are compared using both COCO and PascalVOC. A comparison of detectors is made by Groener et al. [33], using a dataset consisting of overhead satellite imagery.

Chapter 3

Approach

This chapter contains the approach that was chosen for this thesis. The first section presents general information about the surgery videos, which instruments we focused on and how frames were retrieved from the videos. This is followed by a presentation of the datasets. Additionally, we have sections for the research questions presented in the introduction. The final sections will cover the computer setup.

Video documentation was used during this thesis and it is a standard procedure and did not require any change of the workflow during surgery.

3.1 Material and methods

3.1.1 Videos

We have used videos from 23 surgeries, where a video can span from two hours to more than five hours. Each video was recorded from a camera positioned above the patient. Figure 3.1 shows some images from various stages of the surgery. The lighting conditions and level of zoom vary slightly between different videos, which can be seen in figure 3.1. Because the camera was recording from above, the view could sometimes be obstructed by the heads of the surgeons, making it impossible to detect any objects even if they were present. The videos are in resolution 1920×1080 or 1280×720.

There are always two surgeons that perform a surgery, the lead surgeon and the assistant.

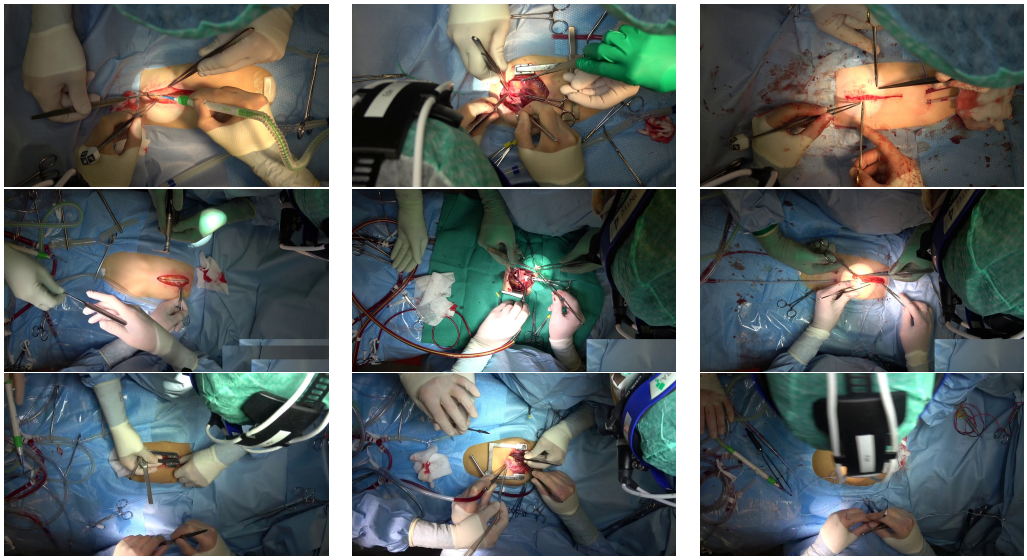
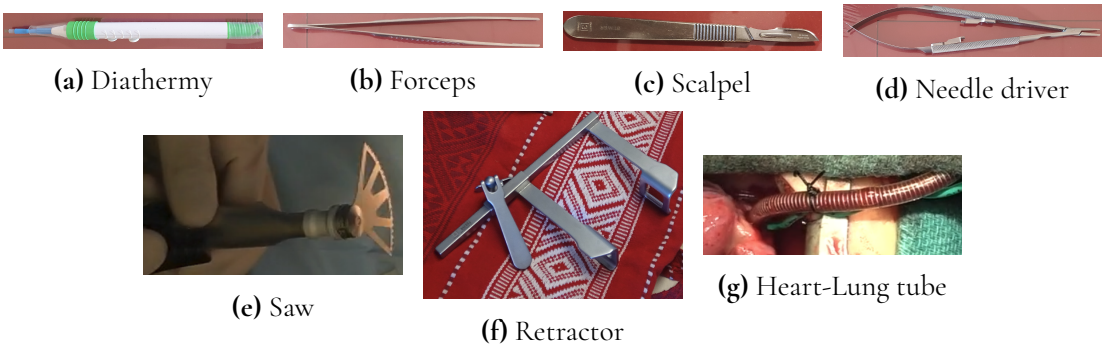


Figure 3.1: Each row contains images from the same video. The leftmost images are from the early stages of the surgery, when the sternum is opened. The center images depict surgical repair, occurring in the middle of the surgery, and the rightmost images depict the final stage where the wound is closed. The lower right corner in the images from the center row is masked out because there were instruments there during the entire duration of the surgery.

3.1.2 Instruments used in evaluation

Different kinds of instruments are used throughout heart surgery and we focused on seven of them in order to limit the scope of the thesis. We chose the instruments presented below since they are widely used throughout a surgery and are considered important. Some of the instruments can be used to represent events in a surgery, such as the retractor which marks the start and end phase of a surgery or the tubes from the heart-lung bypass machine (HL-tube). There are variations of the instruments presented below, so the instruments may not always look identical to figure 3.2. The instruments that we focused on are presented below and shown in figure 3.2.

- Diathermy - figure 3.2a
- Forceps - figure 3.2b
- Scalpel - figure 3.2c
- Needle driver - figure 3.2d
- Saw - figure 3.2e
- Retractor - figure 3.2f
- HL tube - figure 3.2g



(a) Diathermy **(b)** Forceps **(c)** Scalpel **(d)** Needle driver

(e) Saw **(f)** Retractor **(g)** Heart-Lung tube

Figure 3.2: Surgical instruments that we choose to focus on.

3.1.3 Frame retrieval and annotation

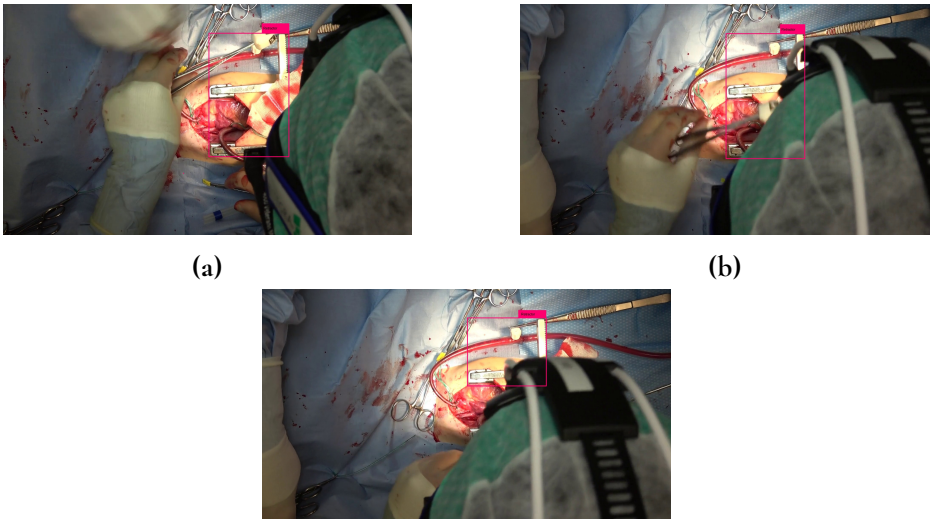
The data preparation step consists of two steps: frame retrieval and annotation.

Frame retrieval

Frame retrieval is a manual process where video-sequences that contained the relevant instruments were selected and frames were extracted. Based on the video sequence's properties, including its length and what instruments it contained, the number of extracted frames was adjusted to create a more balanced dataset. For instance, if the chosen sequence contained a scalpel, more frames were extracted compared to if it only consisted of forceps because scalpels are used much less frequently during surgery.

Annotation

In the annotation process, we created ground truth bounding boxes for each instrument in all images that were extracted during the frame retrieval. We used the python-software **labelImg** to create bounding boxes [6]. In figure 3.3, three similar images are displayed with annotations for the retractor. When annotating, we aimed to cover the whole instrument even though it was partially obscured, as can be seen in figure 3.3a and 3.3b. Figure 3.3c shows an exception to this rule, as one visible part of it is very small and therefore disregarded.



(a) (b)

(c)

Figure 3.3: Each image represents an annotation of a retractor.

3.2 Dataset

We used three different datasets throughout this thesis. The first dataset which is used in section 3.3, referred to as the original dataset, ODS, was created from 14 different videos. A second dataset referred to as the extended dataset, EDS, was created using data from ODS and new data generated using the approach described in section 3.7 with 5 additional videos. The third and smallest dataset was used as an independent test dataset and referred to as ITDS, created from three separate videos not used in the creation of either ODS or EDS. The data in ITDS was completely unseen from the perspective of the networks and was therefore used to test how well the networks handled new data. One additional video, not used in any of the datasets, was used together with one video from ITDS to count instrument changes, see section 3.6

- ODS was divided into training, validation and test split.
- EDS was divided into training and validation split.
- ITDS was used as a test set and therefore not divided into different splits.

The dataset details of the three datasets are presented below in table 3.1 and 3.2. The total number of instrument occurrences is shown below in table 3.3.

3.3 Network comparisons - mAP_{50}

The networks presented in section 2.8 were selected because of their relevance and documented performance. There are multiple implementations of the networks used in this thesis. Our choice of implementation was based on how recently they were released and how well documented they were, see appendix A.

Dataset split- Percentage		
Abbreviation	Dataset	(%)
ODS	Test	10 %
	Validation	10 %
	Train	80 %
EDS	Test	0 %
	Validation	5 %
	Train	95 %

Table 3.1: Dataset split - Percentage

Dataset split - Images - Videos			
Abbreviation	Dataset	Images	Videos
ODS	Test	810	14
	Validation	810	
	Train	6471	
EDS	Test	0	19
	Validation	816	
	Train	15474	
ITDS	Independent test dataset	1278	3

Table 3.2: Dataset split - Quantity

Instrument occurrences		
Instrument	ODS	EDS
Forceps	10181	23960
Retractor	4841	11309
Needle driver	2271	5545
HL-tube	2171	7026
Diathermy	2041	2585
Scalpel	1454	2392
Saw	390	434

Table 3.3: Instrument occurrences - Total number of occurrences per instrument in all images from ODS and EDS.

3.3.1 Model training

To create a fair comparison, the networks were trained with similar settings, using the following below.

- Default parameters.
- No preprocessing of data.
- Maximized image size.
- The same memory limit, the batch size was reduced if the training could not run with the original setting.

- The same hardware, see section 3.8

Default settings were chosen because they have been tuned by the creator of the network and are well adjusted to the network and further hyper-parameter optimization is challenging. Additional preprocessing was excluded since the networks handle input data in different ways, they already apply preprocessing stages. So if we were to apply preprocessing, the comparison would not have been fair since some would most probably handle it better than others. Maximizing the input image size was done to be able to capture as much detail as possible in the input images.

All networks were trained on the ODS train-split, validated on the ODS validation-split, and tested on the ODS test-split. The evaluation was also carried out on ITDS to show how well the models perform on unseen data.

3.3.2 Evaluation

We considered recall, precision, and mAP as evaluation metrics. Both the mAP metric and the F1 score aim to provide a metric that combines the precision and recall score, see section 2.5.6. The mAP metric is widely used in this field. We use both mAP_{50} and $mAP_{50:95}$. The results were obtained by generating predictions on the ODS test-split and the ITDS. This generated four scores, two mAP_{50} and two $mAP_{50:95}$. These scores were then averaged. The averaged score was used when comparing networks, since the score retrieved from the ODS test-split represents how well the networks were trained on the data provided in training and the score retrieved from ITDS represents how well the networks handled new data, both are of equal interest. The averaged mAP_{50} score was used to select the best network. The mAP metrics were calculated with the software *object detection metrics*, see Padilla et al. [15].

3.4 Instrument comparison

To find and evaluate detection efficiency for each instrument, both the ODS test-split and ITDS were used. The detection outputs were used to compute AP_{50} and $AP_{50:95}$ for each instrument.

3.5 Data scalability

To evaluate the scalability of the best-performing network, we created 6 different training sets. All training sets contained different amounts of images, the quantities are shown in table 3.4. In order to evaluate the scalability, we trained our best-performing network on these training sets separately and validated the performance, based on the mAP metric, using both the ODS test-split and ITDS.

Dataset split - number of images	
Dataset	Images
Train_200	200
Train_400	400
Train_800	800
Train_1600	1600
Train_3200	3200
Train_6400	6400

Table 3.4: Dataset split- Total number of images in the scalability train datasets.

3.6 Instrument changes

The networks were trained to detect surgical tools in the videos but not to detect instrument changes. To calculate instrument changes, we wrote a script that took the detection output from an entire video, generated by the best network, and analyzed when instrument changes occurred. Section 3.6.2 provides further specifics about the script. We chose to exclude forceps when detecting instrument changes, as we mainly focused on the instruments the lead surgeon uses in their main hand, and there were almost always at least two forceps present, sometimes up to five, often held in the assistant's hands. Two videos were used for the evaluation, referred to as *Video1* and *Video2*. Video1 and Video2 consisted of 70 changes each when using definition1 presented in section 3.6.1 and focusing on the instruments mentioned above. Video1 consisted of 152 changes while Video2 consisted of 162 changes when focusing on all instruments that occurred in the surgeries, without exclusion.

3.6.1 Defining an instrument change

We created two definitions of an instrument change, where *definition 1* is the more natural definition closely resembling the intuitive notion of changing something, whereas *definition 2* represents a switch which our network will be able to detect, because our network, for instance, cannot distinguish between an instrument that is being used and one that is placed on the table. *Definition 2* is based on how the composition of visible instruments changes between frames, which is represented by something called an event-log, see section 3.6.2.

Definition 1

An instrument change is a pair of instruments $(\textit{instrument}1, \textit{instrument}2)$, where the surgeon first lets go of $\textit{instrument}1$ and then receives $\textit{instrument}2$. This is defined for the lead surgeons main hand.

Definition 2 of an instrument change is directly based on the sequences of entries in the event-log. It is therefore defined after the event-log section.

3.6.2 Event-log

We manually processed two videos and created an event-log by writing down the composition of instrument changes. For example, assume that on frame number 24501, the visible set of instruments changes from consisting of 1 retractor and 1 needle driver, to now also including 1 diathermy. Then 24501 together with all instruments currently in view would become one entry in the event-log. The log would become very long if an entry was added every time the composition of visible instruments change since they are periodically obscured from view by the surgeon's head. Because of this, we added entries to the log based on two rules. The first rule was that an entry is added if an instrument becomes visible and was not a member of the most recent entry. Secondly, an entry was added when an instrument was not viewable anymore and does not return within one minute or if another, new instrument appears on the screen while the previously visible instrument still was out. Each instrument class in the event-log is represented as either visible or not visible. There was most often only one instrument per class used at a time, so the number of visible instruments in each class was not recorded. The forceps class was an exception but was not recorded in the event-log.

Definition 2

An instrument change is an instrument transitioning from **active** to either **inactive** or **outside**. An **active** instrument refers to the instrument the lead surgeon is assumed to currently be using, but it is not guaranteed that it perfectly corresponds to what the lead surgeon actually used. When an instrument becomes visible, it is set to **active**. An **inactive** instrument refers to an instrument the lead surgeon is not using but is still visible in the frame. A **waiting** instrument refers to an instrument that has left the frame but has not been switched out to another instrument. The **outside** instrument refers to an instrument that has left the frame and has been switched to another *or* never entered the frame.

Example - Definition 2

It is not guaranteed that the inactive and active elements perfectly correspond to what the surgeon actually used. An explanation of definition 2 is provided below, refer to table 3.5.

- **Frame id:190**. The scalpel entered the frame when nothing else is visible. Because this is the first instrument in the event-log, it is counted as a change, even though no instrument was replaced. The scalpel is set as **active**.
- **Frame id:231**. The scalpel left the frame. It is not interpreted as a change yet, another element must enter the frame first. The scalpel is set as **waiting**
- **Frame id:301**. The diathermy entered the frame, this counts as a change with the scalpel that was **waiting**. The diathermy is set as **active** and the scalpel to **outside**.
- **Frame id:702**. The needle driver entered the frame, this counts as a change because it became visible. It is changed with the diathermy that is set to **inactive**– because it remains visible– and the needle driver to **active**.
- **Frame id:1302**. The needle driver left the frame and is set to **waiting**. It is not interpreted as a change yet, another element has to enter the frame first. The diathermy remains **inactive** and will do so until it is no longer visible.

- **Frame id:1532**. A scalpel entered the frame, this counts as a change because it became visible. It is changed with the needle driver that is set to **outside** and the scalpel to **active**.

- **Frame id:1905**. The diathermy left the frame. This does not count as a change since the status of the diathermy was **inactive**. The diathermy is set to **outside**.

At frame **frame id:1302** when the needle driver disappears, only the diathermy is visible, yet it remains **inactive**. It is possible that the surgeon starts using the diathermy, in which case the change between needle driver and diathermy will not be noticed. It is also possible that the diathermy will not be used again before it finally exits the video, because it happens that the diathermy remains visible for more than a minute after it has been replaced, before being removed from the video.

Frame id	Visible instruments	Accumulated changes	Event
190	Scalpel	1	Nothing switched to scalpel
231	No elements	1	Scalpel not visible
301	Diathermy	2	Scalpel switched to Diathermy
702	Diathermy Needle driver	3	Diathermy switched to Needle driver
1302	Diathermy	3	Needle driver not visible
1532	Diathermy Scalpel	4	Needle driver switched to Scalpel
1905	Scalpel	4	Diathermy not visible

Table 3.5: Event-log. Active element corresponds to green text and inactive corresponds to red text.

Interpreter script

This script was designed to interpret network output. It was intended to prevent false detections while also attempting to fill in the gaps where the network failed to detect a specific instrument. We accomplished this by passing the network's detection output through this script, which first processed the data using various rules designed to eliminate network failures, such as missing an instrument in a few frames but detecting it in all frames around that time. After the data has been processed, it is passed through another section of the script that splits the data into groups for each instrument, with each group representing an event.

The first step filtered the frames contained in the input data. For each frame, we followed the steps presented below.

- Check for object overlap. If two objects existed at the same spot in the frame, the instrument with the lowest detection confidence was removed.

- Check for flicker. If an object occurred in one frame but not in the surrounding frames, it was interpreted as a flicker and removed.

- Check for duplicates. If there were multiple occurrences of objects that should only occur once, at a time, the object with the lowest confidence was removed.

We generated a list for every instrument describing when they are detected in the video. This was done by iterating through the frames of the video and focusing on individual instruments. The elements of each list were divided into groups, where the time difference of each group had to be larger than one minute. A third step involved removing all groups that did not follow a set of rules. This set of rules, specified how long an object had to be recognized before it could be considered a true detection. Finally, we compared all groups from different instruments to check if any overlap existed. Our goal was to follow the principle presented in **Definition 2**. These steps created an **Event-log**.

3.6.3 Evaluation -Instrument change

The evaluation was carried out in two steps. Firstly an evaluation was carried out on how well the network in combination with the interpreter script could create an event-log. Secondly, an evaluation was carried out regarding how well instrument changes could be counted using the event-log.

A **ground truth event-log** was created manually by using two separate videos which did not occur in training data. This was compared to the **generated event-log**. To calculate precision, recall and F1-score the following definitions below were used.

- TP(true positives) How many detected instrument events occurred in reality as well.
- FP(false positives) How many detected instrument events did not occur in reality.
- FN(false negative) How many events occurred in reality but were not noticed by the network.

Using the **generated event-log** and the **ground truth event-log**, instrument switches for both event-logs were calculated using **Definition 2**. These were compared to each other in order to see how well they matched.

3.7 Extended dataset- YOLOv5

The best network was further improved with the following steps.

3.7.1 Analysing incorrect detections on videos and retrieving frames

The goal was to improve the network and the approach followed was to use the trained network and analyze predictions using the analysis script presented below. The output from the analysis script was annotated and used to extend the ODS. This extended dataset is referred to as the extended dataset, EDS.

3.7.2 Weakness analysis script

The analysis script takes network predictions as input and outputs a log with difficult frames. This is a modified version of the interpreter script, it follows the same steps as the interpreter script presented in section 3.6.2. The only difference is that all modified frames were saved to a log since they are interpreted as difficult.

3.7.3 Evaluation of the analysis script

To present the total improvement, the model trained on EDS was compared to the model trained on ODS with respect to mAP evaluating on ITDS. The models were also compared with a focus on how well they could detect instrument changes.

3.8 Computer setup

The hardware and software setup that was used in this thesis is displayed below.

Hardware	Product name
Motherboard	Asus PRIME Z690M-PLUS D4
CPU	12th Gen Intel(R) Core(TM) i9-12900K
GPU	NVIDIA GeForce RTX 3090 24 GB

Software	Version
CUDA	11.3
Cmake	3.20
OpenCV	4.2.0
cudNN	8.2.0
Ubuntu	20.04

The implementation of Efficientdet assumed eight powerful GPUs would be available. We used one GPU which means Efficientdet most likely was limited by the available hardware. The other models had no such requirements and they only needed minor changes, as described in section 3.3.1.

Chapter 4

Results

This chapter presents the result of the network evaluation, instrument switches and weakness analysis.

4.1 Network comparisons - mAP_{50}

In order to find the highest performing network, we evaluated on ITDS and the ODS test-split. The results are displayed in table 4.1 as a mAP_{50} and $mAP_{50:95}$ score. The mAP with an IOU threshold set to 0.5 is represented as mAP_{50} and the mAP with an IOU threshold from 0.5 to 0.95 is represented as $mAP_{50:95}$. The table also includes the average of mAP_{50} and $mAP_{50..95}$.

YOLOv5 has the highest average mAP_{50} score of 89.23 % compared to the second-best performing network, Scaled-YOLOv4 that has an average mAP_{50} of 88.0 %, see table 4.1. Additionally, YOLOv5 has the highest average $mAP_{50:95}$ score of 69.31 % compared to Scaled-YOLOv4 which has an average $mAP_{50:95}$ of 61.74 %, this corresponds to a difference of 7.57 percentage points.

	ODS - Test-split		ITDS		Average	
Network	mAP_{50}	$mAP_{50:95}$	mAP_{50}	$mAP_{50:95}$	mAP_{50}	$mAP_{50:95}$
YOLOv5	94.5%	76.14 %	83.96 %	62.48 %	89.23 %	69.31 %
Scaled-YOLOv4	93.36 %	69.02%	82.79 %	54.46 %	88.0 %	61.74 %
FasterR-CNN	92.23 %	63.27%	81.39 %	55.31 %	86.81 %	59.29 %
YOLOv4	90.6 %	58.00 %	80.74 %	46.09 %	85.67 %	52.04 %
Efficientdet	85.11 %	59.0 %	70.24 %	46.06 %	77.67 %	52.71 %
retinaNet	78.43 %	54.8 %	65.55 %	44.20 %	71.99 %	49.5 %
SSD	78.22 %	43.43 %	63.80 %	32.83 %	71.01 %	38.63 %

Table 4.1: mAP scores for ODS test-split and ITDS

4.2 Instrument comparison

The results below show how well YOLOv5 can handle the instruments, measured with AP_{50} and $AP_{50:95}$. Table 4.2 shows results using both ODS test-split and ITDS for evaluation.

The results in table 4.2 show that HL-tube has an AP_{50} score of 86.27 % when evaluated on the ODS test-split and 52.33 % when evaluated on the ITDS. The scalpel show a decrease from an AP_{50} score of 93.46 % when evaluated on the ODS test-split to 74.11 % when evaluated on the ITDS. The retractor show a decline in AP_{50} score from 95.78 % when evaluated on the ODS test-split to 79.75 % when evaluated on the ITDS. The diathermy, forceps, needle driver and saw display similar results when comparing AP_{50} scores from the ODS test-split as compared to the ITDS.

	ODS test-split		ITDS	
Instrument	AP_{50}	$AP_{50:95}$	AP_{50}	$AP_{50:95}$
Diathermy	97.42 %	79.1 %	95.58 %	70.7 %
Forceps	96.08 %	83.3 %	94.14 %	82.1 %
Scalpel	93.46 %	77.0 %	74.11 %	56.6 %
Needle driver	92.69 %	78.1 %	90.80 %	75.9 %
Retractor	95.78 %	83.6 %	79.75 %	63.2 %
HL tube	86.27 %	44.0 %	52.33 %	16.2 %
Saw	99.79 %	87.6 %	100 %	72.3 %

Table 4.2: mAP scores for all instruments, using ODS test-split and ITDS

4.3 Data scalability

Figure 4.1 shows training results for YOLOv5, retrieved by training on multiple datasets of different sizes (see section 3.5). mAP_{50} and $mAP_{50:95}$ are shown for both ODS test-split and ITDS.

All curves show an overall improvement which demonstrates that both mAP_{50} and $mAP_{50:95}$ improve with more data. The red curve, representing the $mAP_{50:95}$ score for ITDS show a

decrease of 0.02 percentage points when trained on 1600 images compared to 800 images. The green curve, representing the mAP_{50} score for ITDS show a decrease of 0.03 percentage points when trained on 1600 images compared to 800 images. The reason for this decrease is not entirely certain but one hypothesis is discussed in section 5.1.3

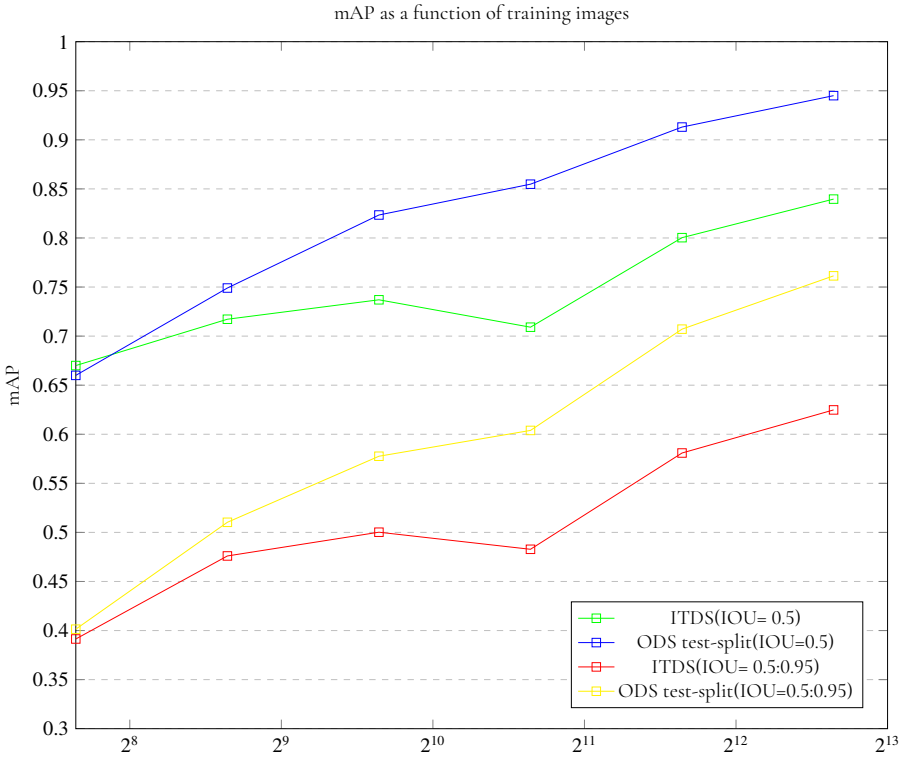


Figure 4.1: Shows YOLOv5 performance in mAP on training-sets of different sizes. The x-axis is the training-set size.

4.4 Instrument changes

Table 4.3 shows precision, recall and F1-score for two videos and the average for these metrics when generated event-log is compared to the ground truth event-log. The main reason for the high score of the Hl-tube and retractor is that they are just switched in once, which was detected. The reason for the high score of the saw is that the network perform well on that instrument and that it is only switched in a few times per video.

Video ID	Metric	Diathermy	Scalpel	Needle driver	Retractor	Saw	HL tube
Video 1	Precision	83.3 %	100 %	65.5 %	100 %	100 %	100 %
	Recall	100 %	71.4 %	84 %	100 %	100 %	100 %
	F1-score	90.9 %	83.3 %	73.6 %	100 %	100 %	100 %
Video 2	Precision	94.7 %	93.7 %	84.8 %	100 %	100 %	100 %
	Recall	90 %	68.1 %	82.3 %	100 %	100 %	100 %
	F1-score	92.3 %	78.9 %	83.5 %	100 %	100 %	100 %
Average	Precision	89 %	96.8 %	75.1 %	100 %	100 %	100 %
	Recall	95 %	69.7 %	83.1 %	100 %	100 %	100 %
	F1-score	91.6 %	81.1 %	78.5 %	100 %	100 %	100 %

Table 4.3: Precision, recall and F1-score for two separate surgery videos.

Table 4.4 presents how many instrument switches that were detected in each video based on the **generated event-log** and the **ground truth event-log**, see definition of these in section 3.6.3. One instrument switch corresponds to the case when an instrument is considered to be switched out to another.

Video ID	Generated event-log	Ground truth event-log
Video 1	27	26
Video 2	36	43

Table 4.4: Number of calculated instrument switches for two separate surgery videos.

4.5 Extended dataset - YOLOv5

4.5.1 YOLOv5 improvements

The bottom row of table 4.5 shows results for YOLOv5 when trained on EDS and evaluated on ITDS. The result shows an increase of 4.5 percentage points when comparing to YOLOv5 trained on ODS (see table 4.5 and 4.1), with respect to mAP_{50} on ITDS. It also shows an increase in $mAP_{50:95}$ of 2.5 percentage points. Figure 4.1 shows the effect training data size has on mAP, where mAP_{50} on average increases with 3.5 percentage points when the training set is doubled, evaluated on ITDS. The training set of EDS contains 139 percent more images than the ODS training set, and with an increase of 4.5 percentage points, the extended dataset upholds the trend we observe in figure 4.1.

4.5.2 Instrument comparison

Table 4.5 shows results from evaluating ITDS with the new model trained on EDS. The AP_{50} of the HL tube is 64.9% compared to 52.33% when trained on ODS while the score of the retractor was increased to 88.3% from 79.75%. This correspond to an increase of 12.57 percentage points for the HL tube and 8.55 percentage points for the retractor.

ITDS - YOLOv5		
Instrument	AP_{50}	$AP_{50:95}$
Diathermy	97.84 %	71.67 %
Forceps	96.12 %	82.49 %
Scalpel	77.2 %	57.34 %
Needle driver	95.14 %	78.8 %
Retractor	88.33 %	70.1 %
HL tube	64.9 %	21.5 %
Saw	100 %	72.5 %
Network	mAP_{50}	$mAP_{50:95}$
YOLOv5	88.5%	64.93 %

Table 4.5: AP score for all instruments and mAP for YOLOv5.

4.5.3 Instrument changes

Table 4.6 shows the results for the event-log that the new model in cooperation with the analysis script created, calculated based on the ground truth event-log. The results are generated by using the same videos as in section 4.4. The F1 score for the needle driver increased from 78.5 % when evaluating YOLOv5 trained on ODS to 83.75 % when evaluating YOLOv5 trained on EDS. The scalpel increased from 81.1 % to 91.65 %. This would correspond to an increase for the needle driver of 5.25 percentage points and 10.55 percentage points for the scalpel.

Video ID	Metric	Diathermy	Scalpel	Needle driver	Retractor	Saw	HL tube
Video 1	Precision	100 %	87.5 %	67.7 %	100 %	100 %	100 %
	Recall	100 %	100 %	84 %	100 %	100 %	100 %
	F1-score	100 %	93.3 %	75 %	100 %	100 %	100 %
Video 2	Precision	100 %	100 %	93.3 %	100 %	100 %	100 %
	Recall	95 %	81.8 %	91.1 %	100 %	100 %	100 %
	F1-score	97.4 %	90 %	92.5 %	100 %	100 %	100 %
Average	Precision	100 %	93.75 %	80.5 %	100 %	100 %	100 %
	Recall	97.5 %	90.9%	87.55 %	100 %	100 %	100 %
	F1-score	98.7 %	91.65 %	83.75 %	100 %	100 %	100 %

Table 4.6: Precision, recall and F1-score for two separate surgery videos.

Table 4.7 presents how many instrument switches were counted in each video based on the **generated event-log** and the **ground truth event-log**, see definition of these in section 3.6.3. The results on Video1 show an difference of 5 switches compared to YOLOv5 trained on ODS which had a difference of 1 switch. The results on Video2 show an difference of 4 switches compared to YOLOv5 trained on ODS which had a difference of 7 switch.

Video ID	Generated event-log	Ground truth event-log
Video 1	31	26
Video 2	39	43

Table 4.7: Number of calculated instrument switches for two separate surgery videos.

Chapter 5
Discussion and Future work

5.1 Results

5.1.1 Network comparisions - mAP_{50}

When choosing the best network for instrument detection, the average mAP_{50} score is in focus instead of $mAP_{50:95}$, since mAP_{50} describes how well we can detect instruments while $mAP_{50:95}$ describes more how well the bounding boxes are fitted to the actual instrument. In this application, it is more important to detect an instrument, than to detect it with a well-fitted bounding box.

YOLOv5 is the best-performing model of the mAP comparison. It did not only have the highest performance with respect to mAP_{50} for both ODS test-split and ITDS but also the highest average $mAP_{50:95}$ score. The difference between YOLOv5 and the other networks when it comes to the average $mAP_{50:95}$ score is bigger than when comparing mAP_{50}. The difference between YOLOv5 and Scaled-YOLOv4 in terms of average $mAP_{50:95}$ score, is 7.57 percentage points. The difference of 7.57 percentage points could indicate that YOLOv5 is much better at actually fitting the bounding boxes which may be important for future applications that focus more on following the exact location of the instruments. Since YOLOv5 has the best performance, it was further used to detect instrument changes and for iterative improvement (see section 4.4 4.5.2).

5.1.2 Instrument comparison

The results presented in table 4.2 for the HL-tube are considered low when compared to the other instruments. The bad results that the HL-tube demonstrates on the ITDS show that the network seems to handle the instrument much worse on new unseen data. The reason for this is most likely that, compared to the other instruments, the HL-tube does not have any well-defined borders, causing a greater variability in the ground truth bounding boxes for

similar-looking HL-tubes, because the process of annotation becomes more difficult. This variability could mean that the network has difficulties detecting the HL-tube on unseen data from ITDS as compared to data similar to training data from the ODS test-split.

As can be seen in table 4.2, the scalpel is more difficult for the network when evaluating on ITDS as compared to the ODS test-split. One possible reason is that there are fewer scalpels in the training dataset than other instruments, ODS contain 1454 scalpels, 10181 forceps and 2271 needle drivers, see table 3.3. It also has similar properties to both the forceps and the needle driver, they all being long metallic objects. Another possible reason is that the scalpels in the dataset do not have a good variation because they occur so rarely in surgery, many images with similar-looking scalpels exist in the dataset.

The decline that can be seen for the retractor in table 4.2 when comparing the AP_{50} score of the ODS test-split, could be the result of the annotation variation, see section 5.4.2. The saw shows great results, even though it only occurs 390 times in the ODS. The reason for this could be the distinct shape that the saw has, as can be seen in figure 3.2e.

The diathermy, forceps, saw and needle driver display similar results when comparing AP_{50} score on the ODS-testsplit to the AP_{50} on ITDS. This is a promising result since the network handles these instruments similarly on completely new data from ITDS as compared to data similar to training data from the ODS.

5.1.3 Data scalability

As can be seen in figure 4.1, the x-axis grows with a factor 2, meaning each datapoint is generated with twice the data size from the previous one. The curves have a more or less constant slope, indicating diminishing returns when adding more images. To further increase performance, more images could be added to the dataset and a possible way to do this in a more effective manner is to use synthetic data, as discussed in section 5.5.2. Observing figure 4.1, there is a slight decrease when evaluating ITDS with YOLOv5 trained on 1600 images. The reason for this is difficult to establish since the network is complex. One reason could have been that the dataset containing 1600 images had less similarities to the ITDS than the dataset containing 800 images.

5.1.4 Instrument changes

As can be seen in table 4.3, the combination of the interpreter script and the trained network gave good results when the generated event-log was compared to the ground truth event-log, even though there are flaws with the YOLOv5 predictions when evaluating unseen videos. It shows good performance with all instruments, with the worst results on the scalpel and the needle driver.

The results in table 4.4 indicate that the combination of the generated event-log and the definition of an instrument switch, see section 3.6.1, yields a similar number of instrument changes, as compared to the number of switches retrieved from the ground truth event-log. The actual number of changes in each of the videos as presented in section 3.6 are 70 which

is more than double the results in table 4.4. The reason for this difference is that we defined an instrument change in a way that the network would be able to process. The results could be improved if a more realistic instrument switch definition was possible to use.

It is not possible to detect all instrument switches with the network and interpreter script. The reason for this is mostly that the network makes some errors on unseen videos which will be registered as an instrument switch when running the output through the analysis script. In new videos, there is a risk that it is a new angle, object, or different lightning that the network has not seen before. For example, a pencil was confidently detected as a needle driver even though it has no similarities, but we had no occurrences of any pencil throughout our training data. The combination of the instrument changes script and network output would of course perform better if the network improved. One additional reason for not being able to notice all switches is that the event-log and our instrument switch definition does not directly correspond to reality.

5.1.5 Extended dataset - YOLOv5

YOLOv5 improvements

The mAP_{50} score for YOLOv5, generated by training on EDS, follows the data scalability trend which can be observed in figure 4.1. This shows that collecting more data by using the network itself is a viable approach, especially as the annotation time is reduced because frame retrieval is not used anymore.

Instrument comparison

Observing table 4.5 and table 4.1, wee see that AP_{50} and $AP_{50:95}$ has increased for all instruments. The instruments with the highest percentage point increase is the needle driver, HL tube and retractor. This indicates that the new YOLOv5 model handles new, unseen data better since the evaluation was performed on ITDS.

Instrument changes

The results displayed in table 4.6 show promising results for the improvement of the event-log when compared to table 4.3. The improved ability to detect events was expected after knowing that the new YOLOv5 model handles new data better with respect to mAP.

The number of calculated instrument switches for YOLOv5 trained on EDS was more off than YOLOv5 trained on ODS. The reason for this could be the switch definition which was created to cover flaws of the network. A better performing object detector might not work as well with the interpreter script. It cannot be concluded that YOLOv5 trained on ODS detect switches better since this is just a comparison on the total number of switches rather than if the detected switches are correct.

5.2 Model choice

There exist many object detection networks that we could have used for the comparison in this project. We choose these seven networks since they had shown evidence of good performance and since our time frame was limited so there was no time for a more complete search. There are multiple implementations of each of the networks with both minor and major differences. We choose one implementation of each network, the choice was based on which appeared to be the most recently updated version and how straightforward it was to use.

5.2.1 Ease of use

Some networks were easier to use than others, to get them running, and to adjust and understand the hyper-parameters. If the goal would have been to optimize a network for our task, then some version of YOLO would have been preferred since these networks are well documented.

5.3 Hardware limitations

Because the training takes place on the GPU, it is critical to have a GPU that can handle the network's complexity and volume of data efficiently. The GPU was limited by memory shortage and computational power.

5.3.1 GPU memory shortage

To run a network on the GPU, a specific amount of GPU memory is required. In order to reduce and fit the amount of GPU memory required to our hardware, two techniques were used. First, one may reduce the input image resolution to reduce the GPU's memory requirements, but this would degrade image quality. Another alternative is to reduce the batch size. Increased GPU memory and/or the number of GPUs would have allowed the model to train on high-resolution images while maintaining a well-adjusted batch size, which could have improved the training of the networks reported in this thesis. Because the networks cannot be completely utilized with the existing arrangement, the lack of GPU memory may have had an impact on the network ranking presented in section 4.1, since some networks may perform better with extended hardware than others.

5.4 Dataset

There are many things that could be tested regarding the dataset. For instance, annotation techniques, dataset balance and variation.

5.4.1 Unbalanced data

A straightforward way to extract frames from a video is to extract one frame every other second or one frame every ten seconds, for example. However, the danger of completely

missing certain instruments that are not utilized very often is rather great with this method. The scalpel, for example, is only used in short intervals. In addition, some instruments, such as the forceps, appear in practically every frame. This would result in a very unbalanced data set with many forceps and few scalpels. Our method of extracting frames, presented in section 3.1.3 was therefore used to create a more balanced dataset.

5.4.2 Rules of annotation

Section 3.1.3 presents how annotations were conducted, where all *clearly visible* parts of an object are included in a bounding box, even though it is partially obscured. This does, however, present an issue, since what is *clearly visible* is subjective. In figure 3.3c, the lower left part of the retractor is still visible, but the bounding box excludes this region. A network detection that includes this small part will receive a large error, even though it can be regarded as correct because the retractor is still visible. The dataset contains many examples of these difficult objects, where the size of the bounding box is affected by who annotated it. This variation in annotation could cause the network to perform worse. Observing the results for the retractor in table 4.2, it seems to handle unseen data from ITDS worse than data similar to the training data from ODS, one possible factor for this could be the variation in the annotation. It is difficult to say how or how much it was affected by this. This could have been improved at the outset by either establishing more stringent criteria for annotating all of the various instruments or allowing only one person to annotate, which would, however, have been inefficient.

5.5 Improvements

5.5.1 Annotation strategies

In this work we choose a specific strategy when annotating the frames retrieved from the videos, specifically to only annotate the instrument itself. Other strategies could have been chosen, like for example to include the hand when annotating the instruments.

We conducted such an experiment with a dataset of 2000 images, where the annotation included both the instrument and the hand that held it. This showed convincing results. Due to the fact that most instruments are held differently, this strategy might make it easier to notice different instruments. It might also be easier to notice instrument switches because an actual switch is made when the instrument is no longer in the hand and not when it is no longer in the frame.

5.5.2 Synthetic data

Synthetic data has been used in many applications to improve performance, it can help with data imbalance, data variation and data quantity. Kiyohito et al. used 3D models to generate synthetic data to improve the dataset on which they trained their CNN network [34]. If accurate and reliable 3D models of the instruments were acquired, it could be worth exploring their approach to improve the dataset. Ekbatani and Pujol used synthetic data to train their model on recognizing the number of pedestrians [35]. This approach could be used to further

improve our model by improving the dataset balance, variation and to provide data quantity in a more efficient way.

5.5.3 Hardware

The computational power of the GPU will affect the total time it takes to train the network. If future goals are to add data and retrain the model for each newly recorded surgery, then the training speed will be of significant importance. So if this is the future goal, it could be worth considering adding another GPU or upgrading the current GPU. Another reason for upgrading the hardware or choosing to test cloud services would have been to experiment more with input parameters such as image size, network models, and batch size. We have been forced to abstain from testing optimal settings due to hardware limitations.

5.6 Future work

This thesis presents results that can be used to develop AI and robotics applications. To do this in practice, all instruments used during surgery must be included in the dataset. One aspect that has to be improved is how to handle situations when, for example, a surgeon's head covers a part of the frame. One approach that could be investigated is to annotate the head as a separate class and use its position to analyse which instruments may be under it. Another approach is to use additional cameras from different angles to create a broader picture of what is happening. Possible future applications include tools for automated video annotation, a tool for automatic feedback and analysis for the surgeons and in the long run, robot applications such as a robotic operating room nurse that hands over instruments to the surgeon.

Chapter 6
Conclusion

We begin this chapter by answering the research questions presented in section 1.1.

- Which object detector network performs best according to the metric mean average precision (mAP)?

  - YOLOv5

- How does the best network handle different instruments?

  - It handles the forceps, needle driver, diathermy and saw well. The worst instruments for the network is the scalpel and the HL tube.

- How does the best model react to different amounts of training data?

  - The performance is increased by on average 3.5 percentage points when the dataset doubled.

- How well can the model detect instrument changes?

  - Our approach to detecting can be considered acceptable to detect instrument changes. To get a more reliable detector for instrument changes, further development would be required and a new more realistic switch definition should be introduced.

- How does a model trained on the extended dataset compare to the best model?

  - YOLOv5 has an mAP_{50} score of 88.5 % which corresponds to an increase of 4.5 percentage points. It also showed better performance when detecting instrument events.

We conclude that object detectors based on deep learning can be used to detect surgical instruments in images retrieved from open-heart surgery videos. An evaluation tool that counts instrument changes performed by the surgeon can be created based on the output from such an object detector. Although the evaluation tool shows promising results, further improvements would have to be made in order for surgeons to be able to trust the results. Two ways to potentially improve the evaluation tool are to use a more realistic switch definition and add additional classes for the object detector to recognize.

References

[1] Pediatric heart surgery. https://www.socialstyrelsen.se/kunskapsstod-och-regler/regler-och-riktlinjer/nationell-hogspecialiserad-vard/arlig-uppfoljning/tillstandsomrade/hjartkirurgi-pa-barn-och-ungdomar. Accessed: 2022-05-15.

[2] Hang Cheng, Jeffrey W. Clymer, Brian Po-Han Chen, Behnam Sadeghirad, Nicole C. Ferko, Chris G. Cameron, and Piet Hinoul. Prolonged operative duration is associated with complications: a systematic review and meta-analysis. *Journal of Surgical Research*, 229:134–144, 2018.

[3] Natalia Cornellà, Joan Sancho, and Antonio Sitges-Serra. Short and long-term outcomes after surgical procedures lasting for more than six hours. *Scientific Reports*, 7(1), 2017.

[4] Timothy D. Jackson, Jeffrey J. Wannares, R. Todd Lancaster, David W. Rattner, and Matthew M. Hutter. Does speed matter? the impact of operative time on outcome in laparoscopic surgery. *Surgical Endoscopy*, 25(7):2288–2295, 2011.

[5] Sandeep Ganni, Sanne M. Botden, Magdalena Chmarra, Richard H. Goossens, and Jack J. Jakimowicz. A software-based tool for video motion tracking in the surgical skills assessment landscape. *Surgical Endoscopy*, 32(6):2994–2999, 2018.

[6] Tzutalin. labelimg. https://github.com/tzutalin/labelImg, 2015.

[7] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. The MIT Press, 2017.

[8] Saad Albawi, Tareq Abed Mohammed, and Saad Al-Zawi. Understanding of a convolutional neural network. In *2017 International Conference on Engineering and Technology (ICET)*, pages 1–6, 2017.

[9] Zhengxia Zou, Zhenwei Shi, Yuhong Guo, and Jieping Ye. Object detection in 20 years: A survey, 2019.

[10] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection, 2016.

[11] Xianzhi Du, Tsung-Yi Lin, Pengchong Jin, Golnaz Ghiasi, Mingxing Tan, Yin Cui, Quoc V. Le, and Xiaodan Song. Spinenet: Learning scale-permuted backbone for recognition and localization, 2019.

[12] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. Yolov4: Optimal speed and accuracy of object detection, 2020.

[13] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection, 2016.

[14] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection, 2018.

[15] Rafael Padilla, Wesley L. Passos, Thadeu L. B. Dias, Sergio L. Netto, and Eduardo A. B. da Silva. A comparative analysis of object detection metrics with a companion open-source toolkit. *Electronics*, 10(3), 2021.

[16] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *International Journal of Computer Vision*, 88(2):303–338, June 2010.

[17] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollár. Microsoft coco: Common objects in context, 2014.

[18] Luis Perez and Jason Wang. The effectiveness of data augmentation in image classification using deep learning, 2017.

[19] Karl Weiss, Taghi M. Khoshgoftaar, and DingDing Wang. A survey of transfer learning. *Journal of Big Data*, 3(1), 2016.

[20] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks?, 2014.

[21] M. Everingham, S. M. A. Eslami, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The pascal visual object classes challenge: A retrospective. *International Journal of Computer Vision*, 111(1):98–136, January 2015.

[22] Glenn Jocher, Ayush Chaurasia, Alex Stoken, Jirka Borovec, NanoCode012, Yonghye Kwon, TaoXie, Jiacong Fang, imyhxy, Kalen Michael, Lorna, Abhiram V, Diego Montes, Jebastin Nadar, Laughing, tkianai, yxNONG, Piotr Skalski, Zhiqiang Wang, Adam Hogan, Cristi Fati, Lorenzo Mammana, AlexWang1900, Deep Patel, Ding Yiwei, Felix You, Jan Hajek, Laurentiu Diaconu, and Mai Thanh Minh. ultralytics/yolov5: v6.1 - TensorRT, TensorFlow Edge TPU and OpenVINO Export and Inference, February 2022.

[23] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg. Ssd: Single shot multibox detector. *Lecture Notes in Computer Science*, page 21–37, 2016.

[24] Mingxing Tan, Ruoming Pang, and Quoc V. Le. Efficientdet: Scalable and efficient object detection, 2020.

[25] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks, 2016.

[26] Geert Litjens, Thijs Kooi, Babak Ehteshami Bejnordi, Arnaud Arindra Adiyoso Setio, Francesco Ciompi, Mohsen Ghafoorian, Jeroen A.W.M. van der Laak, Bram van Ginneken, and Clara I. Sánchez. A survey on deep learning in medical image analysis. *Medical Image Analysis*, 42:60–88, 2017.

[27] Nilay Ganatra. A comprehensive study of applying object detection methods for medical image analysis. In *2021 8th International Conference on Computing for Sustainable Global Development (INDIACom)*, pages 821–826, 2021.

[28] Zijian Zhao, Tongbiao Cai, Faliang Chang, and Xiaolin Cheng. Real-time surgical instrument detection in robot-assisted surgery using a convolutional neural network cascade. *Healthcare Technology Letters*, 6(6):275–279, 2019.

[29] Yoshiko Bamba, Shimpei Ogawa, Michio Itabashi, Shingo Kameoka, Takahiro Okamoto, and Masakazu Yamamoto. Automated recognition of objects and types of forceps in surgical images using deep learning. *Scientific Reports*, 11(1), 2021.

[30] Lu Tan, Tianran Huangfu, Liyao Wu, and Wenying Chen. Comparison of retinanet, ssd, and yolo v3 for real-time pill identification. *BMC Medical Informatics and Decision Making*, 21(1), 2021.

[31] Shrey Srivastava, Amit Vishvas Divekar, Chandu Anilkumar, Ishika Naik, Ved Kulkarni, and V. Pattabiraman. Comparative analysis of deep learning image detection algorithms. *Journal of Big Data*, 8(1), 2021.

[32] S A Sanchez, H J Romero, and A D Morales. A review: Comparison of performance metrics of pretrained models for object detection using the TensorFlow framework. *IOP Conference Series: Materials Science and Engineering*, 844:012024, jun 2020.

[33] Austen Groener, Gary Chern, and Mark Pritt. A comparison of deep learning object detection models for satellite imagery. *2019 IEEE Applied Imagery Pattern Recognition Workshop (AIPR)*, Oct 2019.

[34] Matthew Z. Wong, Kiyohito Kunii, Max Baylis, Wai Hong Ong, Pavel Kroupa, and Swen Koller. Synthetic dataset generation for object-to-model deep learning in industrial applications, 2019.

[35] Hadi Keivan Ekbatani, Oriol Pujol, and S. Seguí. Synthetic data generation for deep learning in counting pedestrians. In *ICPRAM*, 2017.

Appendices

Appendix A
Network implementations

All configuration files used for each network are provided on the drive linked below.

https://drive.google.com/drive/folders/1r5Aev3d2I3cOLCrLAqb83jIMOWbN-V3g?usp=sharing

A.1 YOLOv5

Download YOLOv5 from:
https://github.com/ultralytics/yolov5
Commit:7a2a118 (Mar 25, 2022)

A.2 YOLOv4

Download YOLOv4 from:
https://github.com/AlexeyAB/darknet
Commit:2c137d1 (Mar 3, 2022)

A.3 ScaledYOLOv4

Download ScaledYOLOv4 from:
https://github.com/WongKinYiu/ScaledYOLOv4
Commit:6768003 (Jun 16, 2021)

A.4 SSD

Download SSD from:
https://github.com/uvipen/SSD-pytorch
Commit:a9bc7b8 (Feb 10, 2021)

A.5 Efficientdet

Download Efficientdet from:
https://github.com/NVIDIA/DeepLearningExamples/tree/master/PyTorch/Detection/Efficientdet
Commit:228277d (Feb 18, 2022)

A.6 retinaNet and Faster-RCNN

Follow guide from:
https://detectron2.readthedocs.io/en/latest/tutorials/install.html
Download RetinaNet and Faster-RCNN from:
https://github.com/facebookresearch/detectron2
Commit:6999554 (Feb 18, 2022)

INSTITUTIONEN FÖR DATAVETENSKAP | LUNDS TEKNISKA HÖGSKOLA | PRESENTERAD 2022-06-03

EXAMENSARBETE Surgical Instrument Detection using Deep Learning
STUDENT Tobias Carlsson, André Svensson
HANDLEDARE Maj Stenmark (LTH), Phan Kiet Tran (SUS)
EXAMINATOR Elin Anna Topp (LTH)

Kirurgisk instrumentdetektion med djup maskininlärning

POPULÄRVETENSKAPLIG SAMMANFATTNING **Tobias Carlsson, André Svensson**

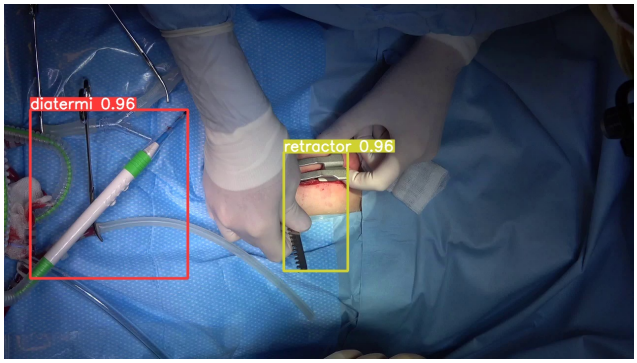
Att använda mjukvara för att identifiera kirurgiska instrument under barnhjärtkirurgiska ingrepp kan ge värdefull data om hur en kirurg använder sina instrument, exempelvis för att ge kirurgen återkoppling i deras arbete.

Det finns ett behov hos kirurger att kunna utvärdera sina egna prestationer för att utveckla de kirurgiska förmågorna. Detta kan bland annat göras genom att spela in operationer och titta igenom dessa. Det manuella arbetet att gå igenom de inspelade filmerna tar tid vilket är något som en kirurg inte har i överflöd. Därför kan det vara fördelaktigt med en annan metod att jämföra och utvärdera sin prestation, nämligen instrumentbyten.

Vetskapen om hur många byten som utförs under en operation kan hjälpa på många sätt. Det kan bland annat användas av erfarna kirurger, där kollegor kan jämföra med varandra. Exempelvis skulle en kirurg kunna ha 40 byten under en tidsperiod medan en annan kirurg endast har 20. Kirurgerna kan då lokalisera skillnader och jämföra operationsteknik.

För att räkna byten har vi evaluerat sju neurala nätverk för objekt-igenkänning som kan identifiera kirurgiska instrument under en barnhjärtoperation. Det nätverket som presterade bäst var YOLOv5. Vi använde sedan det bästa nätverket i kombination med ett analysprogram för att räkna antalet instrumentbyten som kirurgen utförde under operationen. De resultaten vi fått är lovande, och visar på att det är möjligt att räkna instrumentbyten. Instrumentbyten beräknas utifrån två steg. Det första steget är att en video analyseras av vårt nätverk som predikterar objekten som förekommer på varje bild. Resultatet blir en lista som beskriver vad som finns på varje bild. Det andra steget består av ett program som analyserar listan för att beräkna antalet byten.

Mjukvaran som räknar instrumentbyten kan även användas i utbildningssyfte då aspirerande kirurger kan använda verktyget kontinuerligt för att få en siffra som kan hjälpa till att beskriva deras prestation. För närvarande så sker evaluering ofta genom att en erfaren kirurg observerar utförandet och ger återkoppling. Det



kan vara fördelaktigt för studenten att också få data på hur det gick för att lättare kunna hitta förbättringsområden. Om till exempel studenten får feedback att operationen tog för lång tid eller att det var för stor osäkerhet med vilka instrument som skulle användas så kan studenten titta på antalet byten. Till exempel så kunde det varit 100 byten och studenten kan då följa sin utveckling ner mot färre byten. Denna minskning i instrumentbyten skulle kunna bero på förbättrad förståelse för vilket instrument som är lämpligt vid en given situation.

Vårt arbete kan komma till användning för att vidareutveckla mjukvaran som räknar instrumentbyten. Det kan också användas för att hitta andra evalueringsverktyg för kirurgerna. Ett exempel är tid med aktiva instrument, alltså tid som kirurgen aktivt använder de instrument hen håller i. En hypotes är att en erfaren kirurg bör vara mer effektiv och därför ha mindre tid med ett aktivt instrument.

Neurala nätverk för objektigenkänning är vanligt inom robotik. I framtiden kan nätverket användas som byggsten i robotapplikationer inom sjukvården, t ex för att räcka instrument till kirurgen eller sortera instrument efter disken.