OSTBAYERISCHE TECHNISCHE HOCHSCHULE REGENSBURG
INFORMATIK UND MATHEMATIK

OSTBAYERISCHE TECHNISCHE HOCHSCHULE REGENSBURG
REGENSBURG MEDICAL IMAGE COMPUTING

Detection of surgical instruments in laparoscopic videos using artificial neural networks with oriented bounding boxes

Bachelorarbeit
von

Sergi Saperas Lopez
Matrikelnummer: 3373576

Fakultät Informatik und Mathematik
Ostbayerische Technische Hochschule Regensburg
(OTH Regensburg)

Gutachter: Prof. Dr. Christoph Palm
Zweitgutachter: Prof. Dr. Axel Doering

Abgabedatum: February 9, 2023

Herr
Sergi Saperas Lopez
Dr.-Gessler-Str. 15A
93049 Regensburg

Studiengang: Medizinische Informatik

1. Mir ist bekannt, dass dieses Exemplar der Masterarbeit als Prüfungsleistung in das Eigentum des Freistaates Bayern übergeht.

2. Ich erkläre hiermit, dass ich diese Masterarbeit selbstständig verfasst, noch nicht anderweitig für Prüfungszwecke vorgelegt, keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie wörtlich und sinngemäße Zitate als solche gekennzeichnet habe.

Regensburg, den February 9, 2023

Sergi Saperas Lopez

Contents

1 Abstract

Artificial neural networks have unlocked countless applications in medical image computing. Thanks to artificial intelligence development over the past few years, it is possible for machines to visually detect in real time almost anything. This has proven very helpful for vision based minimally invasive surgery, like Laparoscopy.

The objective of this thesis is to implement a one-stage object detection network and train it to detect laparoscopic instruments in real time video images. Then, adapt the network to be able to predict rotating bounding boxes with arbitrary orientation in the most optimal way.

An object detection model was created based on SSD, and was trained to detect 18 different laparoscopic tools. Then this model was modified to not only detect the 18 different objects but also, to predict the angle and exact rotated bounding box measures. Both models shared pre-trained layers from VGG16 for feature extraction.

The first model proved to be capable of detecting laparoscopic tools with axis aligned bounding boxes accurately and precisely. The second model proved to be capable of more precisely fitting bounding boxes around objects, using rotated bounding boxes. However, further refinement is necessary for it to function optimally. Nevertheless, the model holds great potential in enhancing the performance of object detection in laparoscopic surgeries.

KEYWORDS: object detection, object tracking, computer vision, machine learning, deep learning, artificial neural network, surgical instruments, laparoscopic videos, oriented bounding boxes.

2 Acknowledgements

“I would like to extend my deepest gratitude to everyone in the lab for making me feel so welcomed during my time working on my thesis. Special thanks to Tobias Ruckert for his unwavering support and invaluable guidance throughout the entire process. His expertise and patience have been instrumental in the success of my research. I am truly grateful for all the time and effort he has invested in helping me. Thank you.”

3 Introduction

Minimally invasive surgery (MIS) has become one of the most rapid developing areas in modern surgery. Techniques like Laparoscopic Surgery have become the preferred approach for many surgical procedures [1]. It consists of using cameras to observe the internal anatomy without having to make large incisions in the skin, and therefore, reducing operative trauma, shortening hospitalization and speeding up recovery. However, minimally invasive operations are highly complex, and surgeons must deal with difficult hand-eye coordination, restricted mobility, and a narrow field of view. To support surgeons in their work, work is being done on robot-assisted methods based on the detection and tracking of surgical instruments in the human body. For this task, vision-based techniques have become the state-of-the-art method. Unlike other methods like the application of optical or electromagnetic markers on the instruments, vision-based techniques do not require expensive tool modifications and do not interfere with surgical workflow [2].

However, laparoscopic tool detection and tracking can be a very complex task for the following reasons. Firstly, there is a wide variety of different scenarios and tools. Secondly, complicated surgical scenes lead to limited inter-phase and high intra-phase variance. Lastly, endoscopic images are often blurry due to camera motion or occluded, sometimes even completely, by blood or gas from the tools. One of the best methods to deal with such a complex task is the use of an Artificial Neural Network or ANN. These methods consist of a set of algorithms, modeled after the human brain, that are designed to recognize patterns. Convolutional Neural Networks (CNN), are a type of neural network that performs exceedingly well on images, due to its high capability of feature extraction. Because of this, numerous studies have begun to apply CNN to the detection and tracking of surgical instruments in endoscopic videos [3].

There are many approaches on object detection and tracking, which can be optimized for a particular task.

Thesis Main Objective The bachelor thesis was conducted at Regensburg Medical Image Computing (ReMIC) lab and the main objectives are the following. Firstly, implementing a one-stage object detection network and training it to detect laparoscopic instruments in real time video images. Then, this network must be adapted to be able to predict rotating bounding boxes with arbitrary orientation in the most optimal way.

4 State of the Art

Object detection is a computer vision task that involves detecting instances of semantic objects of a certain class in digital images or videos and surrounding them with a box, usually referred to as a bounding box. It is a crucial component of many real-world applications such as self-driving cars, security systems, and surveillance cameras.

Over the years, various object detection methods have been proposed and developed, each with its own strengths and weaknesses. These methods can be broadly categorized into two types: single-stage and two-stage object detectors.

Single-stage object detectors, as the name suggests, are designed to perform object detection in a single step. They directly predict the final bounding boxes and class probabilities for the objects in the image. Examples of single-stage object detectors include YOLO (You Only Look Once) [4] and SSD (Single Shot MultiBox Detector) [3]. Single-stage object detectors are typically faster and more memory-efficient compared to two-stage detectors, but may have lower accuracy, especially for small objects or objects with complex shapes.

On the other hand, two-stage object detectors divide the task of object detection into two stages. The first stage generates a set of region proposals, which are regions in the image that may contain objects. The second stage classifies the objects within the region proposals and refines the bounding boxes. Examples of two-stage object detectors include Faster R-CNN (Region based Convolutional Neural Network) [5] and R-FCN (Region-Based Fully Convolutional Networks) [6]. Two-stage object detectors are known for their high accuracy, but are slower and more memory-intensive compared to single-stage detectors.

YOLO is a single-stage object detector that is fast and efficient. It divides an image into a grid of cells and applies a simple regression model to predict the bounding boxes and class probabilities for each cell. YOLO also uses anchor boxes to handle multi-scale objects and suppress false positive detections.

Anchor boxes, also known as prior boxes, are predefined bounding boxes used in object detection to help the model find objects within an image. These boxes serve as reference points for the model to compare and determine the size and shape of the objects it is trying to detect. The model then adjusts the size and location of these boxes based on the features of the objects it is detecting in the image.

The key innovation of YOLO is its single-stage architecture, which allows it to run in real-time on a single GPU. Unlike two-stage detectors such as Faster R-CNN, YOLO does not generate region proposals and instead predicts bounding boxes and class probabilities directly from the input image. This makes YOLO faster and more memory-efficient compared to two-stage detectors.

However, YOLO may have lower accuracy compared to two-stage detectors, especially for small objects or objects with complex shapes. Additionally, YOLO may struggle

with handling large numbers of classes and may not be able to handle more complex object shapes and handle more context information as well as two-stage detectors.

SSD is another single-stage object detector that uses a convolutional neural network to predict bounding boxes and class probabilities. Unlike YOLO and RetinaNet, SSD does not use anchor boxes, but instead uses multiple convolutional layers with varying scales to detect objects at different scales.

The key innovation of SSD is its use of multiple convolutional layers with varying scales, which allows it to detect objects at different scales in a single pass through the network. This makes SSD faster and more memory-efficient compared to two-stage detectors.

Faster R-CNN is a two-stage object detection method that is widely used for its high accuracy and speed. The first stage of Faster R-CNN generates region proposals using a region proposal network (RPN), which is a fully convolutional network that slides over the input image and predicts object proposals. The second stage classifies the objects within the region proposals using a separate network, typically a convolutional neural network.

One of the key innovations of Faster R-CNN is the use of an RPN to generate region proposals, which is much faster than previous methods that used selective search or sliding windows. The RPN also shares convolutional features with the classification network, which reduces memory consumption and speeds up inference time. Another advantage of Faster R-CNN is its ability to handle more complex object shapes and handle more context information compared to single-stage detectors such as YOLO and SSD. However, the two-stage architecture also makes Faster R-CNN slower and more memory-intensive compared to single-stage detectors.

R-FCN is an extension of the Faster R-CNN architecture and is designed to improve the speed and accuracy of object detection.

The main idea behind R-FCN is to separate the region proposal stage from the rest of the network, allowing the network to process multiple regions of the image in parallel. This leads to significant speed improvements compared to the traditional, Faster R-CNN architecture, where each region proposal is processed individually.

In R-FCN, the entire image is processed by a deep convolutional neural network, which generates a feature map. The feature map is then used to generate region proposals, which are fed into a set of fully convolutional layers to produce the final classification and bounding box regression outputs. This architecture allows the network to process numerous regions simultaneously, making it highly efficient for large scale object detection tasks. R-FCN is a highly effective and efficient architecture for object detection and has been used in many real-world applications, such as traffic monitoring, autonomous vehicles, and surveillance systems.

Arbitrary oriented object detection is a more challenging task than traditional object detection, as it requires the detection of objects in any orientation, rather than just those that are upright. This is particularly important in situations where objects may be tilted or rotated, such as in laparoscopic surgery. To address this issue, several recent studies have proposed methods for arbitrary oriented object detection.

Numerous approaches have been proposed to address this problem. These methods have been shown to improve the detection performance for oriented objects, but there

is still room for further improvement. These methods can be broadly divided into two categories: anchor-free methods and rotated proposal methods.

Anchor-free methods, such as FCOS (Fully Convolutional One-stage Object Detection) [7] and FreeAnchor [8],do not rely on predefined anchor boxes, and instead directly predict the object center and size. These methods typically use a single regression head to predict the object's center, size, and orientation, or they use multiple heads to predict different aspects of the bounding box. Anchor-free methods have been shown to improve the detection performance for oriented objects, as they are not restricted by the predefined anchor boxes. One popular anchor-free method is the CenterNet [9] architecture, which predicts the center point of an object and its size, orientation, and class. Another example is the CornerNet [10] architecture, which predicts the corners of an object's bounding box. These anchor-free methods have shown promising results in object detection tasks and are becoming increasingly popular in the field of computer vision.

Rotated proposal methods generate object proposals by using rotated rectangles or quadrilaterals to enclose the detected arbitrary oriented objects of interest. These proposals are then fed into a convolutional neural network for classification and regression, just like in traditional object detection methods. These methods have been shown to improve the detection performance for oriented objects, as they are specifically designed to handle arbitrary object orientations [11]. One popular rotated proposal method is the RRPN (Rotated Region Proposal Network) [12] architecture, which uses rotated rectangles to generate object proposals. Another example is the QuadNet architecture [13], which uses quadrilaterals to generate object proposals. Cascaded RetinaNet (Cas-RetinaNet)[14] is one of the latest oriented object detection models. The idea is to perform object detection in a multi-stage, hierarchical manner. This allows for a more effective and efficient use of computation, as the multiple stages can focus on detecting objects of different scales or at different levels of detail. SCRDet [15] is another multi-arbitrary oriented object detector. It works by first generating object proposals, then adjusting their scales to ensure consistency, and finally combining and refining the proposals to generate a set of accurate rectangles that tightly enclose the objects in the image. Even though these methods achieve high-efficiency oriented object detection, they are not accurate under low visibility conditions, that for example can occur in laparoscopic images.

In this thesis, the method used belongs to the single-stage kind of object detectors, and it is based on an SSD anchor-free architecture.

Object detection in laparoscopic surgery is an active area of research, with the goal of developing algorithms that can assist surgeons in identifying and locating surgical instruments and structures within the surgical field. Some of the object detection methods previously mentioned in this chapter, such as Faster R-CNN, YOLO or SSD have been shown to achieve high accuracy and real-time performance in detecting surgical instruments in laparoscopic images [16]. However, arbitrary oriented object detection for laparoscopic instruments is still an active area of research.

5 Theoretical Background

5.1 Artificial Neural Networks

Neural Networks, also known as ANNs, are a subset of machine learning and deep learning algorithms. Their aim is to learn how to perform a certain task by analyzing training examples. Their structure and terminology are modeled after the human brain, mimicking the communication between biological neurons.

A neural net consists of thousands or even millions of simple processing nodes that are densely interconnected. Most of today's neural nets are organized into layers of nodes, an input layer, one or more hidden layers and an output layer, organized in a so-called feed-forward manner, meaning that data moves through them in only one direction. An individual node might be connected to several nodes in the layer beneath it, from which it receives data, and several nodes in the following layer, to which it sends data. In Figure 1, one can see a simple representation of the structure of an ANN.

To each of its incoming connections, a node will assign a number known as a weight. When the network processes data, the node receives a different data item over each of its connections and multiplies it by the associated weight, and adds a bias value. It then adds the resulting products together, yielding a single number. This number goes through an activation function that returns the output number. If that number is below a threshold value, the node passes no data to the next layer. If the number exceeds the threshold value, the node fires, which usually means sending the sum of the weighted inputs and the bias along all its outgoing connections.

We can think of each individual node as its own linear regression model, composed of input data, weights, a bias, and an output. The model can be represented as follows, where *Wi* are the weights:

$$\sum WiXi + bias = w1x1 + w2x2 + w3x3 + bias \tag{1}$$

For the network to perform as wanted, it must be trained with a considerable amount of labeled data, often hand-labeled in advance. An object recognition system, for instance, might be fed thousands of labeled images of specific objects, and it would find visual patterns in the images that consistently correlate with particular labels. During training, the weights are continually adjusted until a desired level of convergence is reached. [17].

5.2 Activation Functions

Activation functions are specially used in ANNs to transform an input signal into an output signal of a node. We apply the activation function after calculating the sum of

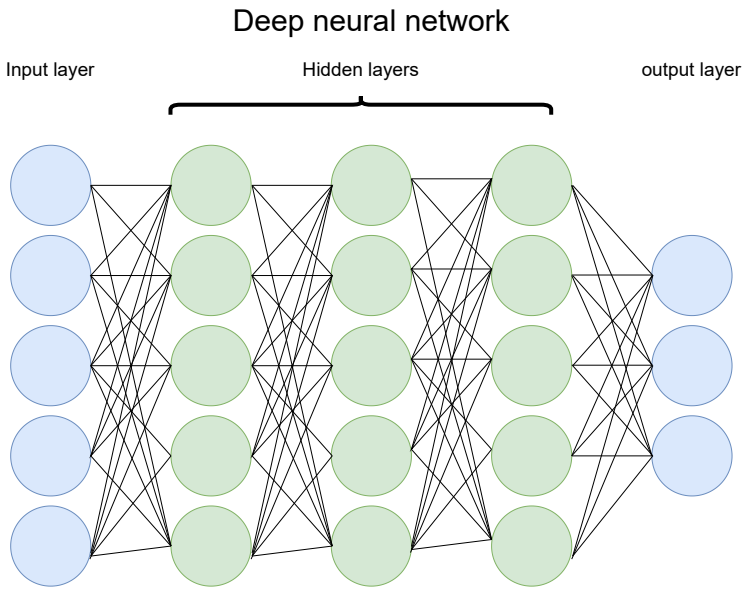

Figure 1: Representation of a Deep neural network.

products of inputs and their corresponding weights of a node. Its main purpose is to add non-linearity and to decide whether the node's input to the network is important or not in the process of prediction. An ANNs prediction accuracy depends on the number of layers used and on the type of activation function used.

If an activation function is not used in an ANN, no matter how many hidden layers are attached, the output signal would simply be a simple linear function, which is just a polynomial of degree one, because the composition of two linear functions is a linear function itself. Even though a linear equation is simple and easy to solve, their complexity is limited, the network can only adapt to the linear changes of the input. In the real world the errors possess non-linear characteristics, that is why without non-linearity ANNs do not have the ability to learn and recognize complex mappings from data [18].

There are many different activation functions, but the ones used in the thesis are the following:

Sigmoid This function takes any real value as input and outputs values in the range [0, 1]. As illustrated in Figure 2, the greater the input, the closer the output value is to 1.0, and the smaller the input, the closer the output is to 0.0. Down below, we can see the sigmoid function:

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{2}$$

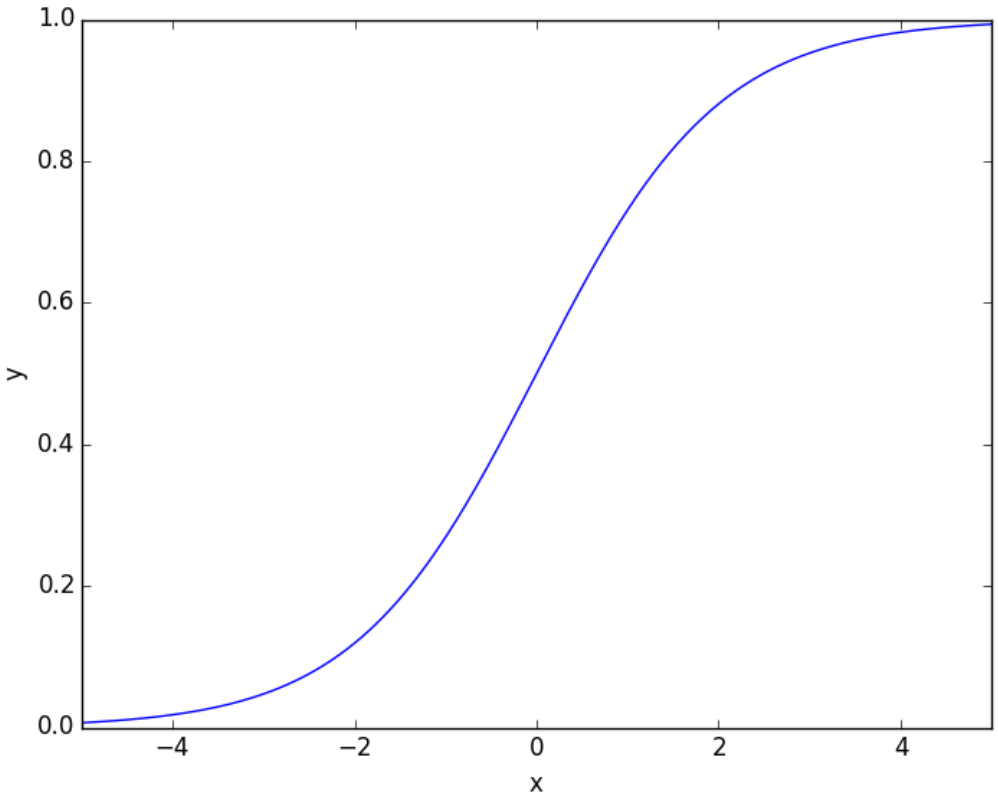

Figure 2: Sigmoid function.

This is one of the most used functions because many models work with probability as an output. Since probability of anything exists only between the range of 0 and 1, sigmoid is the right choice because of its range. Also, this function is differentiable and provides a smooth gradient, this prevents jumps in output values.

This function has the following limitations. Firstly, the sigmoid function is not symmetric around zero, which means that the signs of all output values of neurons will be the same. This gives inefficient gradient updates, because the gradients will always be either all positive or all negative. Another drawback is that saturated neurons can 'kill' the gradients, meaning that very high or very low values will result in either zero or one. Lastly, the function contains an exponential term, which is computationally expensive.

ReLU The Rectified Linear Unit (ReLU) returns zero if for negative inputs, and the value itself for positive inputs.

Even though the ReLU function looks like a linear function, it has a derivative function and allows for backpropagation while simultaneously making it computationally efficient. Down below, the ReLU expression is shown as well as its plot in Figure 3.

$$max(0, x) \tag{3}$$

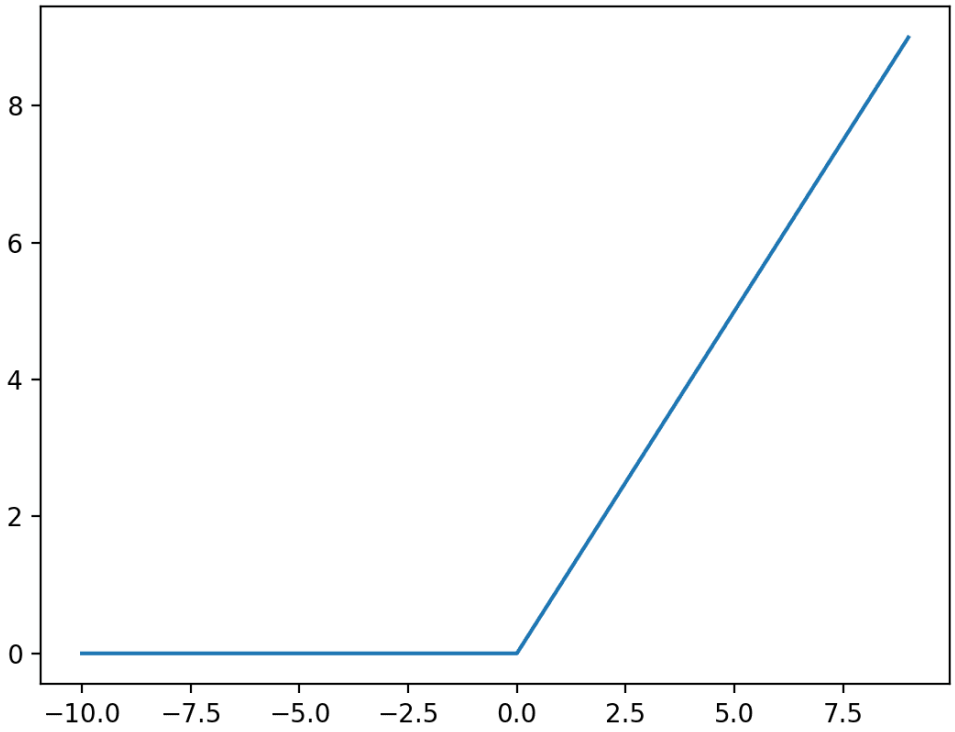

Figure 3: ReLU function.

This function is more computationally efficient than the sigmoid function. Also thanks to its linear, non-saturating property, it accelerates the convergence of gradient descent towards the global minimum of the loss function.

The limitations of this function are that during the backpropagation process, the weights, and biases for some neurons are not updated. This can create dead neurons which never get activated.

Softmax This function is a combination of multiple sigmoid. It is commonly used as an activation function for the last layer of a multi-class classification neural network, as it calculates the relative probabilities of each class. For instance, if we applied an input of (1.8, 0.9, 0.68) the output after the softmax function would be (0.58, 0.23, 0.19). The sum of all the classes probabilities is equal to one. Down below, the softmax equation can be seen, where Zi are the outputs of every class.

$$softmax(z_i) = \frac{e^{z_i}}{\sum_j e^{z_j}} \tag{4}$$

5.3 Training

Training an ANN means finding a valid setting for all the weights and bias so that the neural network can solve a specific problem. When a neural net is being trained, all of its weights and bias are initially set to random values. Training data is fed to the input layer, and it passes through the succeeding layers, getting multiplied and added together as described above, until it reaches the output layer. To evaluate the accuracy of the network, a cost function is used, also called loss function. There are several loss functions for different tasks. The Mean Squared Error (MSE) is the average of the sum of the squares of the differences between the obtained value and the ground truth value. So, the bigger the difference between the expected output and the obtained output, the larger the value of the cost function. Below the equation can be seen, where m is the number of predictions, y is the vector of observed values of the variable being predicted, with $\hat{Y}$ being the predicted values.

$$\mathbf{MSE} = \frac{1}{m}\sum_{i=1}^{m}(y_i - \hat{y}_i)^2 \tag{5}$$

Our goal is to minimize our cost function so that the network finds the right weights and bias for the given set of training data. Since the cost function of a neural network is too complex to determine the minimum explicitly, we use the gradient descent technique.

The gradient of a function gives the direction of the steepest ascent, so the negative gradient will give the direction to step that decreases the function most quickly. Also, the magnitude of the gradient of the cost function tells us what nudges of all the weights and biases cause the fastest change in the value of the cost function, or in other words, which changes to which weights matter the most. The negative gradient of the cost function can be represented as follows, where W represents all the weights and biases of the network put into a single column vector:

$$-\nabla C(\tilde{W}) \tag{6}$$

The gradient descent is the process of repeatedly nudging an input of a function by some multiple of the negative gradient, and it is a way to converge towards some local minimum of the cost function. The algorithm for computing this gradient efficiently is called backpropagation.

The essence of backpropagation is to compute the partial derivative of the residuals for each parameter, by starting from the calculated loss and then propagating back through the network, and learn an association rule between the residuals and the network weights. Then, adjust the weights of the network to make the network output closer to the given expected value. A simple example with one neuron can be seen in Figure 4, where the incoming gradient from the connection z between two neurons is propagated back through the neuron to the connections x and y by applying the chain

rule. This will then give gradients for each of the parameters in the network, and the weights can be updated using an optimization technique.

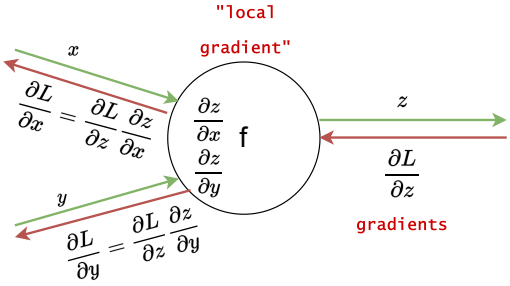

Figure 4: An illustrative example of backpropagation through a node, where L is the calculated loss.

There are different approaches for the optimization of the network weights based on the gradient calculated using backpropagation. Stochastic gradient descent is a technique that updates the weights and biases in the direction of the gradient with a fixed step size called learning rate, that needs to be specified in advance. To converge to an acceptable local minimum faster, a momentum term can be introduced that can pass a saddle point at shallow local minima. A popular optimizer is the Adam optimizer [19] that uses the momentum technique and an adaptive learning rate for every parameter instead of adjusting the learning rate. In all the different types of optimizers, a clipnorm can be introduced in order to set a maximum norm of the gradients to a specific value. Below, we can see how the weight W_i is updated by equation 7, and the bias b_i is updated by equation 8, where $E(W, b)$ is the loss function and η is the learning rate.

$$W_i = W_i - \eta \frac{\partial E(W, b)}{\partial W_i} \tag{7}$$

$$b_i = b_i - \eta \frac{\partial E(W, b)}{\partial b_i} \tag{8}$$

Before a model is trained, there are many parameters that should be taken into account. The Hyperparameters are tunable parameters specified by the developer that can directly affect how well the model trains.

The number of epochs is the number of times that all the training samples get a forward and backward pass, in other words, a full training cycle. This is considered one of the most relevant parameters in training. Increasing the number of epochs

will increase accuracy and reduce loss, but if it is too high, the validation loss on the validation data will start going up again due to overfitting. This happens because the model learns the training dataset too well, including the statistical noise or random fluctuations, and it is no longer able to predict correctly outside the training data. To properly set the number of epochs, it helps to analyze the learning curve, which is a representation of the training and validation loss over the epochs. The learning curve will show whether the given model is underfitted, a correct fit to the training dataset, or overfitted.

When a model is underfitted, the learning curve of the training loss may show noisy values or a flat line of relatively high loss, as can be seen in Figure 5. This means that the model was not able to learn from the training data very well.

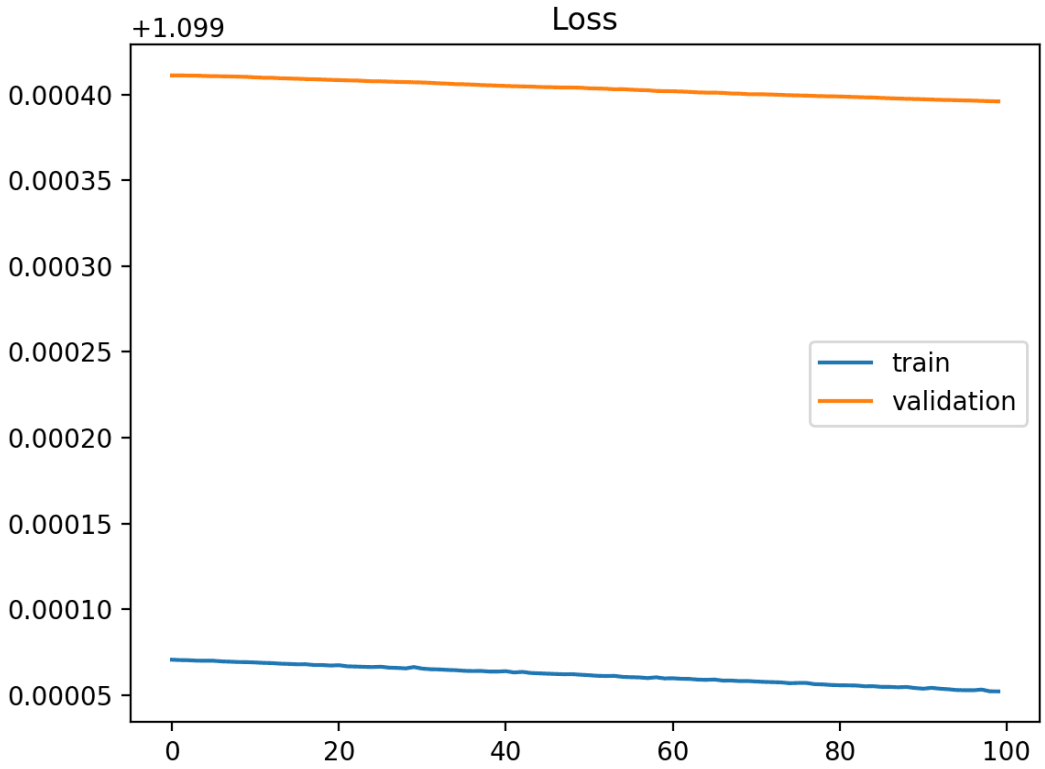

Figure 5: Underfitted learnig curve example.

An underfitted or overfitted model can also be identified by a decreasing training loss that continues to decrease at the end of the training process, as can be seen in Figure 6. This indicates that either the training was stopped prematurely or too late.

If the model is trained for too many epochs, it can overfit. When this happens, the training loss never stops decreasing while the validation loss decreases to a point, and it starts to increase again, as can be seen in Figure 7.

If the training parameters of a model are set correctly, and the model learns correctly,

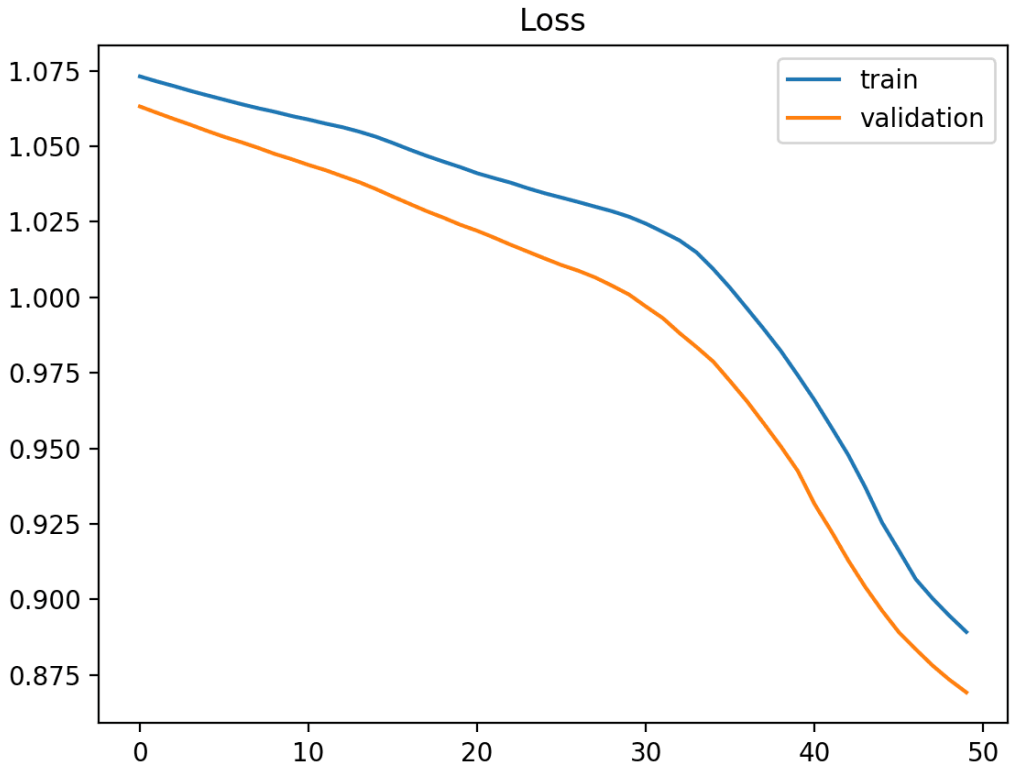

Figure 6: Underfitted or overfitted learnig curve example.

we should see in the learning curve that both the training and validation loss decrease to a point of stability with minimal gap between the two final loss values, as can be seen in Figure 8.

The batch size is the number of training samples used in every forward and backward pass. It can change the accuracy of the estimate of the error gradient when training. There is a relation between batch size and the speed and stability of the learning process. Generally, a small batch size will result in more iterations and thus in more backpropagations. Larger batch sizes results in less backpropagations and more time consuming iterations, as can be seen in Figure 9. This tends to slow down the learning process, but have lower variance in classification accuracy in the final stages of the training.

The number of iterations is the number of passes, each pass using a batch of samples. The total number of iterations per every epoch is the number of samples divided by the batch size.

One of the most important hyperparameters is the learning rate. This hyperparameter controls how much the model should be changed according to the estimated error each time the model weights are updated. The learning rate can also be referred to as the step size of the training, because it is the amount of weights that are updated every

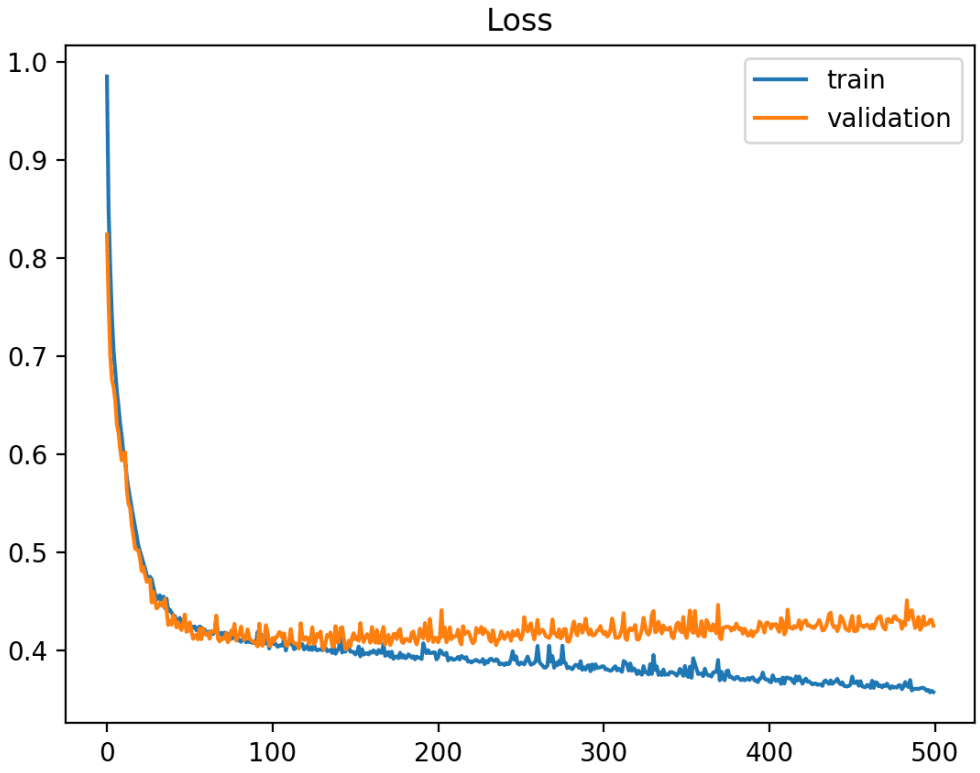

Figure 7: Example of train and validation learning curves showing an overfit model.

time. If small learning rates are used, the training requires more epoch given the smaller changes made to the weights in each update, whereas using larger learning rates results in faster changes and requires fewer training epochs. If the learning rate is too small, the training process could be too long and get stuck, while if the learning rate is too big may result in learning a suboptimal set of weights too fast or a training process that is unstable. In Figure 10 we can see some examples of this behavior.

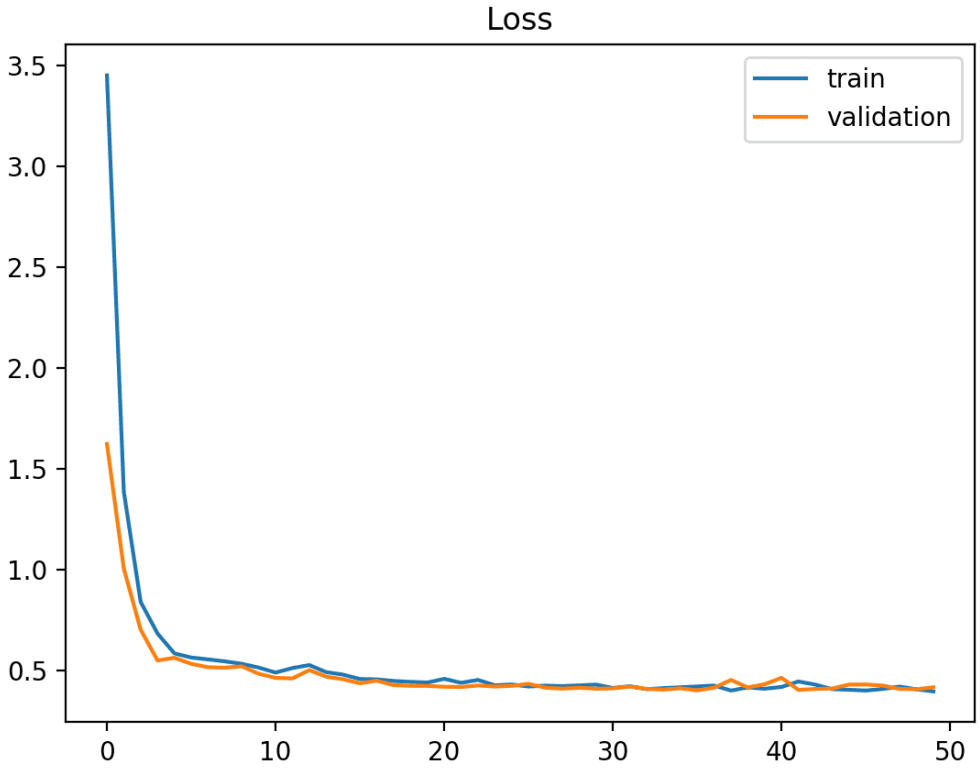

Figure 8: Goodfitted learningcurve example.

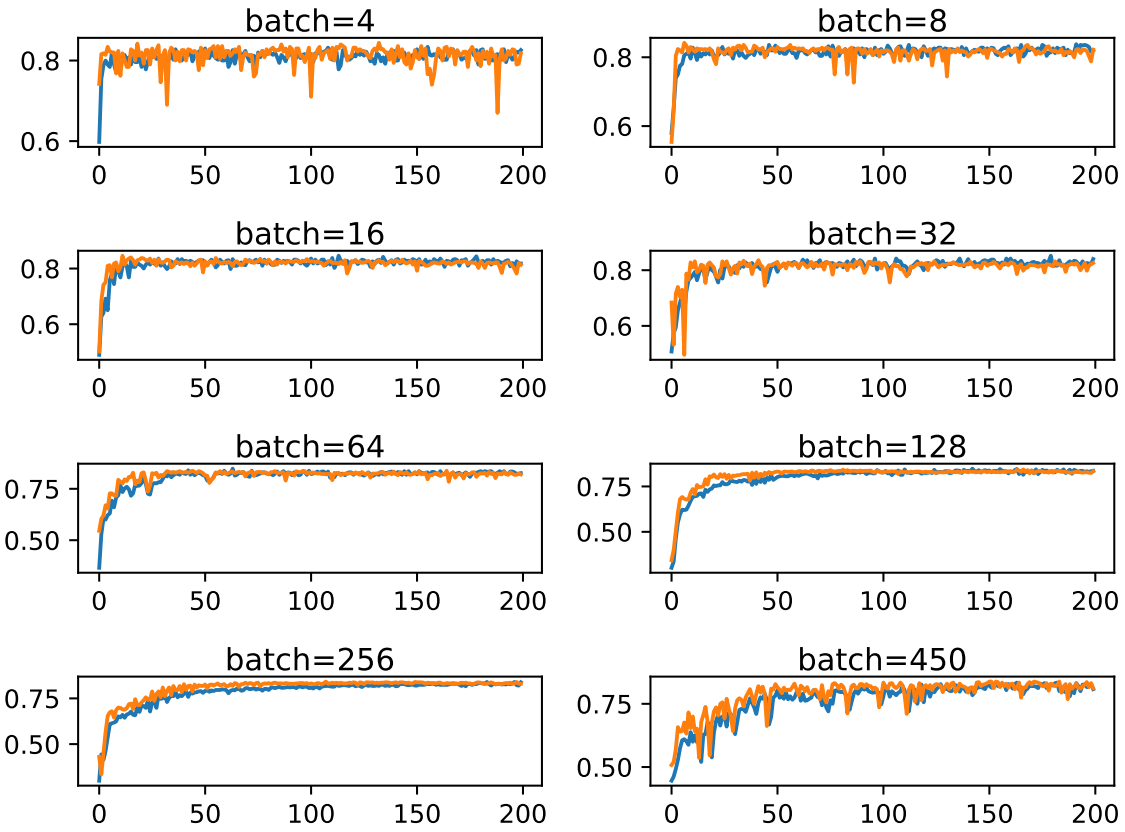

Figure 9: Line plots of classification accuracy on train and test Datasets with different batch sizes example.

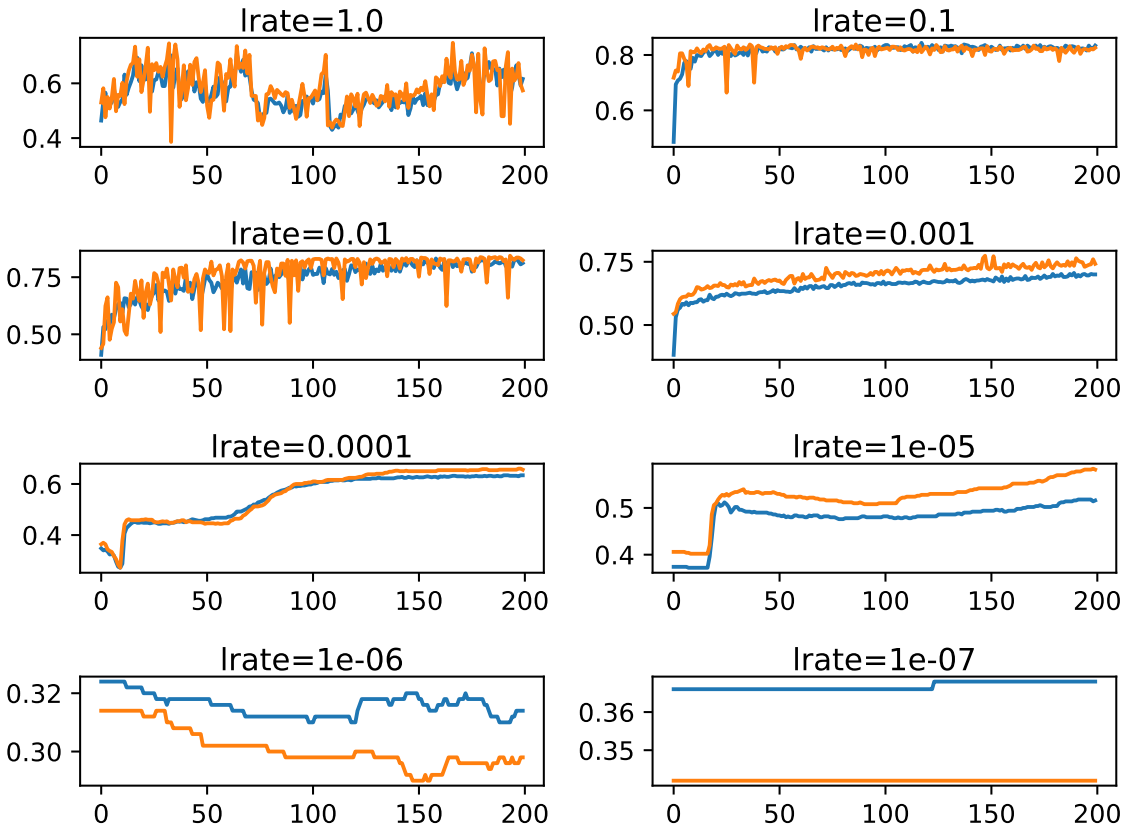

Figure 10: Line Plots of Train and Test Accuracy for a Suite of Learning Rates on a Classification Problem.

5.4 Deep Convolutional Neural Networks

CNNs are a type of ANN that perform very well with images, thanks to their superior feature extraction abilities. CNNs structural design also offers an end-to-end learning model with gradient descent-trained parameters.

The difference between a feedforward neural network and a CNN is the architecture. CNN enhances the idea of receptive field and shared weights [20], which not only reduces the complexity of the network model, but also greatly reduces the parameters of training.

CNNs layers are organized in three dimensions: width, height and depth and the neurons in one layer do not connect to all neurons in the next layer, only a small part of it. After processing, the output of a layer is reduced to a single vector of probability scores, organized along the depth dimension. These characteristics make the CNN more suitable for the learning and representation of image features, and it can also keep the translation and scale invariance to a certain extent. Typically, CNNs are made with the following kinds of layers:

Convolutional layer The convolutional layer, or CONV, is the core building block of a CNN that does most of the computational processing. Convolution is a mathematical term and refers to a combination of two functions (f and g) to produce a third function ($f * g$), merging two sets of information [21]. The convolution symbol $*$, it is defined as the integral of the product of the two functions (f and g) after one is reflected about the $y - axis$ and shifted. As shown in the equation 9, it is a particular kind of integral transform.

$$(f * g)(t) := \int_{-\infty}^{\infty} f(\tau)g(t-\tau)d\tau \tag{9}$$

For every t, the convolution can be thought of as the area under the function $f(\tau)$ weighted by the function $g(-\tau)$ modified by the amount t. As t shifts, the weighting function $g(t-\tau)$ emphasizes different parts of the input function $f(\tau)$. If t is a positive value, then $g(t-\tau)$ is equal to $g(-\tau)$ that slides or is changed along the $\tau - axis$ toward the right ($+\infty$) by t, while if t is a negative value, then $g(t-\tau)$ is equal to $g(-\tau)$ that is shifted toward the left ($-\infty$) by the amount of $|t|$.

The CONV layer's parameters are a set of learnable filters. Every filter is small along width and height, but extends through the entire depth of the input volume. For instance, a typical filter on a first layer of a CNN might have a size of 5 x 5 x 3, 5 pixels width and height, and 3 because images have depth 3, the color channels. During the forward pass, we slide (more precisely, convolve) each filter across the width and height of the input volume and compute dot products between the entries of the filter and the input at any position. As we slide the filter over the width and height of the input volume, we will generate a two-dimensional activation map containing the filter's outputs at each spatial position. On the first layer, the network will learn filters that activate visual features, such as an edge of some orientation or a blotch of some color, and eventually whole shape patterns on higher layers of the network. Each CONV layer will now have a full set of filters, each of which will generate a separate two-dimensional

activation map. The output volume will be produced by stacking these activation maps along the depth dimension.

To visualize this procedure, we will represent this two dimensional simplified example. Think of a 5 × 5 matrix as the input image, and a 3 × 3 filter or kernel, that only contain values of either zero or one, as represented in Figure 11.

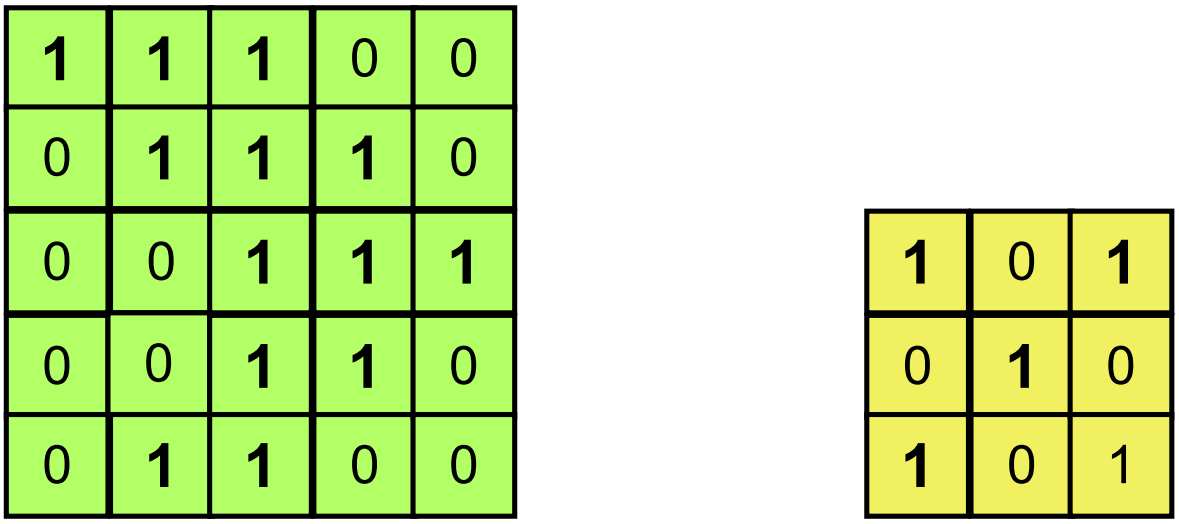

Figure 11: Simple 5 x 5 matrix and 3 x 3 filter representation.

Now, when we slide the 3 × 3 filter over the 5 × 5 matrix, starting from the top-left corner in Figure 12 one can see the acquired result:

The yellow square is the part of the input matrix that is processed by the filter. Filters act as feature detectors from the original input image. The filter is represented in light red.

Pooling layer The pooling is usually added after a convolutional layer. This layer has two functions. Firstly, reducing the dimensions of the feature map to reduce processing time, and secondly, to avoid overfitting.

There are several pooling layer options, max-pooling, average-pooling, stochastic pooling, overlapped pooling etc. Max pooling is the most popular. As shown in Figure 13, max pooling takes the maximum value in each window.

This decreases the feature map size while keeping the significant information [22].

Fully Connected Layer This is the last layer, and it will compute the class scores and return a number associated with a classification class. For instance, in a classification task, zero could mean cat and one could mean dog. As with ordinary ANNs and as the name implies, each neuron in this layer will be connected to all the nodes in the previous volume.

Architecture The first few layers are usually alternating layers of convolution and pooling, and the final layers of the network near the output layer are usually fully

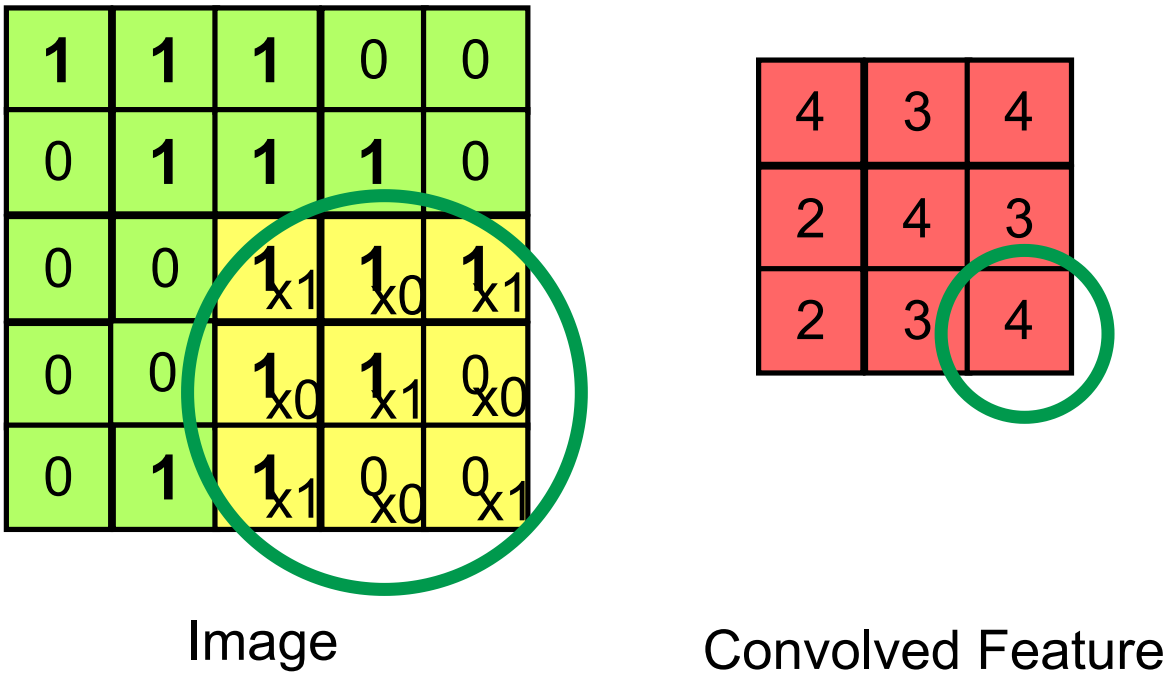

Figure 12: Simple convolution representation.

connected network [23]. In Figure 14 one can see a simple representation of a typical CNN structure. Training a CNN is done using backpropagation or gradient descent. Convolution and pooling layers act as feature extractors and fully connected layers as a classifier. CONV and Fully-connected layers perform transformations in the weights and biases of the neurons. Whereas, the RELU and Pooling layers will implement a fixed function.

5.5 Transfer Learning and Fine Tuning

When image classification and object detection tasks are similar to previously performed tasks, we can use an already made network. One option is to modify the last layers to adapt the model to our own objective, and then train it. We can also use the trained weights used in previous models and only train the last layers of the classier to adjust the old model to detect new objects. Existent high performance models for image classification are very good at capturing the basic essence of an image, for example, lines, edges, or shapes. Models trained on the dataset ImageNet are common to extract the weights from and are used for transfer learning, since ImageNet contains over 14 millions annotated images for image classifications in over 20,000 classes [24]. Transfer learning is often used for computer vision, since features in images are often similar, regardless of the object being recognized. Transfer learning can also reduce the training time of a network if the pretrained weights only need little modifications [25].

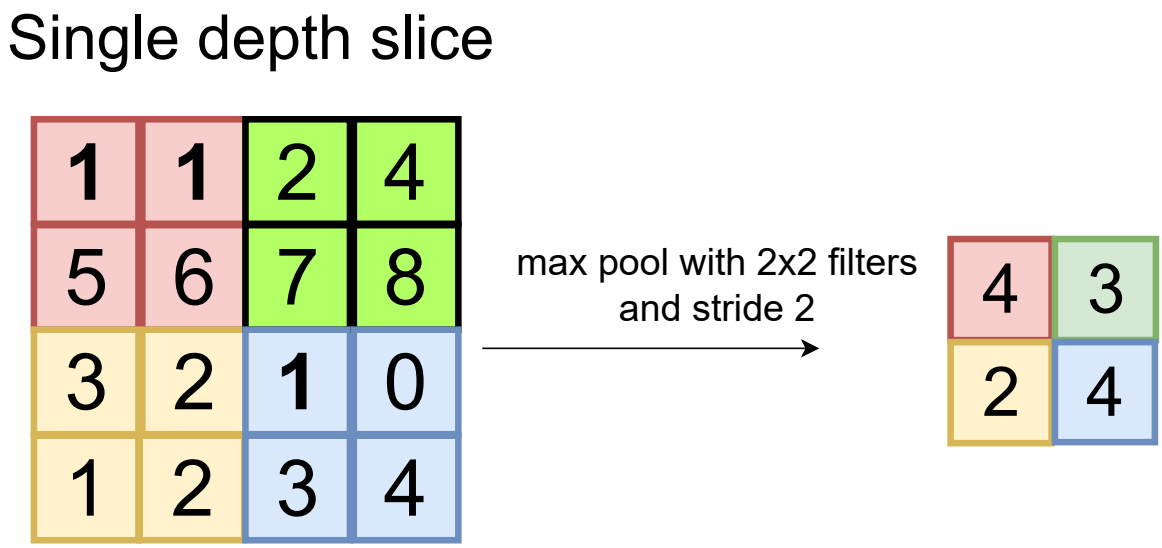

Figure 13: Max pooling example using 2x2 filters and stride 2.

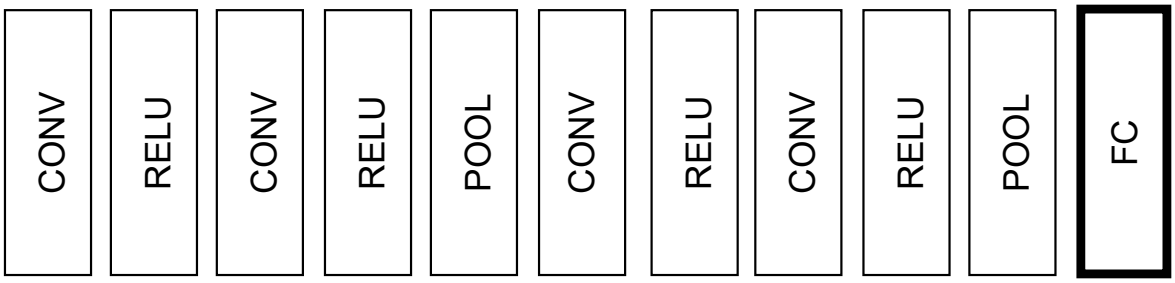

Figure 14: Simple representation of a typical CNN structure.

6 Materials and Methods

The development of the method will be separated in two stages. Firstly, an object detection model will be developed and trained for the detection and classification of laparoscopic instruments. And secondly, the first model will be modified and trained so that it can not only detect and classify laparoscopic instruments, but also rotate and reshape the bounding boxes to perfectly adjust to the object.

6.1 Object Detection

In computer vision, object detection is the technique that allows the computer to detect a certain object and its position within an image and put it in a bounding box for following classification. This method can also be applied to a video, as it is a sequence of images displayed in rapid succession.

Nowadays, there is a huge amount of deep learning based object detection methods. The most widely used methods include Faster R-CNN, You Only Look Once (YOLO) and SSD. However, we have focused our implementation to the latest algorithms, which are faster, more accurate and single-shot or single-pass detectors, specifically SSD.

Single Shot MultiBox Detector (SSD) SSD, is the method that adjusts best to this thesis, because even though other methods such as YOLO, would process faster, SSD is far more accurate, and it is still fast enough to run on video data, so it is more suited for laparoscopic image application. SSD, as the name states, requires only one single shot for the detection of numerous objects in an image. This means that the convolutional network will only run once and predict the location and class for each object. This makes SSD much faster than methods based on two-shot regional proposal networks (RPN). Experimental results on the COCO, PASCAL VOC and ILSVRC datasets confirm that SSD has higher accuracy and is much faster than two-stage methods that utilize an additional object proposal step, while providing a unified framework for both training and inference. This method recognizes objects in images by discretizing the output space of bounding boxes into a set of default boxes at various aspect ratios and scales per feature map position. The object detector assigns scores to the presence of each object class in each default box and reshapes the box to fit the shape of the object better. In addition, to handle objects of changing sizes, the network uses various feature map predictions with varying resolutions. The early network layers or the base of the network, are based on a standard architecture used for high performance image classification, and then an auxiliary structure to produce detections with the following key features:

Multi-scale feature maps for detection. After the base of the network, convolutional feature layers are added. These layers decrease in size progressively and allow predictions of detections at multiple scales.

Convolutional predictors for detection. Using a set of convolutional filters, each added feature layer can produce a fixed set of detection predictions. For a feature layer of size m $\times$ n with p channels, the basic element for predicting parameters of a potential detection is a 3 $\times$ 3 $\times$ p small kernel that produces either a score for a category, or a shape offset relative to the default box coordinates. At each of the m $\times$ n locations where the kernel is applied, it produces an output value. The bounding box offset output values are measured relative to a default box position relative to each feature map location.

Aspect ratios and default boxes. We associate a set of default bounding boxes with each feature map cell, for multiple feature maps at the top of the network. The default boxes tile the feature map in a convolutional manner, so that the position of each box relative to its corresponding cell is fixed. At each feature map cell, we predict the offsets relative to the default box shapes in the cell, as well as the per-class scores that indicate the presence of a class instance in every box. Specifically, for each box out of k at a given location, we compute c class scores and the 4 offsets relative to the original default box shape. This results in a total of (c + 4)k filters that are applied around each location in the feature map, yielding (c + 4)kmn outputs for an m $\times$ n feature map. Our default boxes are similar to the anchor boxes used in Faster R-CNN [26], however we apply them to several feature maps of different resolutions. Allowing different default box shapes in many feature maps let us efficiently discretize the space of possible output box shapes. In Figure 15 one can see a representation of the SSD structure.

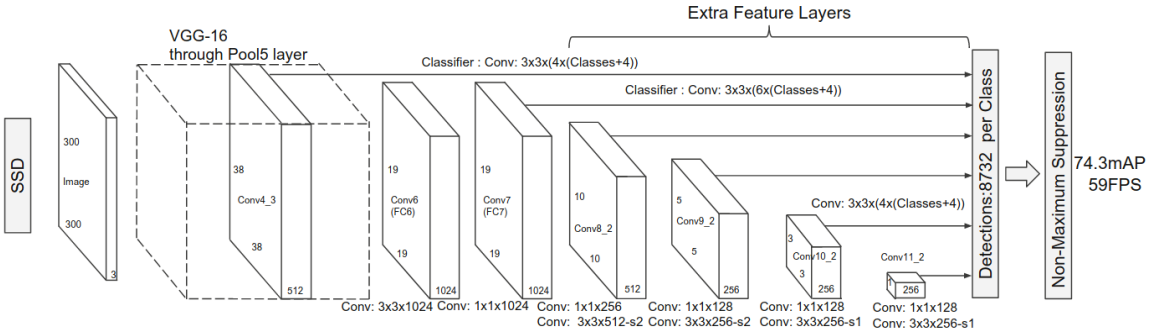

Figure 15: Representation of the SSD architecture.

The SSD is simple to train and to integrate into software systems that require object detection. In SSD training, ground truth information needs to be assigned to specific outputs in the fixed set of detector outputs. Some version of this is also required for training in YOLO [4] and for the region proposal stage of Faster R-CNN [26] and MultiBox [27]. Once this assignment is determined, the loss function and back propagation are applied from end to end. Training also involves choosing the set of default boxes and scales for detection, as well as the hard negative mining and data

augmentation strategies.

Hard negative mining is the method needed to counter the imbalance between the positive and the negative training examples. This imbalance is introduced by the large amount of negative default boxes after the feature matching step. Instead of using all the negative examples, they are sorted using the highest confidence loss for each default box and the top ones are picked so that the ratio between the negatives and positives is at most 3:1. This leads to faster optimization and a more stable training [28].

We need to train the network according to which default boxes correspond to a ground truth detection. For each ground truth box we are selecting from default boxes that vary over location, aspect ratio, and scale. We begin by matching each ground truth box to the default box with the best Jaccard overlap. Then we match default boxes to any ground truth with Jaccard overlap higher than a threshold of 0.5. This simplifies the learning problem, allowing the network to predict high scores for multiple overlapping default boxes, rather than requiring it to pick only the one with maximum overlap.

The objective of the SSD training is similar to the MultiBox objective [[27],[29]] but is extended to handle multiple object categories. The overall objective loss function, which can be seen below, is a weighted sum of the localization loss (L_{loc}) and the confidence loss where N is the number of matched default boxes, and the localization loss is the Smooth L1 loss [30] between the predicted box (l) and the ground truth box (g) parameters. Similar to Faster R-CNN [26], we regress the offsets for the center of the bounding box and for its width and height. Our confidence loss is the softmax loss over multiple class confidences (c) and the weight term α is set to one.

$$L(x, c, l, g) = \frac{1}{N}(L_{conf}(x, c) + \alpha L_{loc}(x, l, g)) \tag{10}$$

To develop the object detection model for this thesis, we will use transfer learning. We will modify a pretrained model called SSD300 and adapt it for the detection and classification of laparoscopic instruments in 18 different categories and then train it, updating all weights and biases. The final implemented model has a pretrained VGG-16 architecture in the base network, consisting of the 20 layers that can be seen in Figure 16.

The dataset used to train this model is given as two JSON files. One with 1499 image paths and another one with the class type and the coordinates of every bounding box. The class types are given in the form of index numbers. These can be translated to the actual class name. The bounding box coordinates are given in the COCO format [31], upper left corner x and y coordinates and height and width of the bounding box.

6.2 Evaluation

To evaluate the training and the performance of our models, we will use the following indexes:

We use tensorboard to plot every loss, including the location loss, the confidence loss and the total loss of both training and validation.

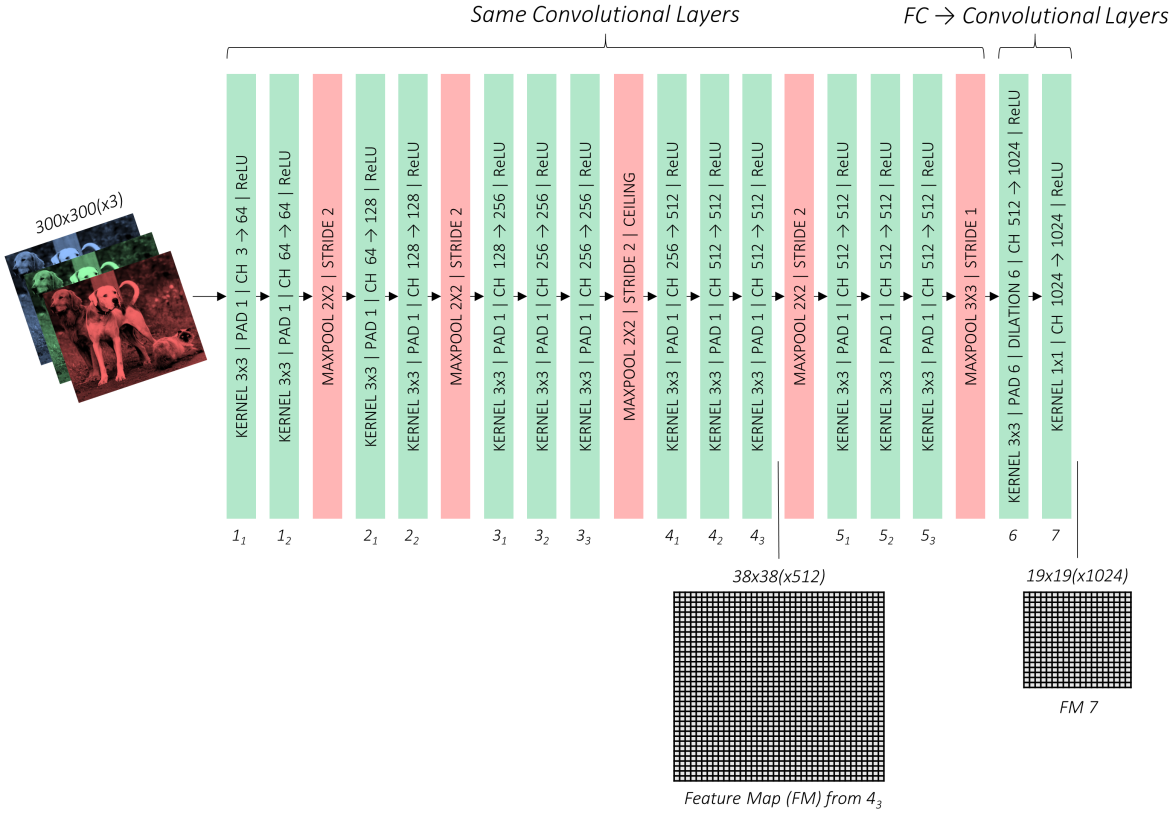

Figure 16: Implemented model layers.

We will also calculate the mean Average Precision or mAP. The mAP is a metric used to evaluate the performance of object detection models. It measures the average precision of a model across multiple object classes. The precision is the ratio of true positive detections to the total number of positive detections (true positives and false positives). The average precision is calculated for each class separately, and then the mean of these values is taken. A high mAP value indicates that the model is able to accurately locate and classify objects in the image. In formula 11 we can see the mAP expression where AP_i is the average precision of every class.

$$mAP = \frac{1}{N} \sum_{i=1}^{N} AP_i \tag{11}$$

To evaluate how well the bounding box fits the object, we calculate the average Jaccard index, also known as the Jaccard similarity coefficient, between the bounding boxes and the actual object area. We calculate the object area from the segmentation data in the dataset. The higher the Jaccard index, the less empty area is contained inside the bounding box, or in other words, the bounding box fits the object better. The Jaccard Index is a measure of the similarity between two sets of data. The index ranges from 0 to 1. The closer to 1, the more similar the two sets of data. The Jaccard similarity is calculated, dividing the number of observations in both sets between the number in either set. The Jaccard index expression can be seen in formula 12, where A

and B are the datasets.

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|} \tag{12}$$

6.3 Rotating Bounding Box

As stated earlier, the aim of the thesis is to develop a method that can not only detect laparoscopic instruments, but also orient the bounding boxes to match the instrument angle. To address the development of rotating bounding boxes, we will implement the following approach. We will add a rotation angle in the dataset as a new learnable parameter, so that the network learns how to predict not only an object's location and category but also its rotation angle.

For this approach, we will add a rotation angle parameter in the dataset and modify the model, so it can also predict angle rotation. However, If we only add the rotation angle, the neural network will learn to predict the proper rotation to the bounding boxes, but the boxes will not fit the object once rotated, as represented in Figure 17. That is because the boxes will be shaped to fix the object before rotation.

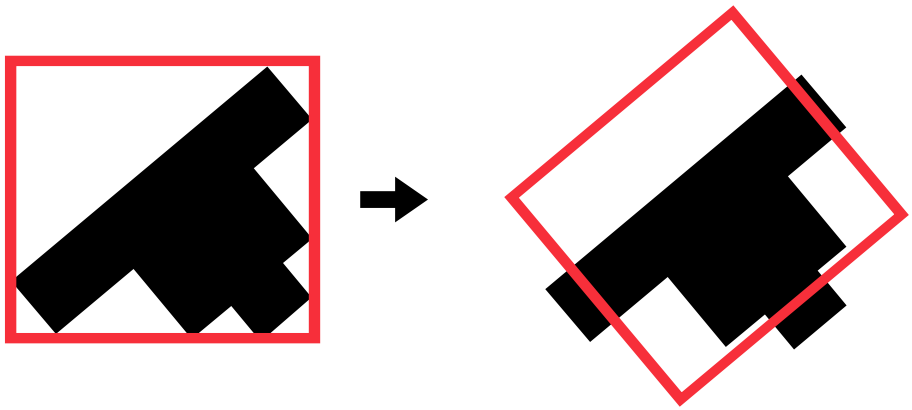

Figure 17: Example of a rotated box that does not fit the object.

To avoid this, we have to train the model with the rotated bounding box coordinates instead of the original ones. The model should learn to predict both the rotation angle and the bounding box shape that adjusts best to the rotated object, as represented in Figure 18.

Dataset The dataset used to train this model consists of one JSON file with 1499 image paths and another one that required some preprocessing. The not rotated bounding boxes coordinates, with their class type and a segmentation dot map that outlines the detected objects were taken from the original data file. From the segmentation point map, we are able to find the minimum bounding box and then calculate the rotation angle. The minimum bounding box is calculated with a function that gets the convex hull of all the segmentation dots and then fits a box around it. To calculate the angle

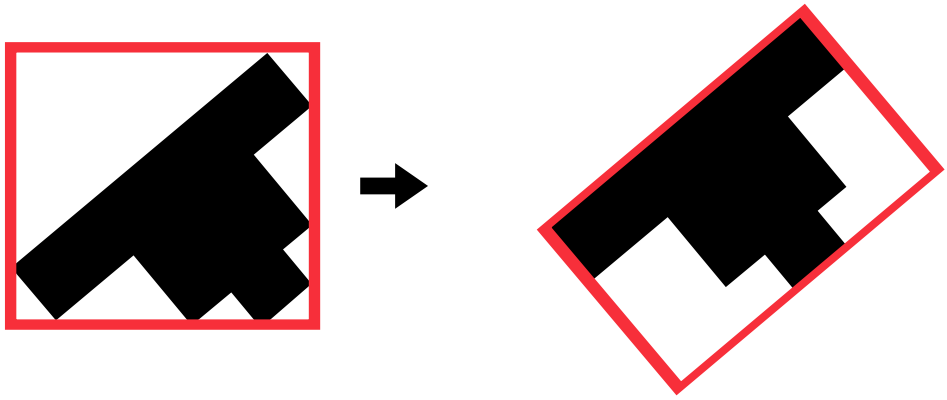

Figure 18: Example of a rotated box that properly fits the object.

of the rotated bounding box, we calculate the vector of the longest side of the rotated bounding box. Then calculate the angle between this vector and the horizontal axis defined by the vector (1, 0). Below, one can see the equation to calculate the angle between two vectors, a and b.

$$\Theta = \cos^{-1}[\frac{ab}{|a||b|}] \tag{13}$$

The processed JSON data file contains: the rotated bounding box coordinates, given in the COCO format [32], the rotation angle in rad, the object class and segmentation area for later evaluation.

Evaluation For the evaluation of this model, we will use the same indexes used in the upright object detection model, with the addition of the angle loss plot in tensorboard.

7 Experiments and Results

7.1 Object detection with axis aligned boxes

The first implementation was the detection and classification of laparoscopic instruments, without rotated bounding boxes. As we mentioned earlier, we used a modified SSD300 pre-trained model. Since SSD300 has a very specific structure, there are only a few model parameters to take into account. We used the following parameters:

Number of classes: 18 different class objects, including many laparoscopic instruments and their different parts.

Epochs: 200 forward and backward passes of all training samples.

Batch size: 8 training samples in every forward and backward pass.

Iterations: 120000 passes of 8 samples.

lr: A learning rate of 1e-3.

Results For the previous parameters, the training reached the following results. It achieved a final mean average precision of 0.77 in the training and 0.145 in the validation. The losses reached the following values: A train loss of 1.504, a validation loss of 4.085, a confidence loss of 1.159 in training and 3.440 in validation and a location loss of 0.345 in training and 0.644 in validation. In Figure 19 we can see the learning curve with the validation and training loss.

After training, we evaluated the model. We calculated the average Jaccard index of a certain batch of images to evaluate how well the bounding box fits the object. The calculated Jaccard index for the axis aligned bounding boxes was 0.587.

In Figures 20 and 21 we can see the model prediction results with axis aligned boxes.

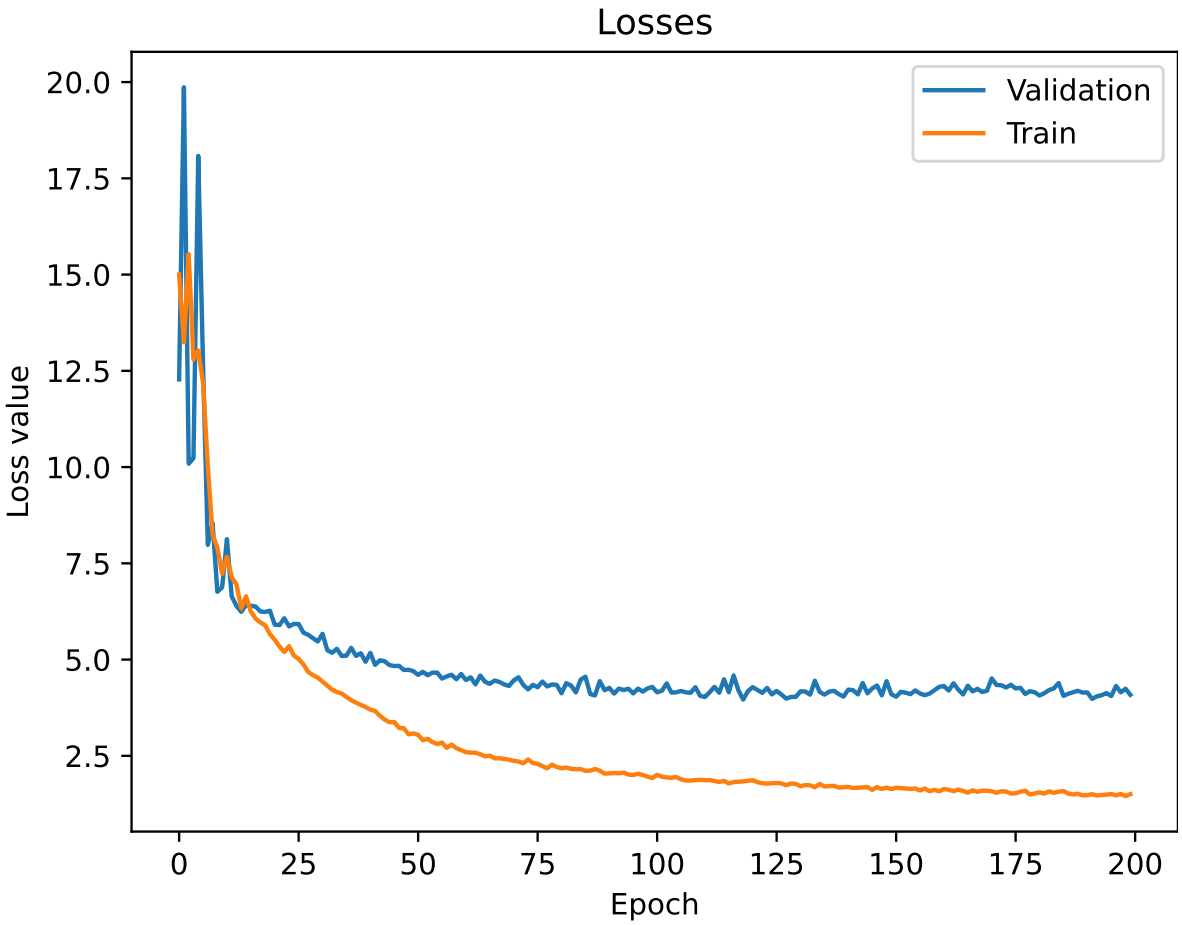

Figure 19: Learning curve with train and validation loss for axis aligned object detection experiment.

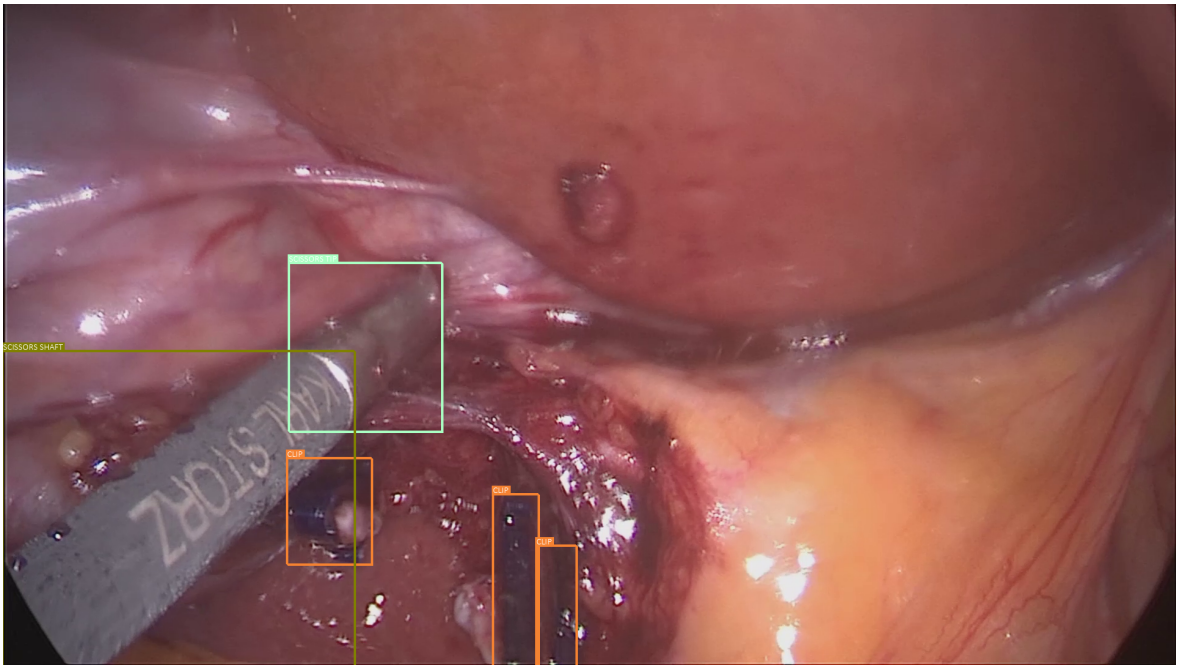

Figure 20: Prediction results with axis aligned boxes.

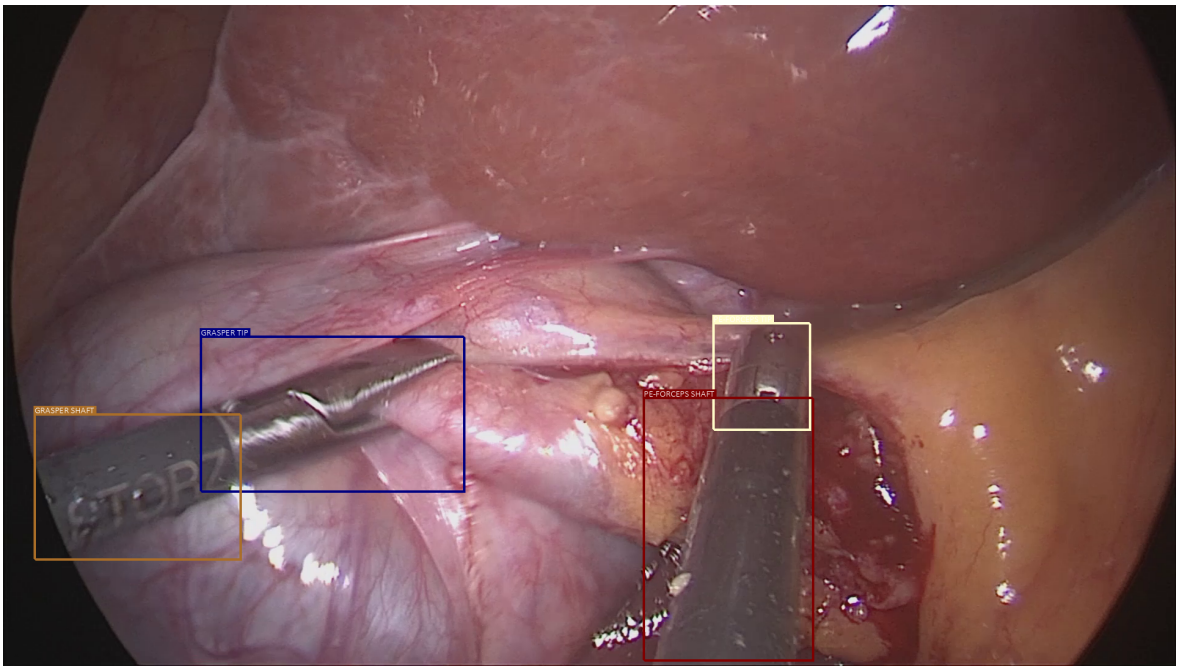

Figure 21: Prediction results with axis aligned boxes.

7.2 Object detection with rotating bounding boxes

For the previous parameters, the training reached the following results. It achieved a final mean average precision of 0.77 in the training and 0.145 in the validation.

The losses reached the following values: A train loss of 1.504, a validation loss of 4.085, a confidence loss of 1.159 in training and 3.440 in validation and a location loss of 0.345 in training and 0.644 in validation. In Figure 19 the learning curve with validation and training loss can be seen.

After training, we evaluated the model. We calculated the average Jaccard index of a certain batch of images to evaluate how well the bounding box fits the object. The calculated Jaccard index for the axis aligned bounding boxes was 0.587.

In Figure 20 and 21 we can see the model prediction results with axis aligned boxes.

As stated earlier, we added the rotation angle and the minimum bounding box shape as learnable parameters of the model. The model should learn to predict both the rotation angle and the bounding box shape that adjusts best to the rotated object.

We trained the same model using different parameters in each experiment in order to find the best performance. By varying the model's parameters, we aim to determine the optimal settings for the model that will result in the highest mean average precision (mAP) on the validation data. This approach allows us to evaluate the impact of different parameters on the model's performance and to identify the configuration that yields the best results. The results of this analysis will provide insights into the optimal design of the model and inform future developments in the field of object detection.

7.2.1 Experiment 1

For the first experiment, we used the following parameters:

Number of classes: 18 different class objects, including many laparoscopic instruments and their different parts.

Epochs: 200 forward and backward passes of all training samples.

Batch size: 18 training samples in every forward and backward pass.

Iterations: 120000 passes of 8 samples.

lr: A learning rate of 1e-3.

To give more importance to the new parameter, the angle loss, we will double it as shown in loss function 14.

$$Loss = Loss_{conf} + Loss_{loc} + 2 * Loss_{angle} \tag{14}$$

Results The training reached the following results. It achieved a final mean average precision of 0.239 in the training and 0.045 in the validation, as shown in Figure 22. The losses reached the following values: A train loss of 4.476, a validation loss of 8.819, a confidence loss of 2.075 in training and 4.9404 in validation and a location loss of 0.623 in training and 1.053 in validation. And finally, an angle loss of 0.888 in training and 1.412 in validation.

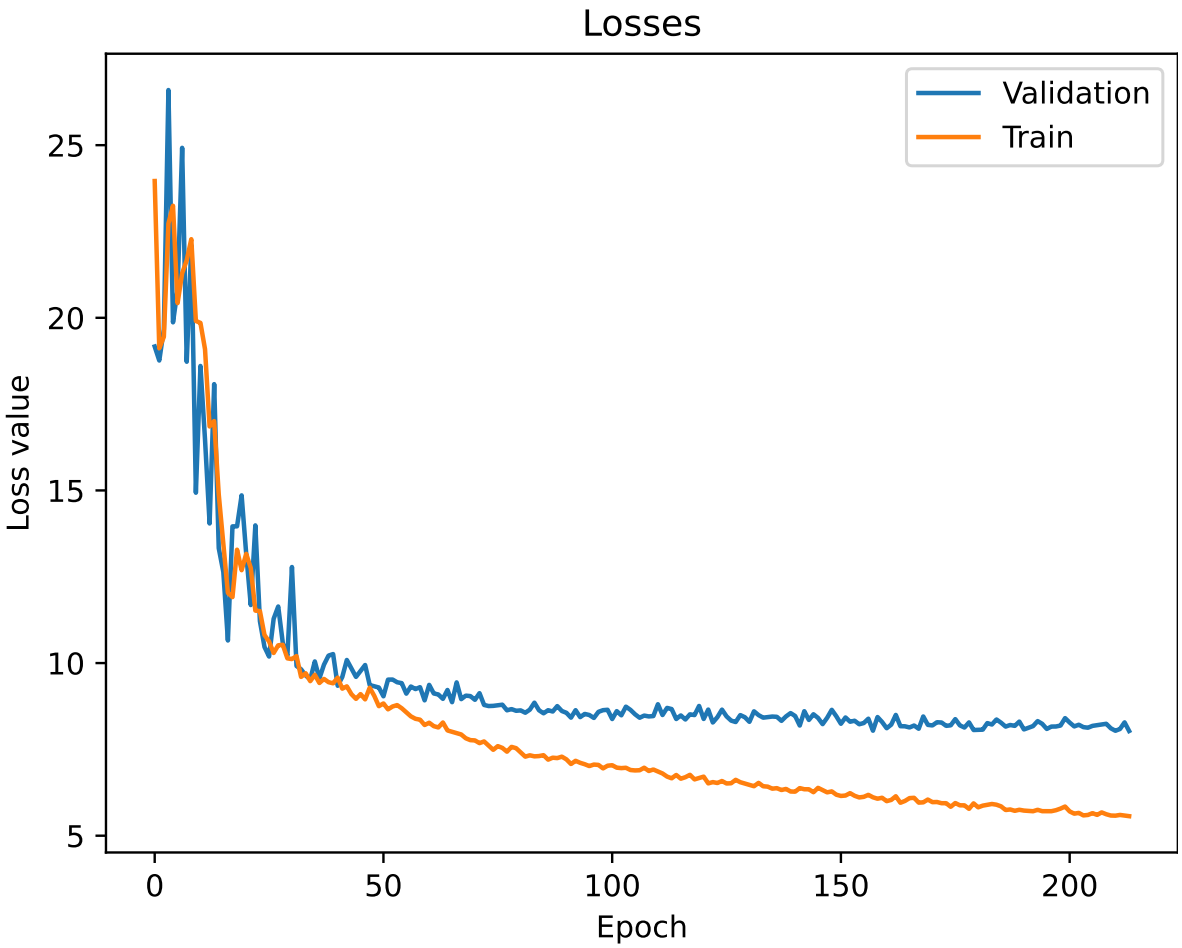

Figure 22: Learning curve with train and validation loss for rotated bounding boxes in experiment 1.

After training, we calculated the average Jaccard index of a certain batch of images to evaluate how well the bounding box fits the object. The calculated Jaccard index for the rotated bounding boxes for this experiment was 0.89.

In Figure 23 the model prediction results for this experiment can be seen.

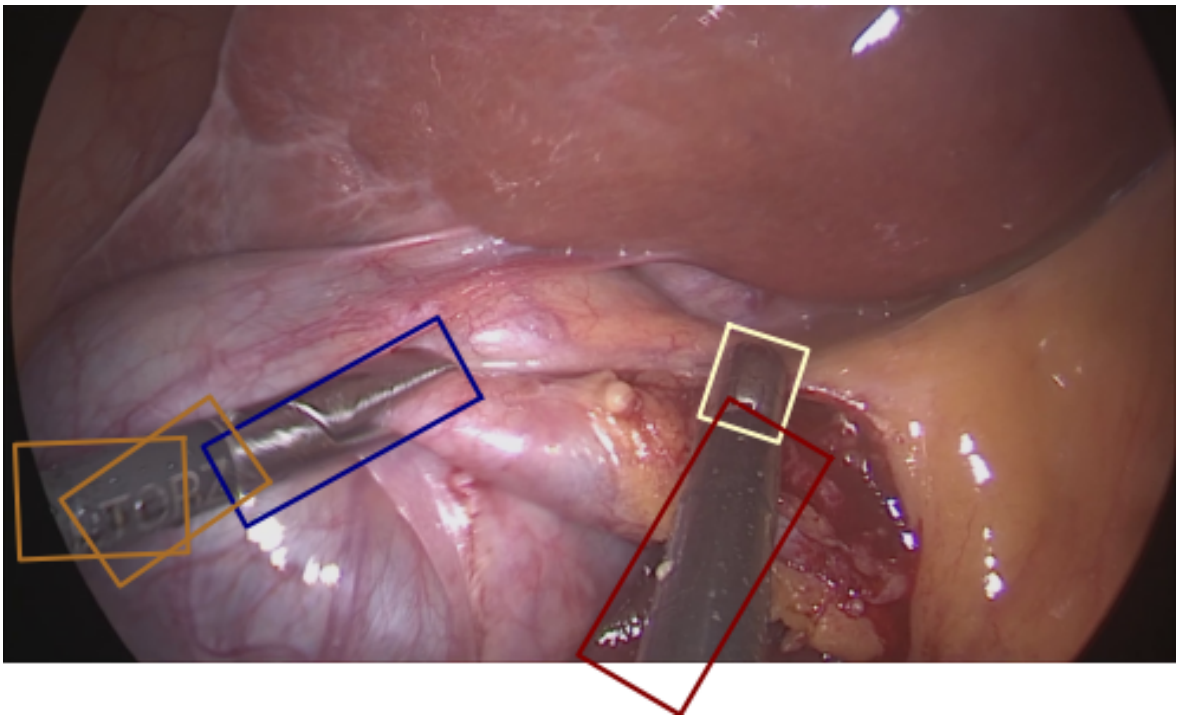

Figure 23: Prediction results with rotated boxes from experiment 1.

7.2.2 Experiment 2

For this experiment we used the same exact parameters, and loss function 15 we used in experiment 1, but this model is trained for 400 epochs, instead of 200.

Number of classes: 18 different class objects, including many laparoscopic instruments and their different parts.

Epochs: 400 forward and backward passes of all training samples.

Batch size: 18 training samples in every forward and backward pass.

Iterations: 120000 passes of 8 samples

lr: A learning rate of 1e-3

$$Loss = Loss_{conf} + Loss_{loc} + 2 * Loss_{angle} \tag{15}$$

Results The training of the second experiment reached the following results. It achieved a final mean average precision of 0.239 in the training and 0.045 in the validation, as shown in Figure 24. The losses reached the following values: A train loss of 4.476, a validation loss of 8.819, a confidence loss of 2.075 in training and 4.940 in validation and a location loss of 0.623 in training and 1.053 in validation. And finally, an angle loss of 0.888 in training and 1.412 in validation.

After training, we calculated the average Jaccard index of a certain batch of images to evaluate how well the bounding box fits the object. The calculated Jaccard index for the rotated bounding boxes for this experiment was 0.893.

In Figure 25 the model prediction results for experiment 2 can be seen.

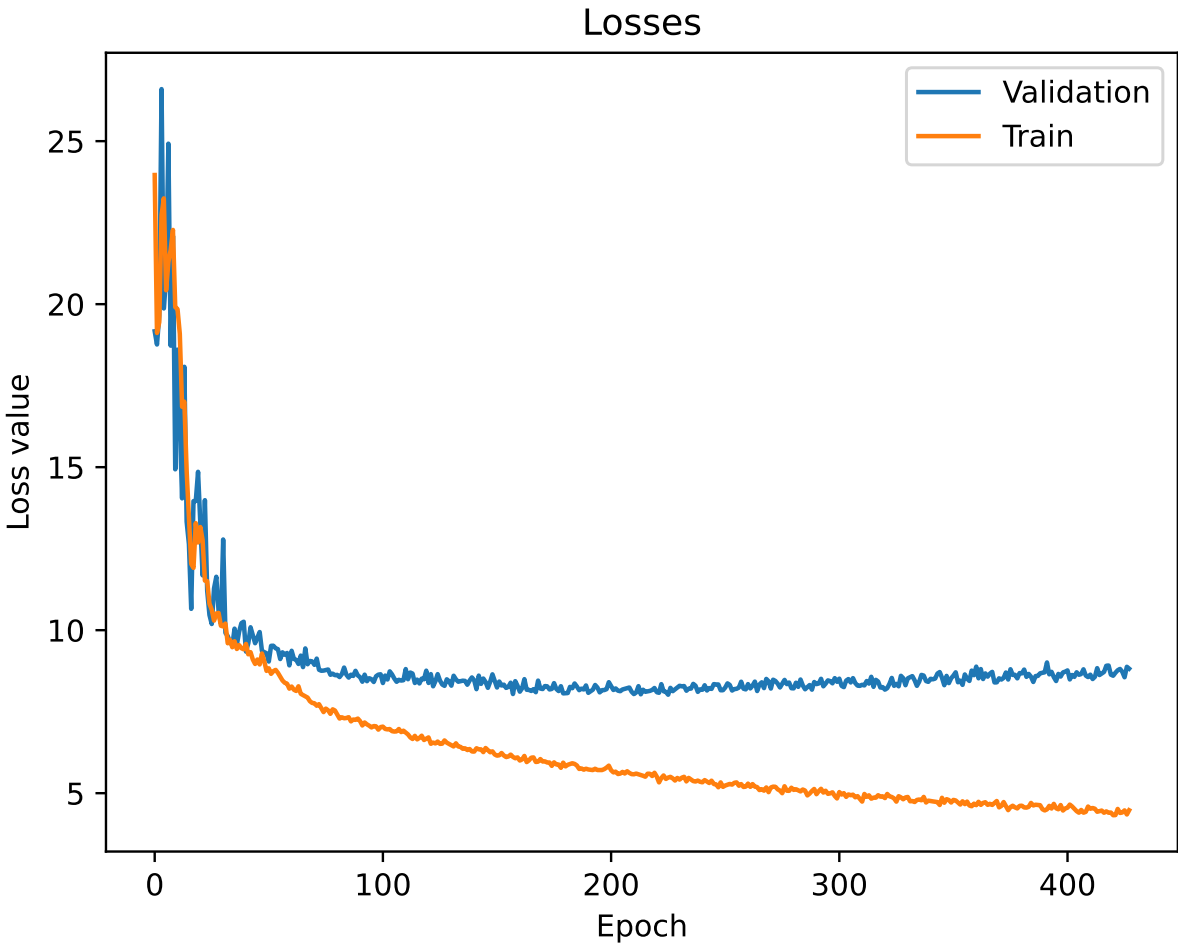

Figure 24: Learning curve with train and validation loss for rotated bounding boxes in experiment 2.

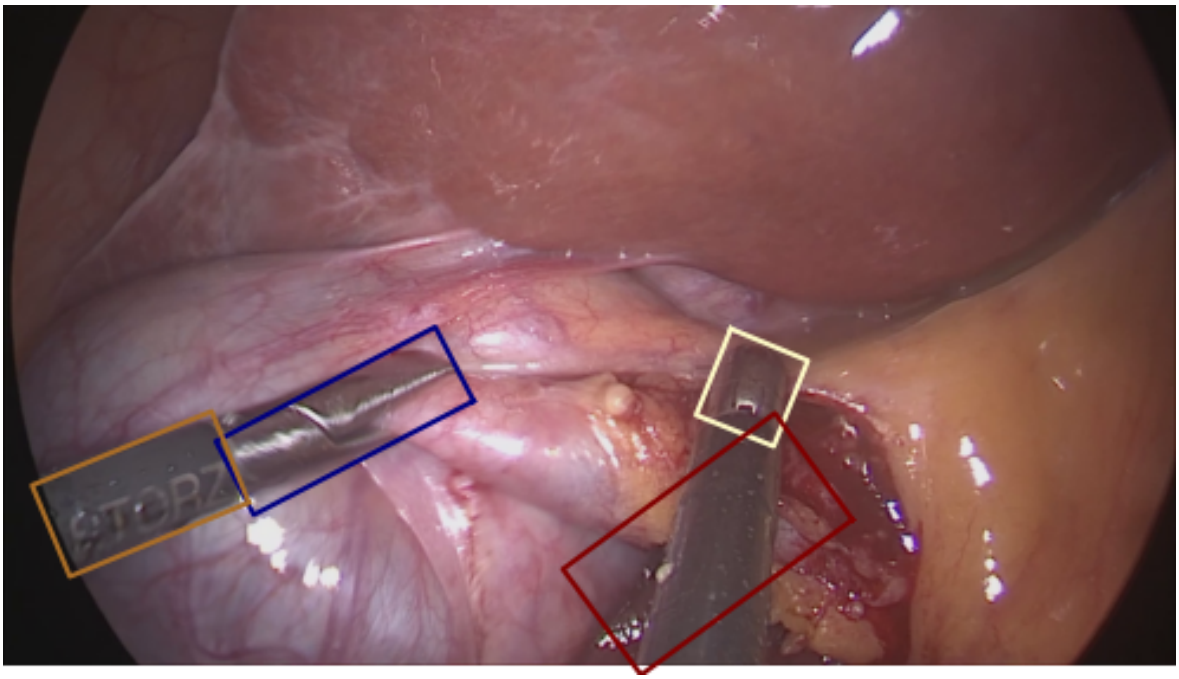

Figure 25: Prediction results with rotated boxes from experiment 2.

7.2.3 Experiment 3

For this experiment, we used a different strategy. We changed our optimization algorithm, SGD (Stochastic Gradient Decent)[33] optimizer, for the Adam (Adaptive Moment Estimation) optimizer [34]. The Adam optimizer is a more sophisticated optimization algorithm that calculates an adaptive learning rate for each parameter in the model. It uses running averages of the gradient and second moments of the gradient to scale the learning rate. This allows Adam to handle sparse gradients and noisy optimization landscapes better than SGD with a fixed learning rate.

In general, Adam is a more advanced optimization algorithm that tends to work well for a wide range of problems, but it is more computationally expensive than SGD. We set the learning rate to the Adams optimizer default, 1e-4. We used the following parameters:

Number of classes: 18 different class objects, including many laparoscopic instruments and their different parts.

Epochs: 85 forward and backward passes of all training samples.

Batch size: 18 training samples in every forward and backward pass.

Iterations: 120000 passes of 8 samples.

lr: A learning rate of 1e-4.

Also, we used a loss function where all losses have equal weight in the function 16.

$$Loss = Loss_{conf} + Loss_{loc} + Loss_{angle} \tag{16}$$

Results This training reached the following results. It achieved a final mean average precision of 0.209 in the training and 0.045 in the validation. As shown in Figure 26, the losses reached the following values: A train loss of 3.846, a validation loss of 6.701, a confidence loss of 2.137 in training and 4.334 in validation and a location loss of 0.643 in training and 1.039 in validation. And finally, an angle loss of 1.065 in training and 1.327 in validation.

After training, we calculated the average Jaccard index of a certain batch of images to evaluate how well the bounding box fits the object. The calculated Jaccard index for the rotated bounding boxes for this experiment was 0.89.

In Figure 27 the model prediction results for experiment 3 can be seen.

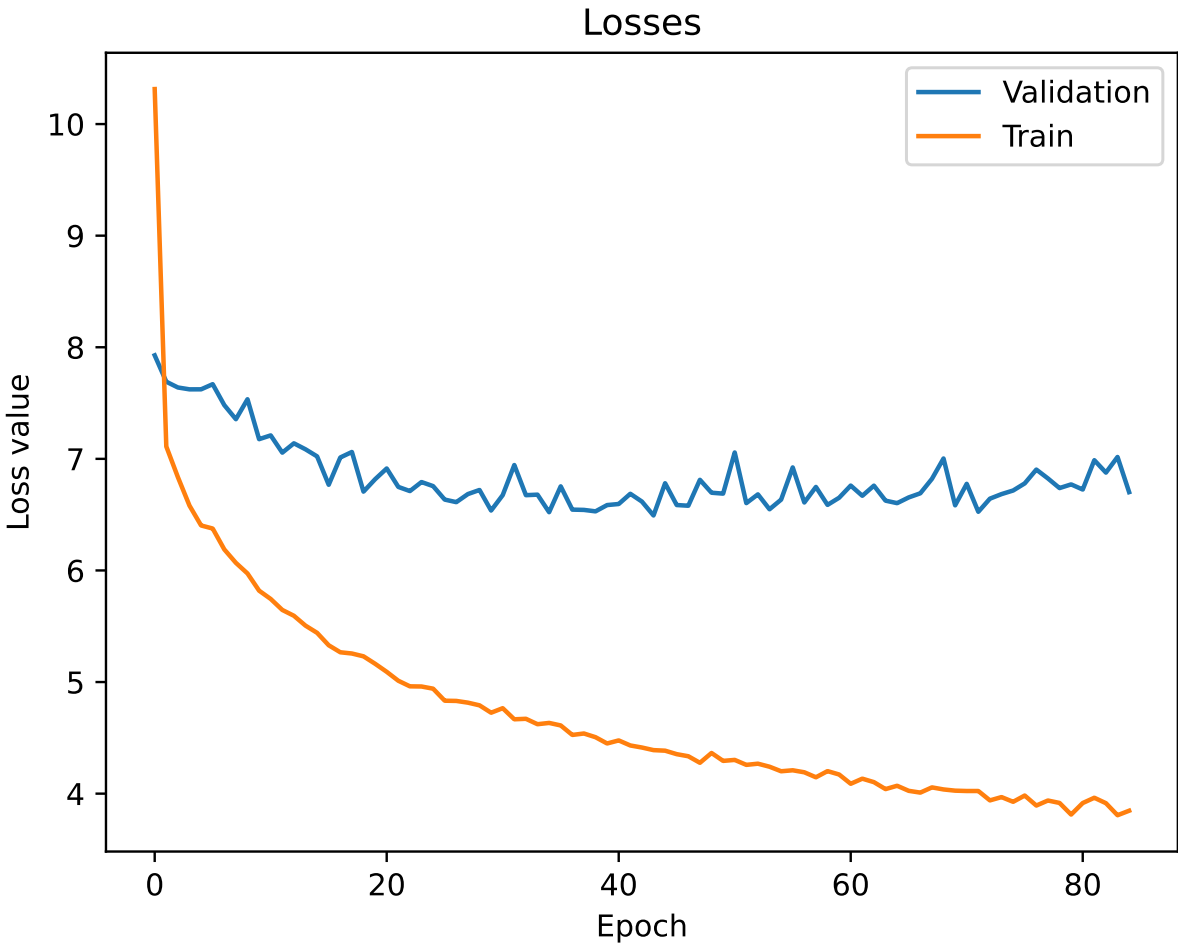

Figure 26: Learning curve with train and validation loss for rotated bounding boxes in experiment 3.

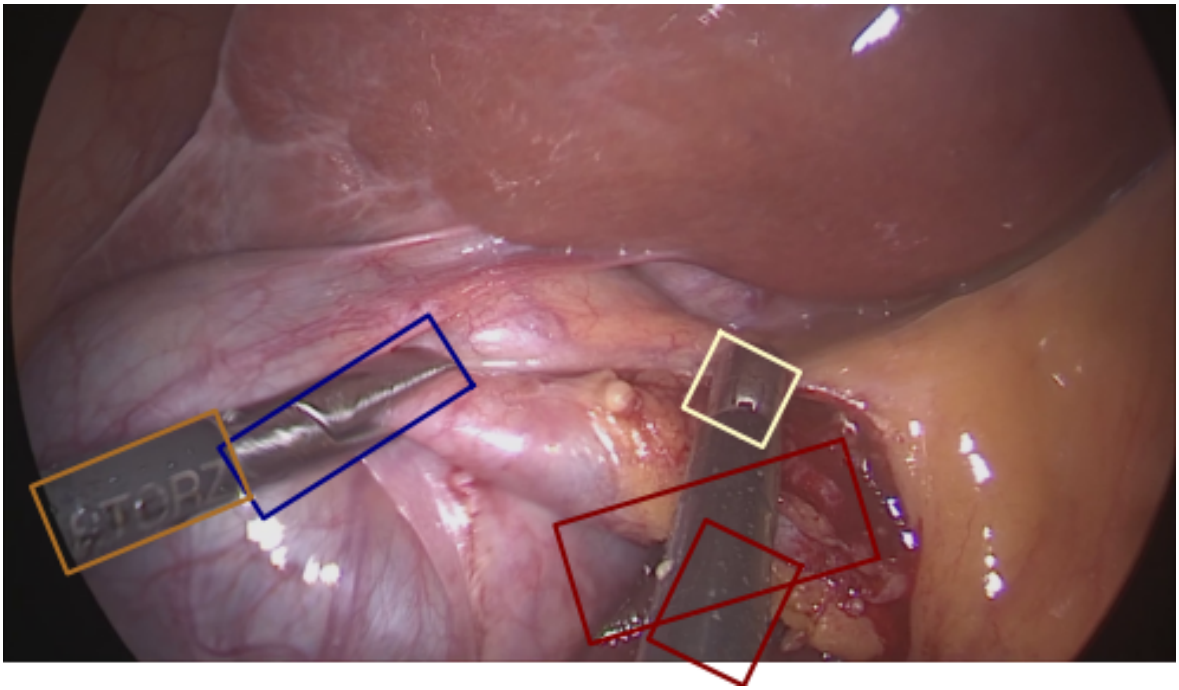

Figure 27: Prediction results with rotated boxes from experiment 3.

7.2.4 Experiment 4

For this experiment, we used the SGD optimizer, and the following parameters:

Number of classes: 18 different class objects, including many laparoscopic instruments and their different parts.

Epochs: 200 forward and backward passes of all training samples.

Batch size: 18 training samples in every forward and backward pass.

Iterations: 120000 passes of 8 samples.

lr: A learning rate of 1e-4.

Also, we gave more importance to the angle loss, as shown in loss function 17.

$$Loss = Loss_{conf} + Loss_{loc} + 2 * Loss_{angle} \tag{17}$$

The main difference between this experiment and the others is that we used a scheduler [35]. A scheduler in training artificial neural networks refers to a mechanism for adjusting the learning rate during the training process.

In deep learning, it is often useful to change the learning rate over time. This is because a high learning rate can help the model escape from local minima early in training, while a lower learning rate can help the model converge towards a global minimum later in training.

A scheduler provides a systematic way to adjust the learning rate during training, typically based on some pre-defined schedule or metric. We used the Step decay scheduler, in which the learning rate is reduced by a fixed factor after a fixed number of training steps.

Using a scheduler can significantly improve the performance of a deep learning model and can help to reduce the need for manual tuning of the learning rate.

Results This training reached the following results. It achieved a final mean average precision of 0.093 in the training and 0.008 in the validation. As shown in Figure 28, the losses reached the following values: A train loss of 4.771, a validation loss of 7.123, a confidence loss of 2.72 in training and 4.584 in validation and a location loss of 0.86 in training and 1.132 in validation. And finally, an angle loss of 1.201 in training and 1.407 in validation.

After training, we calculated the average Jaccard index of a certain batch of images to evaluate how well the bounding box fits the object. The calculated Jaccard index for the rotated bounding boxes for this experiment was 0.888.

In Figure 29 the model prediction results for experiment 4 can be seen.

7.2.5 Overview

In Table 1 we can see for every experiment in the thesis, the total loss, confidence loss, angle loss, mAP for both training and validation. Also, the jaccard index for every experiment.

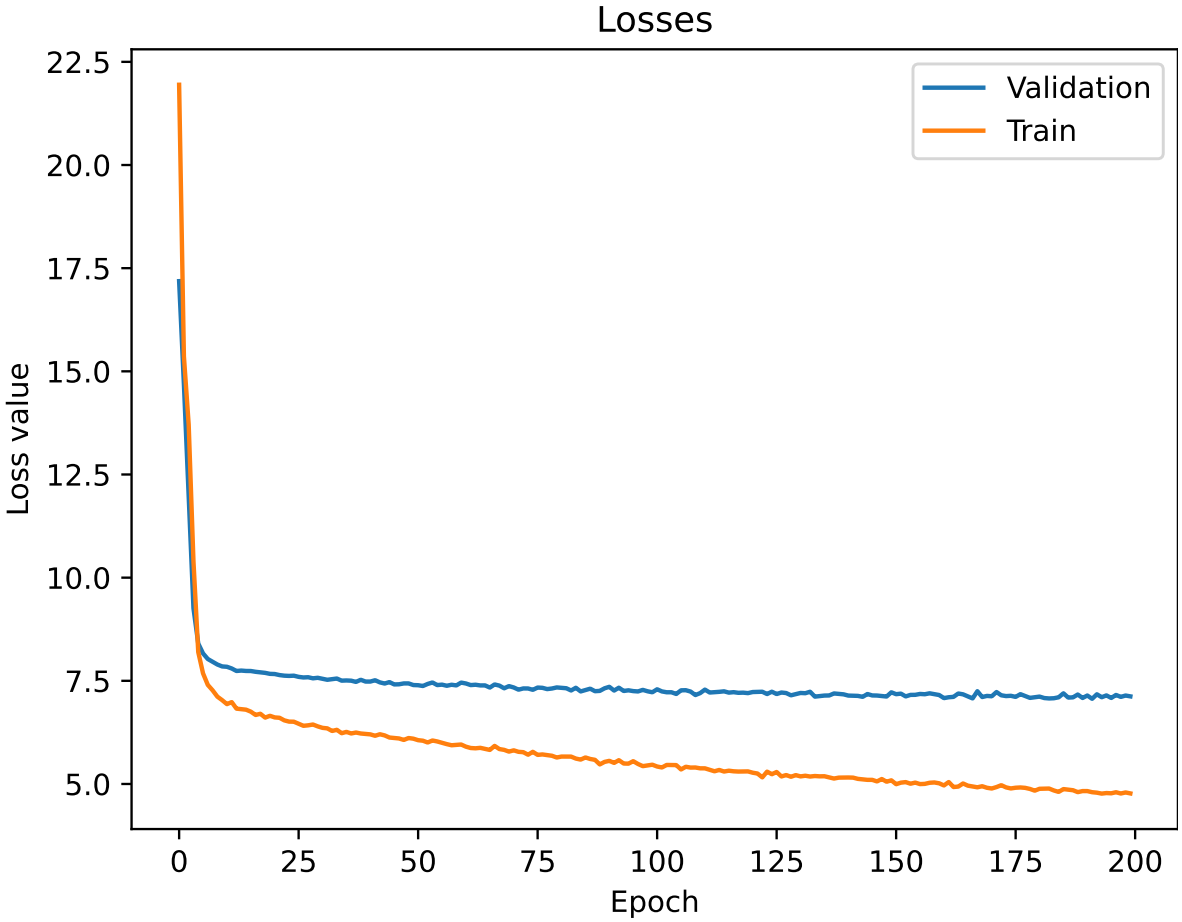

Figure 28: Learning curve with train and validation loss for rotated bounding boxes in experiment 4.

		Total Loss	Conf Loss	Loc Loss	Angle Loss	mAP	Jaccard index
E0	T	1.504	1.159	0.345	-	0.772	0.580
	V	4.085	3.440	0.644	-	0.145	
E1	T	4.476	2.075	0.623	0.888	0.239	0.890
	V	8.819	4.940	1.053	1.412	0.045	
E2	T	4.475	2.075	0.621	0.888	0.240	0.895
	V	8.819	4.940	1.053	1.412	0.045	
E3	T	3.846	2.137	0.643	1.065	0.209	0.890
	V	6.701	4.334	1.039	1.327	0.045	
E4	T	4.771	2.72	0.86	1.201	0.093	0.880
	V	7.123	4.584	1.132	1.407	0.008	

Table 1: Table containing the total loss, confidence loss, angle loss, mAP for both training (T) and validation (V), and the Jaccard index for every experiment. E0 represents experiment object detection with axis aligned boxes and from E1, E2, E3 and E4, correspond to the 4 object detection with rotated bounding boxes experiments.

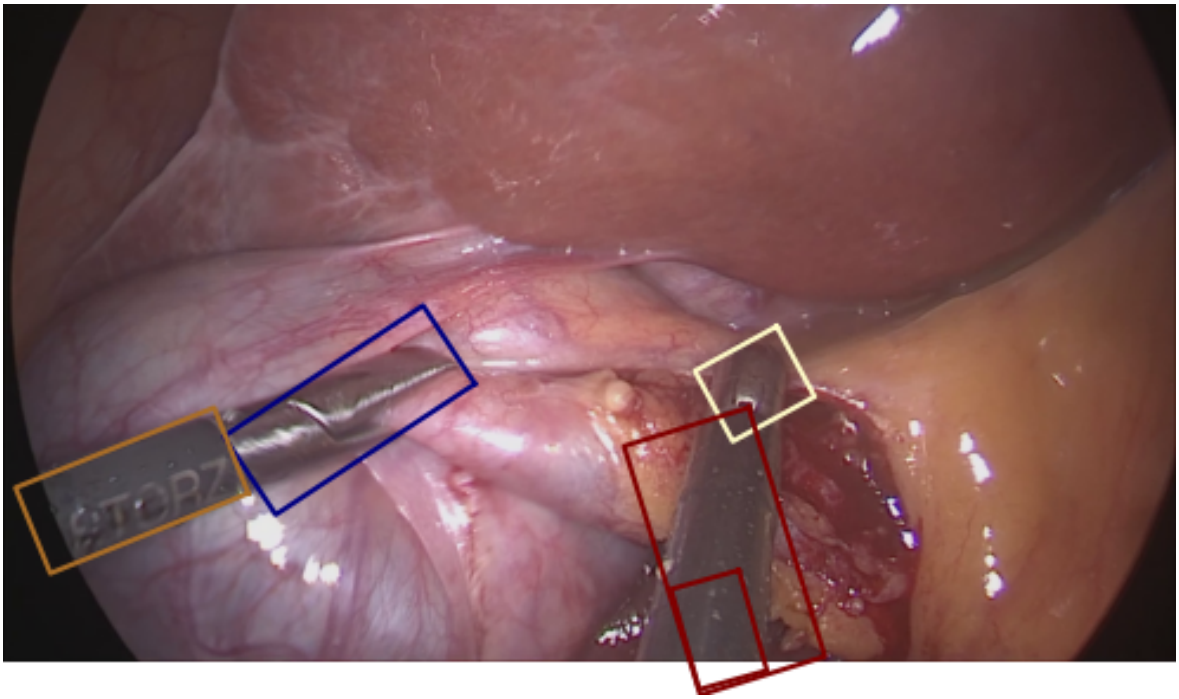

Figure 29: Prediction results with rotated boxes from experiment 4.

8 Discussion

8.1 Object detection with axis aligned boxes

The results of the training for the object detection with axis aligned boxes model show high mAP, which indicates that the model is able to detect objects accurately and precisely. Also show low loss, which means that the model is making fewer errors and is better fitting the data.

In the learning curve we can observe that the training loss and validation loss seems to converge to a specific value and stops decreasing, this could indicate a plateau, where the network has reached its maximum performance and further training may not improve the results.

As we can see in Figures 20 and Figure 21 the model is capable of detecting different Laparoscopic tools in an image and surround them with an upright bounding box with a corresponding label. Even though the validation losses are higher than the training the model is still very effective with unseen validation data. The loss of the model will almost always be lower on the training dataset than the validation dataset. This means that we should expect some gap between the train and validation loss learning curves. This gap is referred to as the generalization gap.

Regarding the Jaccard index, we obtained an average jaccard index of 0.58, from a batch of images from validation and training. This means that on average only 58% of the area contained by the bounding box is occupied by the actual object. This could be significantly improved with the implementation of rotated bounding boxes.

8.2 Object detection with rotating bounding box

Experiment 1 The results of the first experiment show very low mAP specially for the validation data. It seems like the model is able to predict the class, location, and size of the rotated bounding box accurately but lacks the ability of predicting the angles with full precision. Even though the angle loss was given double weight in the loss function, it still stays behind in the learning process compared to the others. Looking at the loss function, we can see that the model could be trained further for the training data, but might get overfitted for the validation data.

Regarding the Jaccard index, we have decided to calculate it from a batch of training images, because the validation results are not good enough to get reasonable values. Even if this decreases the expressiveness, it will be easier to see the difference between the experiments. For this particular batch of images, we obtained a Jaccard index of 0.890. This means that on average, 89% of the area contained by the bounding box is occupied by the actual object.

Experiment 2 In this experiment, the same parameters as in experiment 1 were used, but this one was trained for 200 epochs longer. The analysis of the learning curves indicates that the model begins to overfit after epoch 200, as evidenced by an increase in the validation loss. Given the inability of the model to attain desired performance levels, it is concluded that further training alone can not resolve the issue. Consequently, it is recommended to consider implementing significant modifications to the training strategy. As a consequence, the results are almost the same as in the first experiment. The model can accurately predict the class and location of the rotated bounding boxes but lacks precision in predicting the angles, specially for validation data.

Regarding the Jaccard index, for the same particular batch of training images, we obtained a Jaccard index of 0.895. This means almost 89.5% of the area contained by the bounding box is occupied by the actual object.

Experiment 3 For the third experiment, we used the more sophisticated optimization algorithm, Adam optimizer. Despite being trained for fewer epochs than the other experiments, In the learning curve we can observe that the validation loss seems to converge to a specific value and stop decreasing very early, this could indicate a plateau, where the network has reached its maximum performance and further training may not improve the results.

With the Adam optimizer, all losses are a bit lower than with the SGD optimizer, but the mAP seems to be a bit higher. The model can predict the class and location of the rotated bounding boxes but lacks precision in predicting the angles, specially for validation data.

For the same particular batch of training images, we obtained a Jaccard index of 0.89. This means almost 89% of the area contained by the bounding box is occupied by the actual object.

Experiment 4 The results of the scheduler experiment show the lowest mAP of all the experiments. In the learning curve, we can observe that the training loss and validation loss seem to rapidly converge to a specific value in the beginning of the training, and then it stops decreasing, thanks to the scheduler. However, the model stops where the network has reached its maximum performance, where angle predictions are still inaccurate and further training may not improve the results.

This experiment has the lowest Jaccard index of the four rotated bounding box experiments for the particular batch, with 0.88, or in other words, 88% of the area contained by the bounding box is occupied by the actual object.

8.3 General

In all the experiments, the validation losses were higher than the training losses during the object detection training process. This is a common trend observed in deep learning models, as the validation loss is meant to reflect the model's ability to generalize to new data.

The proposed upright object detection model for laparoscopic tool detection has been shown to be highly accurate in identifying and positioning the tools. However, for applications that require detection of tool rotation, a rotated bounding boxes object detection model should be considered.

The experiment of the proposed rotated bounding boxes object detection model that performed the best was the second one, this one achieved a Jaccard index of 0.895, a considerable improvement from the 0.58 achieved by the model using axis aligned boxes. Although the model can correctly predict the class and location of the rotated bounding boxes, it falls short in predicting the angles accurately, particularly for the validation data. We believe that this is caused by the lack of complexity in the model due to the axis-aligned orientation of the anchor boxes, and could be solved by adding another learnable parameter for the angle of the anchor boxes.

9 Conclusion

In conclusion, the present research aimed to address the issue of detecting surgical instruments in laparoscopic videos using artificial neural networks with oriented bounding boxes. The proposed rotated boxes model was compared with the traditional non-rotated boxes model to determine its effectiveness in fitting bounding boxes around objects. The Jaccard index results of the experiments indicate that the proposed model is capable of more precisely fitting bounding boxes around objects. Nonetheless, further refinement is necessary for it to function optimally, as the model can correctly predict the class and location of the rotated bounding boxes, but it falls short in predicting the angles needed to rotate the bounding box accurately, particularly for the validation data.

Despite these challenges, the proposed object detection model with rotated bounding boxes model holds great potential in enhancing the performance of object detection in laparoscopic surgeries. By providing a more accurate representation of the objects in question, this model can play a key role in improving the overall efficiency and safety of such surgeries. It is also worth noting that the model can be further improved and extended to other medical imaging applications where object detection is crucial. This research provides a strong foundation for future research in this area and can pave the way for the development of more advanced and sophisticated models.

Bibliography

[1] A. A. Bodenstedt S Allan M. Comparative evaluation of instrument segmentation and tracking methods in minimally invasive surgery. *Comput Vis Pattern Recogn*, 2018.

[2] Z. Zhao, S. Voros, Z. Chen, and X. Cheng. Surgical tool tracking based on two CNNs: from coarse to fine. *The Journal of Engineering* 2019(14), 467–472, 2019.

[3] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg. Ssd: Single shot multibox detector. In: *European conference on computer vision*, 21–37, 2016.

[4] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, 779–788, 2016.

[5] R. Girshick. Fast r-cnn. In: *Proceedings of the IEEE international conference on computer vision*, 1440–1448, 2015.

[6] J. Dai, Y. Li, K. He, and J. Sun. R-fcn: Object detection via region-based fully convolutional networks. *Advances in neural information processing systems* 29, 2016.

[7] Z. Tian, C. Shen, H. Chen, and T. He. Fcos: Fully convolutional one-stage object detection. In: *Proceedings of the IEEE/CVF international conference on computer vision*, 9627–9636, 2019.

[8] Y. Lin, P. Feng, J. Guan, W. Wang, and J. Chambers. IENet: Interacting embranchment one stage anchor free detector for orientation aerial object detection. *arXiv preprint arXiv:1912.00969*, 2019.

[9] K. Duan, S. Bai, L. Xie, H. Qi, Q. Huang, and Q. Tian. Centernet: Keypoint triplets for object detection. In: *Proceedings of the IEEE/CVF international conference on computer vision*, 6569–6578, 2019.

[10] H. Law and J. Deng. Cornernet: Detecting objects as paired keypoints. In: *Proceedings of the European conference on computer vision (ECCV)*, 734–750, 2018.

[11] S. Li, Z. Zhang, B. Li, and C. Li. Multiscale rotated bounding box-based deep learning method for detecting ship targets in remote sensing images. *Sensors* 18(8), 2702, 2018.

[12] Z. Zhang, W. Guo, S. Zhu, and W. Yu. Toward arbitrary-oriented ship detection with rotated region proposal and discrimination networks. *IEEE Geoscience and Remote Sensing Letters* 15(11), 1745–1749, 2018.

[13] H. Yang, X. Chu, L. Zhang, Y. Sun, D. Li, and S. J. Maybank. QuadNet: Quadruplet loss for multi-view learning in baggage re-identification. *Pattern Recognition* 126, 108546, 2022.

[14] Q. Ming, L. Miao, Z. Zhou, X. Yang, and Y. Dong. Optimization for arbitrary-oriented object detection via representation invariance loss. *IEEE Geoscience and Remote Sensing Letters* 19, 1–5, 2021.

[15] X. Yang, J. Yang, J. Yan, Y. Zhang, T. Zhang, Z. Guo, X. Sun, and K. Fu. Scrdet: Towards more robust detection for small, cluttered and rotated objects. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 8232–8241, 2019.

[16] A. Boonkong, D. Hormdee, S. Sonsilphong, and K. Khampitak. Surgical Instrument Detection for Laparoscopic Surgery using Deep Learning. In: *2022 19th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON)*, 1–4, 2022.

[17] IBM. What is a neural network? URL: `https://www.ibm.com/topics/neural-networks` (visited on 12/11/2022).

[18] S. Sharma, S. Sharma, and A. Athaiya. Activation functions in neural networks. *towards data science* 6(12), 310–316, 2017.

[19] D. Kingma and J. Ba. Adam: A Method for Stochastic Optimization. Published as a conference paper at ICLR 2015. *arXiv preprint arXiv:1412.6980*, 2015.

[20] W. Zhiqiang and L. Jun. A review of object detection based on convolutional neural network. In: *2017 36th Chinese control conference (CCC)*, 11104–11109, 2017.

[21] A. Maier, C. Syben, T. Lasser, and C. Riess. A gentle introduction to deep learning in medical image processing. *Zeitschrift für Medizinische Physik* 29(2), 86–101, 2019.

[22] ujjwalkarn. An Intuitive Explanation of Convolutional Neural Networks. URL: `https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/` (visited on 12/20/2022).

[23] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE* 86(11), 2278–2324, 1998.

[24] Autor unbekannt. ImageNet. URL: `https://image-net.org/` (visited on 12/20/2022).

[25] J. Talukdar, S. Gupta, P. Rajpura, and R. S. Hegde. Transfer Learning for Object Detection using State-of-the-Art Deep Neural Networks. In: *2018 5th International Conference on Signal Processing and Integrated Networks (SPIN)*, 78–83, 2018.

[26] R. Faster. Towards real-time object detection with region proposal networks. *Advances in neural information processing systems* 9199(10.5555), 2969239–2969250, 2015.

[27] D. Erhan, C. Szegedy, A. Toshev, and D. Anguelov. Scalable object detection using deep neural networks. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2147–2154, 2014.

[28] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. *arXiv preprint arXiv:1312.6229*, 2013.

[29] C. Szegedy, S. Reed, D. Erhan, D. Anguelov, and S. Ioffe. Scalable, high-quality object detection. *arXiv preprint arXiv:1412.1441*, 2014.

[30] L. Zhang, L. Lin, X. Liang, and K. He. Is faster R-CNN doing well for pedestrian detection? In: *European conference on computer vision*, 443–457, 2016.

[31] D. Hu. An introductory survey on attention mechanisms in NLP problems. In: *Intelligent Systems and Applications: Proceedings of the 2019 Intelligent Systems Conference (IntelliSys) Volume 2*, 432–448, 2020.

[32] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft coco: Common objects in context. In: *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*, 740–755, 2014.

[33] J. Yang and G. Yang. Modified convolutional neural network based on dropout and the stochastic gradient descent optimizer. *Algorithms* 11(3), 28, 2018.

[34] Z. Zhang. Improved adam optimizer for deep neural networks. In: *2018 IEEE/ACM 26th international symposium on quality of service (IWQoS)*, 1–2, 2018.

[35] Y. Peng, Y. Zhu, Y. Chen, Y. Bao, B. Yi, C. Lan, C. Wu, and C. Guo. A generic communication scheduler for distributed dnn training acceleration. In: *Proceedings of the 27th ACM Symposium on Operating Systems Principles*, 16–29, 2019.

List of Figures

List of Tables

List of Abbreviations

Adam Adaptive Moment Estimation

ANN Artificial Neural Network

CNN Convolutional Neural Networks

COCO Common Objects in Context

FCOS Fully Convolutional One-stage Object Detection

GPU Graphics processing unit

ILSVRC ImageNet Large Scale Visual Recognition Challenge

JSON JavaScript Object Notation

lr Learning Rate

mAP Mean Average Precision

MIS Minimally invasive surgery

MSE Mean Squared Error

R-CNN Region-based Convolutional Neural Network

R-FCN Region-Based Fully Convolutional Networks

RELU Rectified Linear Unit

ReMIC Regensburg Medical Image Computing

RPN Region Proposal Network

RRPN Rotated Region Proposal Network

SGD Stochastic Gradient Decent

SSD Single Shot MultiBox Detector

YOLO You Only Look Once