Detecting red blood cells and platelets in blood smears using a single multi-class object detector

Timothy Burfield, Sophia Carlsson

Master's thesis
2022:E20

Lund University

Faculty of Engineering
Centre for Mathematical Sciences
Mathematics

Centrum Scientiarum Mathematicarum

LUND UNIVERSITY

Master Thesis

Detecting red blood cells and platelets in blood smears using a single multi-class object detector

Spring 2022

Timothy Burfield
Sophia Carlsson

Supervisors:
Robin Bram, CellaVision
Charlotte Oom, CellaVision
Anders Heyden, Lund University

Examinator:
Niels Christian Overgaard, Lund University

Abstract

Blood analysis is an integral part of diagnostic medicine and used in most medical fields. The concentrations of red blood cells and platelets, and ratio between these, are used for diagnosing several diseases. CellaVision develops machines and software for automatically capturing images of blood sample smears and detecting its cellular contents. The company currently has separate algorithms for detecting red blood cells and platelets. The aim of this master's thesis is to develop an object detection model that simultaneously detects these blood cell types, with sufficiently high accuracy and speed for use on CellaVision systems.

The object detection model YOLOv5 was selected to develop the detector. Several model parameters, hyper parameters and improvement techniques were evaluated, and the ones maximising the performance were selected. Image augmentations proved to be the most important improvement technique added during development in terms of detection performance. Pseudo labelling was successfully used for creating a large training data set. The results obtained show that it is possible to combine red blood cell and platelet detections in a single object detector with higher speed than when using separate algorithms. Comparing performance with the current individual algorithms, platelet detection was almost as good and red blood cell counting significantly better when using the detector developed during this thesis.

Acknowledgements

First and foremost, we would like to thank our supervisors, Robin Bram and Charlotte Oom, at CellaVision. With his great knowledge, Robin has always been available to answer our questions and an exceptional asset when discussing different machine learning approaches. Charlotte has supported us in every step of the process and given us knowledge about what is important both clinically and for CellaVision. We would also like to express our gratitude towards CellaVision for giving us the opportunity to write our master thesis at the company. Several employees have helped us with understanding blood samples and classified cell types, and they have all made us feel welcome here. We would also like to thank Anders Heyden, our supervisor at the Department of Mathematics at Lund University, for the constant support and feedback during our work, as well as insightful thought on how to improve our obtained results.

Finally, we would like to acknowledge our friends and family that have not only given us feedback on the report and constant support during this thesis, but during our entire education. None of this would be possible without you.

Contents

Abbreviations

- **ANN** - Artificial Neural Network
- **AP** - Average Precision
- **AUC** - Area Under the Curve
- **CNN** - Convolutional Neural Network
- **FN** - False negative
- **FP** - False positive
- **FPN** - Feature Pyramid Network
- **GIoU** - Generalised Intersection over Union
- **IoU** - Intersection over Union
- **LR** - Learning Rate
- **MGG** - May Grünwald Giemsa stain
- **NMS** - Non-Maximum Suppression
- **NN** - Neural Network
- **PLT** - Platelet
- **PLT_CV** - Current PLT detector at CellaVision
- **RBC** - Red blood cell
- **RBC_CV** - Current RBC segmentation algorithm at CellaVision
- **R-CNN** - Region based Convolutional Neural Network
- **RoI** - Region of Interest
- **SSL** - Semi-Supervised Learning
- **TN** - True negative
- **TP** - True positive
- **W** - Wright stain
- **WBC** - White blood cell
- **WG** - Wright Giemsa stain
- **YOLOv5_SoTi** - Our final RBC + PLT detection network

1 Introduction

Blood analysis plays a central role in modern medicine. Samples of blood are taken across most medical fields and used to detect numerous deficiencies and diseases. One of the more common uses of blood samples is taking a drop of peripheral blood and smearing it on a glass microscope slide, with the aim of part of the smear being a single layer of cells. CellaVision has developed hardware that automatically take high-resolution pictures of blood smears, in many cases replacing the need for manual microscopes. In addition to hardware, CellaVision develops automation software for locating blood cells and abnormalities. This means significantly faster analyses compared to manual microscopy methods and a more standardised method.

There are three main types of blood cells: red blood cells, white blood cells and platelets. Their occurrence and morphology in blood smears may indicate diseases. In this project we primarily look at the occurrence of red blood cells and platelets but also touch upon their morphology. The ratio between red blood cells and platelets is normally around 20:1 and deviations from this may be an indication of illness. The size of platelets can be a further indicator, which was considered during the project.

A low execution time on the CellaVision hardware's CPUs is paramount when developing new detection methods. CellaVision currently use separate algorithms for counting the number of red blood cells and platelets in blood smears. Red blood cells are detected using a segmentation algorithm and platelets are detected using an object detection network. For the sake of speed and simplicity, it is of interest to combine the detections. It is also of interest to see if an object detection algorithm is superior to the segmentation algorithm used for red blood cell counting.

For this task we studied and applied a sub-class of artificial neural networks, called object detectors. Object detectors are algorithms which both detect objects in images and classify them. There are a variety of approaches which each have their individual strengths and weaknesses. After studying a few of them a method called YOLOv5 was chosen. It combines very high detection speed with accuracy amongst the state-of-the-art.

1.1 Aim and Limitations

The aim of this master's thesis is to develop an object detector that simultaneously detects and classifies red blood cells and platelets, with an execution time lower than what is today possible with CellaVision's red blood cell and platelet detection algorithms. Due to this aim as well as the limited project duration, only the YOLOv5 algorithm has been evaluated.

2 Background

2.1 Blood Cell Analysis

Blood cell analyses are most commonly done using cell counters or microscopy. Automated cell counters force the blood sample through a tube and use electrical or optical impedance to count how many of each cell type goes through the tube based on the size of them. Microscopy on the other hand can be done either manually where a person counts all the different cell types based on their appearance, or automatically where this is done with image analysis and object detection.

2.1.1 Peripheral Blood

Human blood consists of two components: plasma and blood cells. Plasma makes up approximately 60% of the blood volume. It mainly consists of water, but also carries important particles such as proteins, sugars and fats. Blood cells are primarily created in the bone marrow as stem cells, and then mature into erythrocytes (red blood cells, RBCs), leukocytes (white blood cells, WBCs) or thrombocytes (platelets, PLTs), see [1]. Their appearance can be seen in Figure 2.1.

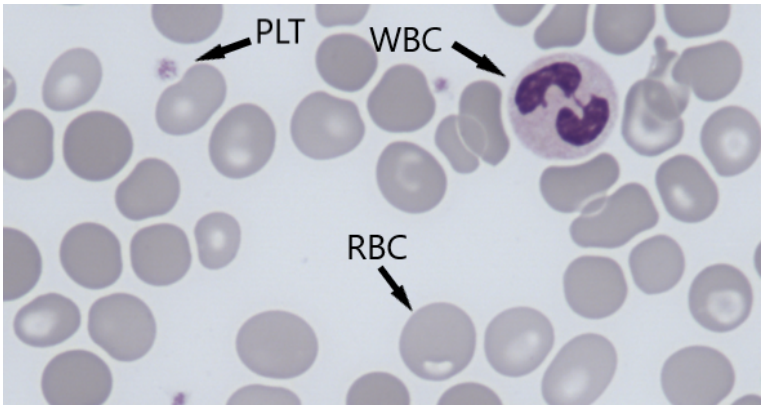
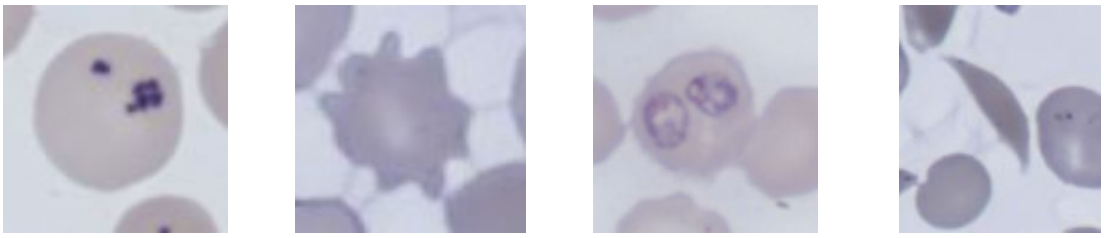


Figure 2.1: Image of a blood sample containing RBCs, a WBC and PLTs. The image was captured with CellaVision's systems.

Erythrocytes
Erythrocytes are the most common blood cells, with the main function of transporting oxygen to all parts of the body. They also carry carbon dioxide and other waste products out of the body. When red blood cells are formed in the bone marrow, they have a nucleus and do not contain the oxygen-carrying protein haemoglobin. During maturation the cells lose their nucleus and haemoglobin appears in the cell. A healthy adult person should not have nucleated erythrocytes in their blood stream, see [2].

According to [3], normal RBCs have the shape of a biconcave disk with a diameter of about 7-8µm. Deviations from the normal appearance may indicate medical conditions or diseases. Abnormal RBCs can vary in size, colour and shape and can contain inclusions and parasites. Examples of abnormal RBCs are shown in Figure 2.2.



(a) Pappenheimer inclusions (b) Acanthocyte (c) Malaria parasites (d) Sickle cell

Figure 2.2: Examples of RBC abnormalities, classified by a morphology expert at CellaVision.

Leukocytes
White blood cells are, unlike red blood cells, nucleated even in the blood stream. They appear in various shapes and sizes and are specialised for different tasks. They play a big role in both the body's immune defence mechanisms and reparation. As they contain a nucleus and are able to produce RNA they can synthesise proteins, see [4].

Thrombocytes
Platelets are the smallest blood cells in humans. They do not carry a nucleus and are very important in the prevention and control of bleeding. When the surface of a blood vessel is injured, the platelets immediately attach to the injured site and to each other forming a mesh that stops the bleeding. The platelets also contribute with substances important for normal coagulation, as well as for reduction and retraction of already formed blood clots, see [5]. It is important to know if there is an abnormal number of thrombocytes present in the blood to be able to give patients the right treatment.

The medical term for a decreased number of thrombocytes is thrombocytopenia. This condition impairs the body's ability to form blood clots, therefore prolonged bleeding from small wounds is likely to occur. When the platelet count is *very* low, spontaneous internal bleeding could start, such as a stroke. Thrombocytopenia can also indicate a more serious condition such as cancer or a severe infection.

An increased number of platelets is called thrombocytosis. It can happen because of overproduction in the bone marrow or due to an ongoing condition or disease such as anemia, inflammation, infection or cancer. Thrombocytosis can lead to blood clots in arms or legs, and this in turn can lead to a heart attack or stroke, see [6], [7]. According to [8], a normal RBC:PLT ratio is around 20:1.

In addition to the number of platelets, the size of them can also be of importance. Large platelets are called macrothrombocytes and have a diameter of ca 4-7µm. Platelets over

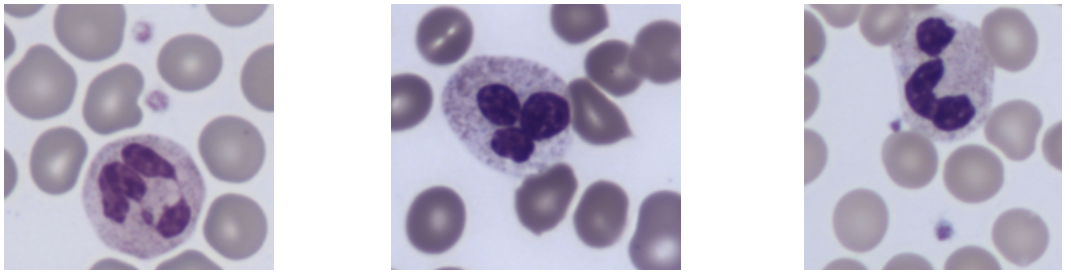
ca 7µm are called giant platelets. In automated cell counters that only classify cells based on size, giant platelets can sometimes be classified as RBCs and a high presence of these may cause the platelet count to be incorrectly low, see [9].

When calculating the PLT concentration at CellaVision, the number of RBCs and PLTs in the image as well as the RBC concentration (million cells per microliter blood) from the cell counter are needed. The concentration of PLTs is then calculated using Equation (2.1), where $RBC_{Cellcounter}$ is the concentration received from the cell counter, and $\#RBC$ and $\#PLT$ the received numbers of cells from the CellaVision application.

$$\mathrm{PLT}_{\mathrm{concentration}} = \mathrm{RBC}_{\mathrm{Cellcounter}} * \frac{\#\mathrm{PLT}}{\#\mathrm{RBC}} \tag{2.1}$$

2.1.2 Blood Smear Preparation

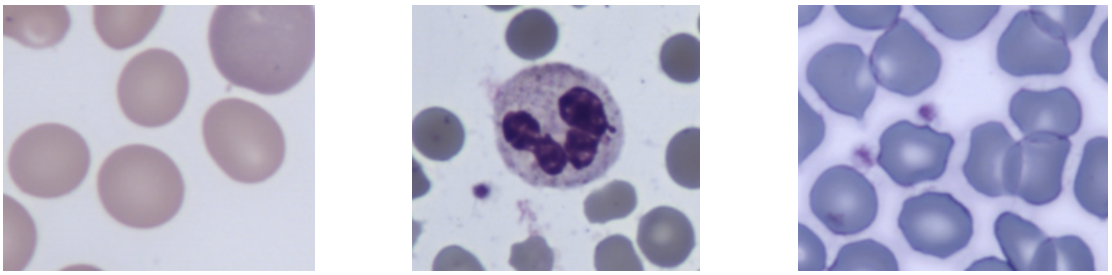
There are several techniques for making blood smears, both regarding the smearing and the staining. The applications that CellaVision offer can handle both manually and automatically made smears, and three stains: May Grünwald Giemsa (MGG), Wright Giemsa (WG) and Wright (W), see [10]. These three are all Romanowsky stains and are commonly used in studies within haematology and for bone marrow samples, see [11]. Staining is used to be able to differentiate cells in microscopy. It could also be used to highlight and find malaria or other parasites. These stains are all natural and consist of oxidized methylene blue (azure) dyes and Eosin Y. The azure and Eosin react with different cellular components and the idea of the stain is to be able to differentiate the components more easily through the formation of varying hues. Which stain is used can vary between labs and depending on what is being examined. The different stains also create a variety in colour and intensity, as can be seen in Figure 2.3.



(a) MGG (b) Wright Giemsa (c) Wright

Figure 2.3: Three different stains.

In addition to using different stains, different labs may also use varying protocols for the staining procedure. The concentration of the dyes and the time each slide is dyed can vary. This may result in even more variety between the images. An example of this is shown in Figure 2.4 where the same stain results in different colours for the same blood cell types.



(a) (b) (c)

Figure 2.4: Three blood smear slides dyed with Wright Giemsa, but with varying appearances.

2.2 Artificial Neural Networks

An Artificial Neural Network (ANN) is a special type of machine learning algorithm inspired by the human brain's network of neurons. Different parts of the brain are responsible for recognising and making decisions about different types of information, which is imitated in ANNs by different nodes and layers. There are three types of layers in an ANN: the *input layer* where the data enters the system, the *hidden layer(s)* where the data is processed, and the *output layer* where the system makes a decision on what to do based on the information fed through from the hidden layer(s).

An ANN consisting of multiple hidden layers that process data and feed it forward through the network can also be referred to as a Deep Neural Network. Deep systems are self-teaching where the system learns as it goes by filtering the information through the multiple layers, similarly to how the human brain processes information, see [12]. The architecture of a neuron in a neural network is shown in Figure 2.5, with the corresponding Equation (2.2).

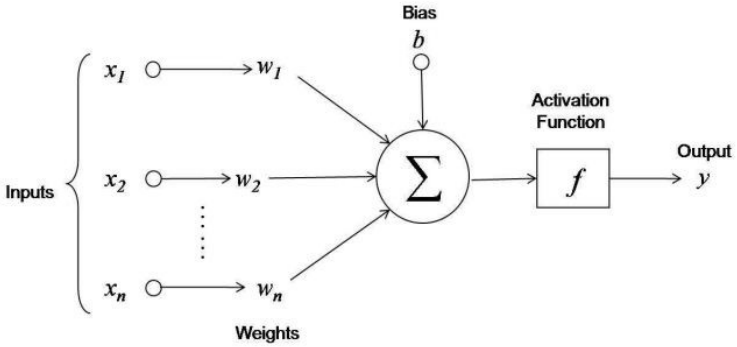


Figure 2.5: Illustration of a neuron in an Artificial Neural Network [13].

$$y = f(\sum_{i=1}^{n} x_i w_i + b) \tag{2.2}$$

2.2.1 Weights and Biases

The large circle in Figure 2.5 represents a node in an ANN that has a threshold. The negative value of the threshold is called the bias. Every node can have connections to several nodes in both the previous and the next layer. Each incoming connection is assigned a weight from the receiving node. While active, the node receives a different data item from each of its connections, and multiplies it with its weight, see [14]. All the resulting products are then summed together with the bias and fed into an activation function. In the beginning of the training phase, the weights and biases are set to random values. During training these values are adjusted until the network yields the desired output, see [15].

2.2.2 Activation Functions

Activation functions are applied to determine the output from a neural network and to decide whether information should be passed to the next layer or not. They map the value of a neuron between for example 0 and 1 or -1 and 1, depending on the activation function applied. These functions can basically be divided into two subgroups: linear and non-linear functions. The linear function is not widely used, since it is simply a line corresponding to the equation $f(x) = cx$. The most commonly used non-linear activation functions are sigmoid, tanh, ReLu and leaky ReLu. Visualisations and the corresponding equations for each of these are shown in Figure 2.6. Two important terms used for non-linear activation functions are *differentiable* and *monotonic*, see [16]. A differentiable function has a derivative at any point, and does not include any break, angle or cusp. This enables back propagation of the error in the model to correctly adjust the weights. The activation function should be differentiable because we want to calculate the change in error with respect to given weights, see [17]. A monotonic function is a function which is either entirely non-increasing or non-decreasing. If this is not the case, a change in the weight of the neuron can cause the neuron to have less influence on the reduction of the error, see [18].

Sigmoid Function
The sigmoid function maps the neuron value in the range $[0, 1]$, and is therefore widely used when the probability of an output is being predicted. The function is both differentiable and monotonic. A function that is very similar to the sigmoid function is the softmax function, which is a more generalised logistic function that is used for multi-class problems. The softmax function assigns each class a probability between 0 and 1, where all probabilities sum up to 1.

ReLu and Leaky ReLu Function
The ReLu function maps the output to the range $[0, \infty]$ and is a monotonic function. If the input value is negative it will be mapped to zero, and otherwise be equal to the input value. The problem with all negative values being directly mapped to zero is that it decreases the ability for the model to learn from the data properly. Mapping all negative

values to zero would result in them being considered as exactly the same. An attempt to address this problem resulted in the leaky ReLu, which introduces ϵ in the equation in Figure 2.6 that allows slightly negative values. Both the leaky and randomised ReLu are monotonic functions, see [16].

Tanh Function
The tanh function maps the output to the range $[-1, 1]$ and is most commonly used for binary classification. The advantage of the tanh function is that negative values can be mapped as strongly negative, while zero inputs can be mapped as very close to zero. The gradient is strong, meaning the function is both differentiable and monotonic.

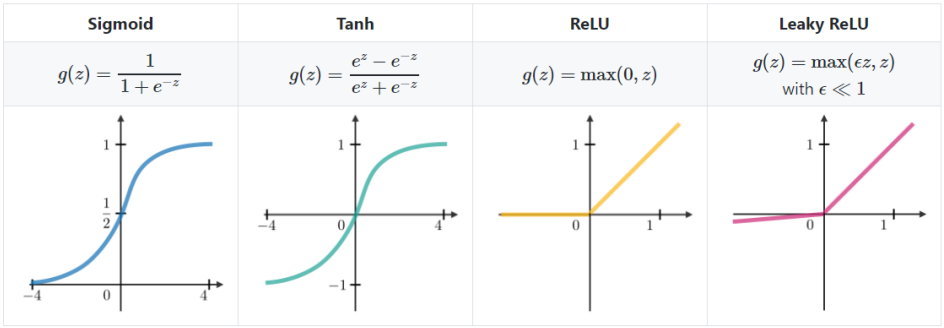


Figure 2.6: Plots of the activation functions described above with their corresponding equations [19].

2.2.3 Loss Functions and Optimisers

In order for an ANN to learn from data there needs to be some type of guidance for how the weights should be updated between training epochs. This guidance comes from what is called a loss function (also called cost function). The goal of the loss function is to quantify the error between the predictions and the ground truth, see [20]. What this error should be depends on the task of the network. For example, it does not make sense to describe the error of a bounding box not being placed correctly in an image in the same way as describing the error of the predicted probability of a class. It is therefore the responsibility of the developer to choose an appropriate loss function for their specific task. The loss functions used in our networks are more elaborately described in section 2.4.4.

Having a loss function is not enough for an ANN to learn during training. There has to be some way of minimising the prediction errors based on the loss function. This problem is solved with a so-called optimiser. Optimisers are mathematical functions that calculate how the weights and biases should change during training in order to minimise the loss. The simplest yet most commonly used is gradient descent and variations of it, such as stochastic gradient descent (SGD). Gradient descent calculates the negative gradient of the loss function through its first order derivative and guides the network towards the minimal loss between iterations of weight updates, see [21] This is illustrated in Figure 2.7 where the dots are training iterations. A more modern algorithm, also widely used, is the Adam optimiser. It is based on gradient descent but adds complexity in the form of two

parameters which, based on the first and second moments of the gradients, control how each individual weight in the network is updated during training, see [22].

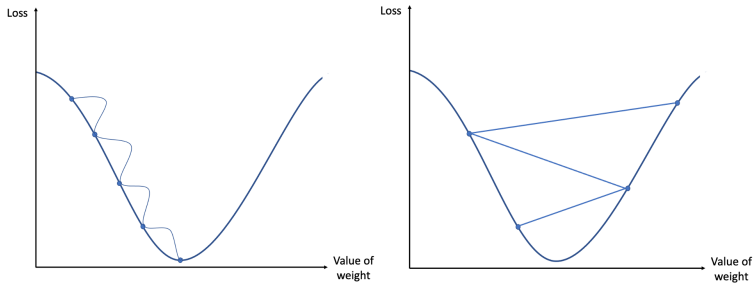


Figure 2.7: An illustration of gradient descent where the weights are adjusted in the direction of the negative gradient between iterations. The left image shows a small learning rate and the right image a large learning rate. Adapted from [23].

2.2.4 Forward and Backpropagation

The learning phase of a neural network is based on *forward propagation* and *backpropagation*. Forward propagation pertains the stage where information if being fed from the input layer to the output layer. The weighted sum from the previous nodes is calculated together with the bias, stored and passed to the activation function, see Section 2.2.2, see [24]. The predictions resulting from the forward propagation together with a loss function result in an error. This error is propagated backwards through the layers and the gradient is computed. Then the optimiser, further explained in Section 2.2.3, uses this gradient to perform learning. Every forward propagation is followed by a backpropagation which means that the network calculates a new error and adjusts its weights between iterations, ultimately minimising the loss, see [25].

2.2.5 Data Annotation

Data annotation is the process where data is labelled in for example text, image or video format. In supervised training algorithms this is required for giving the model correct and precise information. There are several types of annotations, but in this thesis two-dimensional bounding boxes are used. [26]

2.2.6 Learning Methods

Depending on the availability of labelled data, there are different methods for the network to learn from it. The methods relevant to this thesis can be divided into three categories: *supervised, unsupervised* and *semi-supervised* learning.

Supervised Learning
Supervised learning means using a pre-labelled data set to train a network to predict or classify the outcomes, see [28]. This method allows the algorithm to measure the accuracy

and loss correctly, and adjusts the weights and biases according to these until the error is as small as possible. The challenge with supervised learning is that it often requires expertise to label correctly, meaning that it can be very time consuming, difficult and expensive to gather enough data for the task at hand.

Unsupervised Learning
Unsupervised learning clusters data sets into label groups without any human intervention. These algorithms find patterns or groupings on their own, and discover similarities in the data. They are most commonly used for three main tasks: clustering, association, and dimensionality reduction. The down-side of unsupervised learning is the increased probability of inaccurate labelling and that human intervention is needed to validate the results, see [29].

Semi-Supervised Learning
This method involves a small set of labelled data, and a large set of unlabelled data. Neither supervised nor unsupervised learning methods can make use of this mixture of data in an effective way, why semi-supervised learning could be required. Since gathering labelled data can be both time consuming and costly, *pseudo labels* can be used. Pseudo labelling is a method where a model first is trained using a data set containing manually labelled data. That model is then used to predict labels for the unlabelled data set. The final model is trained on a combination of the manually and pseudo labelled data sets, see [27]. This method gets benefits of both supervised and unsupervised learning, but has an uncertainty regarding the pseudo labels being correct, see [30].

2.2.7 Performance Metrics

To evaluate how well a network performs and how much it needs to be adjusted different performance metrics are used. The most common ones in object detection are *intersection over union, precision, recall, F1 score* and *average precision.*

Intersection over Union (IoU)
When working with bounding boxes it is of importance to evaluate how well the predicted boxes align with the ground truth boxes. IoU is the most popular evaluation metric for this task. It is calculated with Equation (2.3) where A is the predicted box, B the ground truth box, I the intersection area and U the union area. The best possible score is 1 meaning perfect alignment, and the worst possible score 0 meaning no alignment, see [31]. A visual explanation of the equation is shown in Figure 2.8.

$$\text{IoU} = \frac{|A \cap B|}{|A \cup B|} = \frac{|I|}{|U|} \tag{2.3}$$

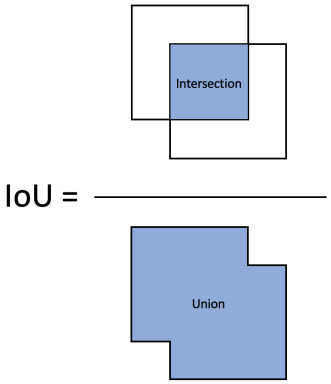


Figure 2.8: Visual explanation of the IoU calculation.

Precision, Recall & F1 score
When labelling the content of the predicted boxes in object detection there is no single metric that mediates a perfect understanding of how well the detection performed. Instead, a combination of metrics is used. Three of the most commonly used are precision, recall and F1 score. They are all based on relations between true positives (TP), true negatives (TN), false positives (FP) and false negatives (FN), and score between 0 and 1 where 1 is the highest score, see [32]

Precision is calculated as shown in Equation (2.4). It gives an idea of how trustworthy a predicted positive is without regard to how often it misses a true positive.

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{2.4}$$

Recall is calculated as shown in Equation (2.5). It shows how successful the detector is at finding all the true positives without regard to how often it detects a false positive.

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{2.5}$$

F1 score is the harmonic mean of precision and recall and is calculated as shown in Equation (2.6). It is used in a similar way to accuracy (all correct predictions divided by total predictions), but is often a superior metric to accuracy especially when working with imbalanced data in binary classification. In multi-class tasks there are several variations of calculating F1 score depending on how the individual class F1 scores are weighted. When dealing with highly imbalanced data, the *macro* F1 score is often used. It is simply the arithmetic mean of the F1 scores from all classes, as shown in Equation (2.7) where r is the number of classes, see [33].

$$\mathrm{F}_1 = 2 \cdot \frac{\mathrm{Precision} \cdot \mathrm{Recall}}{\mathrm{Precision} + \mathrm{Recall}} \tag{2.6}$$

$$\text{macro } \mathrm{F}_1 = \frac{1}{r} \sum_{i=1}^{r} F_{1i} \tag{2.7}$$

Average Precision (AP)

When comparing object detection algorithms average precision is often used as the main metric. The metric is, in slight contrast to its name, a combination of precision and recall. It is calculated as the area under the curve (AUC) of the precision plotted against the recall, see Equation (2.8) where $p(r)$ is the precision-recall curve. When plotting this curve the zig-zag pattern is often smoothed out by replacing the precision at each recall value with the maximum precision to the right of that recall value, see Figure 2.9. The zig-zag pattern is a result of the recall increasing when more TPs are found, but also FPs, which alters the precision.

$$\mathrm{AP} = \int_0^1 p(r)dx \tag{2.8}$$

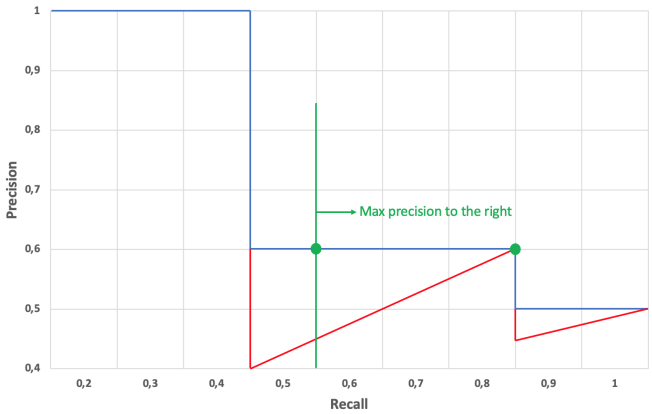


Figure 2.9: Precision-Recall curve illustrating which values are used in the calculation of AP. Adapted from [34].

There are several ways of calculating the AP. The modern approach is to sample the curve at all unique recall values (r_1, r_2,...) generated at points when the maximum precision drops during training. The AUC is calculated as the sum of all the rectangular blocks created by the sampling method, shown in Equation (2.9) where p is the precision as an interpolated function of the recall r. This approach results in measuring the exact AUC after having removed the zig-zag pattern. The best possible AP score is 1 and the worst possible score 0, see [34].

$$\mathrm{AP} = \sum (r_{n+1} - r_n) p_{\text{interp}}(r_{n+1}) \tag{2.9}$$

In order to get the precision-recall curve in the first place one needs to set an IoU threshold. The standard AP is generated from an IoU threshold of 0.5 and is often denoted as AP@0.5.

It is also common to calculate the average AP for a range of IoU thresholds, which can give a better AP performance insight when comparing models. This type of AP is denoted with a colon between the IoU ranges e.g. AP@0.5:0.95 which is a standard range to use. Sometimes AP is referred to as mean average precision (mAP) which is an average over several classes, but the expressions AP and mAP are used interchangeably depending on context.

2.2.8 Overfitting

When a model fits exactly to its training data, it is said to overfit to the data and does not perform well on unseen data, as it fails to generalise. Instead it starts to learn irrelevant and overly specific information, like noise, in the training data. If the training loss *decreases* as the validation loss *increases*, it is a strong indication that the model is overfitting, see [35]. To avoid this, the training process can be stopped earlier, known as "early stopping". If this method is used, it is important to not stop too early, since this could result in the opposite effect, that the model underfits. Because of this, the goal is to find the perfect time to stop training the model. The relation between the training and validation error for this perfect timing (the "sweet spot") is illustrated in Figure 2.10.

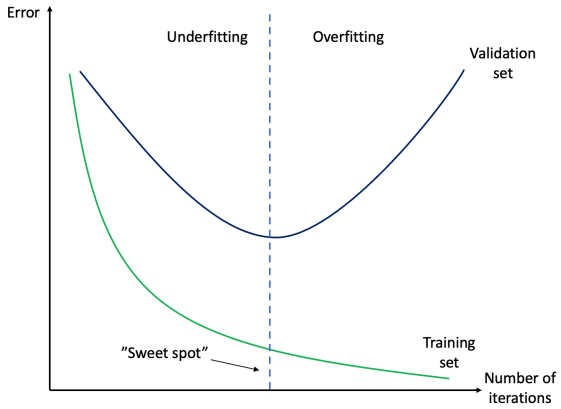


Figure 2.10: The "sweet spot" between under- and overfitting during training. Adapted from [35].

2.2.9 Convolutional Neural Networks

When dealing with images, Convolutional Neural Networks (CNNs) are widely used for object detection. It is a deep learning algorithm that has the ability to take an image as input, assign weights and biases to objects and/or aspects in the image and differentiate them. The convolutional neural network does not map images to vectors, as a feed forward network would. The most noteworthy operation in the algorithm is a special kind of linear operation called convolution. CNNs are basically a neural network which use convolution in at least one of their layers where normally a general matrix multiplication with a weight would occur, see [36]. An illustration of the general architecture of a convolutional neural network is shown in Figure 2.11.

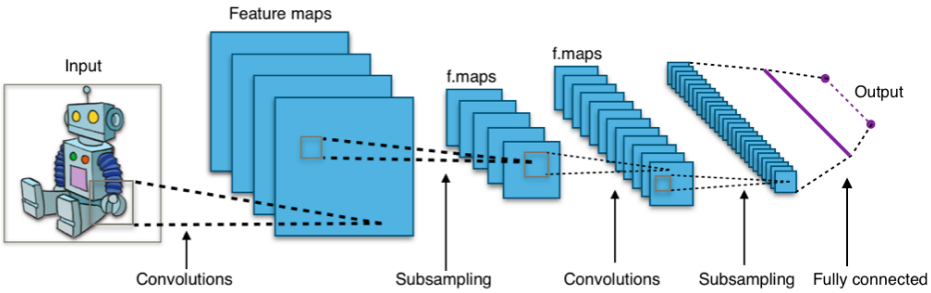


Figure 2.11: Example of a CNN pipeline from input image to output [36].

Convolution

In convolution, a two-dimensional array of weights called a filter or kernel is multiplied with the input data instead of a normal weight. The filter is intentionally smaller than the input data since it allows the same filter to "slide" over the entire input data as shown in Figure 2.12. The type of multiplication used is a dot product and therefore always returns a single number that can be mapped in a so-called feature map as a weighted sum. Each feature map will represent a special feature in the image, hopefully relevant to the problem at hand, such as edges, see [37].

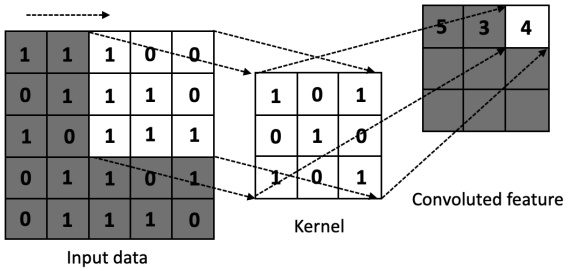


Figure 2.12: How a kernel operates sliding over input data resulting in a convoluted feature. Adapted from [38].

Pooling

When performing convolution the size of the problem can scale up very quickly. In addition to this, the outputs from the feature maps are very sensitive to the location of the features in the input. Making a small change of the location of a feature could create an entirely different feature map. Because of this, pooling is often used to down sample the feature maps, making them more robust to changes. Two commonly used pooling methods are average pooling and max pooling. Average pooling maps the average value in the current patch, while max pooling maps the maximum value of it, see Figure 2.13. The kernel size used for pooling is often 2x2, resulting in an output half the size of the input in each dimension, i.e. each feature map containing a quarter of the pixels from before the pooling, see [39].

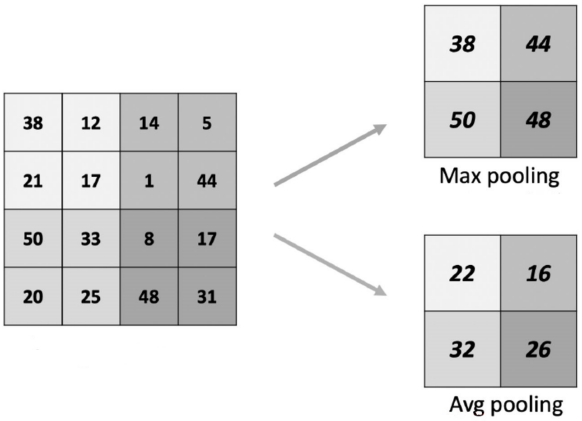


Figure 2.13: How max and average pooling operate on pixel data [40].

Stride and Padding
Sometimes a specific output size is required. If the size is to be decreased, setting the stride is a possible solution. When applying a filter in a neural network, such as pooling, the stride decides how many pixels the weight filter should move each step. The larger the stride, the smaller the output image. If the size reduction is to be minimised, padding could be applied. Padding is a feature that adds empty pixels (set to 0) to the outer border of the image, increasing the input image size thus counteracting the minimising effect the stride has, see [41].

2.3 Object Detection Models

Object detection models detect and classify objects within an image or video. The difference from classification algorithms is that object detection algorithms locate objects of interest with bounding boxes, and detect several objects within a single image, see [42]. There are several different object detection algorithms, often divided into two sub groups: *one-stage* and *two-stage* detectors. In general, one-stage detectors have a lower inference time while the two-stage detectors have higher accuracy. The most distinguishable difference between the two is that the two-stage detectors first find region of interest (RoI) proposals and then sort them out as real bounding boxes or not, while the one-stage detectors find the bounding boxes directly without any region proposals. One of the most outstanding two-stage detectors is the R-CNN and the most prominently used one-stage detector is YOLO.

2.3.1 Region Based Convolutional Neural Network (R-CNN)

R-CNN [43] is a deep convolutional neural network that was developed in 2014 by a group of researchers at UC Berkely. Since then it has been updated twice, resulting in fast R-CNN [44] and faster R-CNN [45], where the latter is the one most used today. R-CNN works by first finding region proposals, and then classifying them. How this is done differs

between the three mentioned versions of the R-CNN, but the state of the art one - the faster R-CNN - does it most efficiently. Both of the previous ones use selective search to find the regions. The creators of faster R-CNN, see [45], came up with a strategy to eliminate this time consuming method.

In the faster R-CNN network, images are provided as input to a convolutional network which produces a convolutional feature map. Instead of using a selective search algorithm on the feature map, as in fast R-CNN, an independent network is used to predict the region proposals. After this step, a RoI pooling layer is used to reshape the proposed regions and sends them to the classifier. The work flow of the faster R-CNN can be seen in Figure 2.14. By doing this modification of the R-CNN, the test time for the faster R-CNN is 8.7% of the fast R-CNN and only 0.4% of the first R-CNN, see [45].

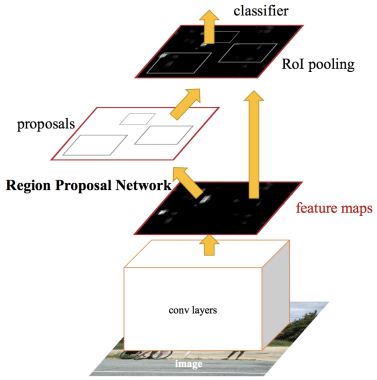


Figure 2.14: Faster R-CNN pipeline [45].

2.4 You Only Look Once (YOLO)

The object detection networks described in Section 2.3.1 are rather large and computationally heavy networks which results in slow training and detection as well as needing powerful hardware. As a counter reaction to the object detection community's trend of moving towards larger and more complex network, the paper "You Only Look Once: Unified, Real-Time Object Detection", see [46], was released describing an object detection model that can process images in real-time. The first version of the model, see Section 2.4.1) had speed but lacked in prediction performance compared to the more classic models described in Section 2.3.1. However, since its release in 2016 the YOLO model has undergone several improvements and is today amongst the state-of-the-art in object detection.

The core idea of YOLO is, as the name suggests, to both find objects in images and classify them whilst only "looking" at the image once. By "looking once" it is meant that bounding boxes and their subsequent class probabilities are predicted simultaneously, straight from pixel level data, by a single convolutional network. Apart from YOLO's approach making it significantly faster than previous detection methods, it also stands out as viewing the whole image means that it implicitly encodes contextual information about classes, thus minimising the number of background errors made.

2.4.1 YOLOv1

YOLOv1 is a single neural network with 24 convolutional layers followed by two fully connected layers, see Figure 2.15 for a structural break down. It takes input images of size 448×448.

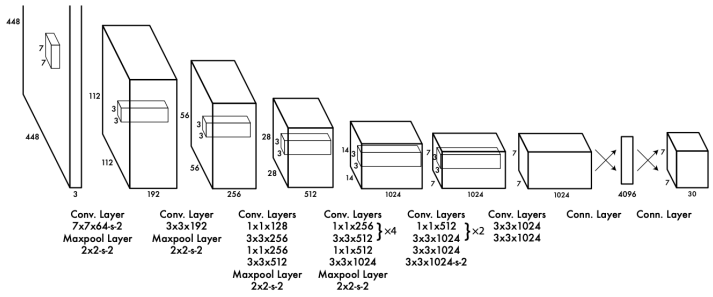


Figure 2.15: YOLOv1 architecture [46].

The input image is divided into an $S \times S$ grid. Each grid cell is responsible for detecting objects that fall within it. During training each cell will predict B bounding boxes, defined as centre coordinates, width and height scaled to image size (x,y,w,h) along with a confidence score. The confidence score is the product of the probability of a box containing an object, see Section 2.4.5, and its IoU with the ground truth, as defined in Equation (2.10). Parallel to predicting bounding boxes the YOLO network creates a class probability map, see Figure 2.16, predicting C conditional class probabilities ($Pr(Class_i|Object)$) per grid cell, where C is the number of classes, see [46]. This results in class specific confidence scores for each box, see Equation (2.11). During inference the confidence replaces the confidence scores, which is simply Equation (2.10) and (2.11) without the IoU, since ground truth is not available during inference.

$$\text{Conf score} = \Pr(\text{Object}) \cdot \text{IOU}_{\text{pred}}^{\text{truth}} \tag{2.10}$$

$$\text{Conf score}_i = \Pr(\text{Class}_\text{i}|\text{Object}) \cdot \Pr(\text{Object}) \cdot \text{IOU}_{\text{pred}}^{\text{truth}} = \Pr(\text{Class}_\text{i}) \cdot \text{IOU}_{\text{pred}}^{\text{truth}} \tag{2.11}$$

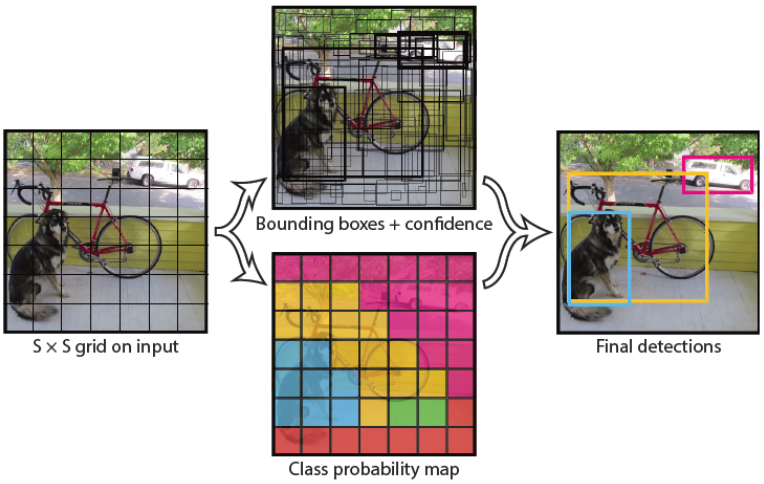


Figure 2.16: Object detection process in YOLOv1 [46].

The series of simplifications introduced in YOLOv1 naturally come with limitations and some cost in performance compared to region-based networks. Apart from the obvious downside of a small network having to use rather coarse features for prediction, the strict division of the images into grids which can only contain one class and a set number of bounding boxes makes it difficult to detect small objects that lie close to each other or overlap. The network also struggles to generalise to objects in new aspect ratios or configurations.

2.4.2 YOLOv2 - YOLOv5

Since the release of YOLOv1 it has undergone several updates with significant improvements: YOLOv2 introduced the concept of anchor boxes, see Section 2.4.3, YOLOv3 updated and modernised the architecture and YOLOv4 provided a plethora of improvement techniques such as more intense data augmentations. There are several more improvements than listed here which can be read in the versions' individual papers, see [47], [48], [49].

The latest version, YOLOv5, is built on PyTorch making it very easy to use. It is based on YOLOv3 but has similarities to YOLOv4. Its structure can be divided into three parts, see [50] and Figure 2.17:

- **Backbone**: Cross Stage Partial Network (CSPNet). A CNN that extracts informative features from the input images and reduces computations for improved processing time.
- **Neck**: Path Aggregation Network (PANet). This layer is an enhanced Feature Pyramid Network (FPN). A FPN is a CNN with two parts: First, a bottom-up structure that takes feature maps through levels of decreasing the spatial resolution. As the spatial resolution decreases the semantic value increases. Secondly, a top-down structure where the semantic information from the coarser upper map layers

is passed on to the layers with high resolution, thus combining the advantages of the top and bottom layers, see [51].

- **Head**: YOLO layer as described in section 2.4.1 where anchor boxes are applied and predictions made on the features fed from the neck. The YOLO layer uses ReLu activation in the hidden layers, sigmoid activation in the final layer and SGD as the optimiser. The head outputs the class predictions along with bounding box coordinates.

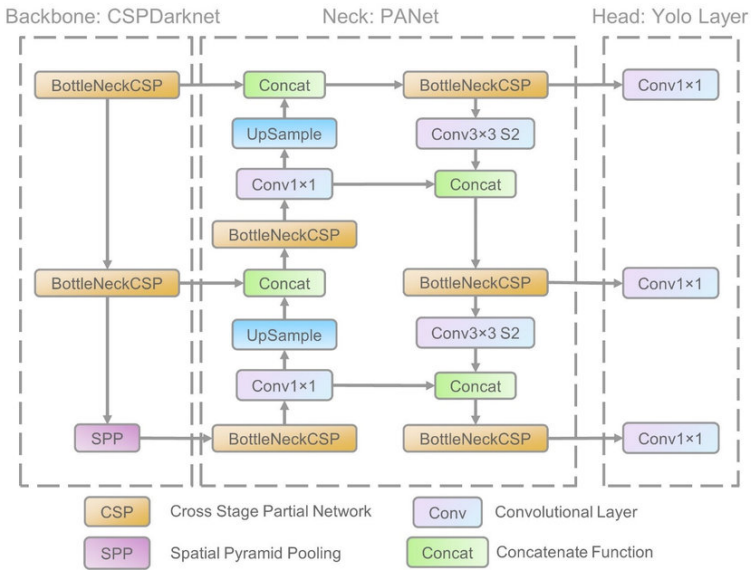


Figure 2.17: Simplified overview of YOLOv5 architecture [52].

The entire architecture of YOLOv5 is presented in Appendix A. Detailed explanations of some integral parts of YOLO are given in the sub-sections below.

2.4.3 Anchor Boxes

Anchor boxes are a way of improving speed and efficiency for predicting the bounding boxes in object detection tasks. They are a set of pre-defined boxes with a set height and width based on what objects are meant to be detected. In other words they are prior guesses on the bounding box shapes before the detection. For example, if an object detector is developed to find balls in images it would make sense to have square anchor boxes of varying size as there are very few balls that are not equal in height and width when enclosed by a bounding box. The network then predicts box probabilities and offsets along with attributes such as IoU compared to the ground truth. These predictions are used to fine tune the placement and dimensions of each anchor box, see [53].

In YOLOv5 anchor boxes are used, but their dimensions are based on clustering. The unsupervised learning method *k-means clustering* maps the ground truth bounding box dimensions and identifies clusters which are translated into anchor boxes. This has shown to be a better method than hand-picking anchor boxes. For the sake of model stability

the anchor boxes are not initialised randomly across the image. Instead, they are initially placed within grid cells similarly to how boxes are placed in YOLOv1, see [47] and Section 2.4.1.

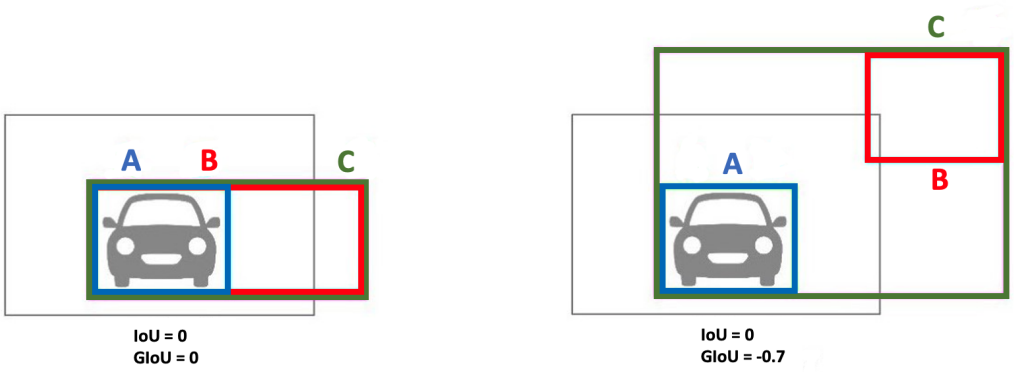
2.4.4 Loss Functions

YOLOv5 calculates three different losses: a box loss, an object loss, and when detecting more than one class, a class loss.

Generalised Intersection over Union

For calculating the box loss, i.e. the loss function that makes the object detector improve its predicted bounding boxes, the YOLOv5 repository uses Generalised Intersection over Union (GIoU). It is the IoU equation with an added term that adjusts for cases where there is no overlap between the predicted and ground truth boxes i.e. IoU = 0, see Equation 2.12. This term is the smallest convex hull (C) that encloses both A and B. As the prediction improves the area of C decreases thus minimising the subtracted term yielding a better GIoU score. If regular IoU was used the loss would not decrease/increase in cases where predictions have no overlap with ground truth despite the predictions moving closer to/further from it, see Figure 2.18. The authors of GIoU show in [54] that there is an analytical solution to Equation (2.12), making it possible to use as a loss function.

$$\text{GIoU} = \frac{|A \cap B|}{|A \cup B|} - \frac{|C/(A \cup B)|}{|C|} = \text{IoU} - \frac{|C/(A \cup B)|}{|C|} \tag{2.12}$$



(a) (b)

Figure 2.18: Illustration of how GIoU is calculated and different from IoU, where A, B and C are the same as for Equation 2.12. In (a) there is no overlap and C is equal to the union of A and B, yielding both IoU = 0 and GIoU = 0. In (b) there is no overlap but C is larger than the union of A and B. This yielding a IoU = 0 and a GIoU = -0.7. Adapted from [55].

Binary Cross-Entropy

For calculating object and class loss YOLOv5 uses binary cross-entropy. The loss is based on the calculation of entropy which is a measure of the uncertainty associated with a given

distribution $q(y)$, where C is the number of classes, see Equation (2.13). In the case of training a neural network the distribution $q(y)$ is available as the ground truth, but we want the network to estimate this distribution and continuously improve that estimation. For this, the equation for entropy can be altered to include an estimated distribution $p(y)$ yielding the so-called cross-entropy, see Equation (2.14). Comparing these it is clear that $H(q) = H_p(q)$ if $p(y) = q(y)$ i.e. the estimated distribution is equal to the true distribution.

Finally, in the case of binary cross-entropy the classifier needs to make the binary decision whether a given box should be labelled true or false, and therefore the distribution of false $(1-p(y_i))$ is added to the loss function, see Equation (2.15), where the score is an average over the N points/samples and i is the class, see [56].

$$H(q) = -\sum_{c=1}^{C} q(y_c) \cdot \log(q(y_c)) \tag{2.13}$$

$$H_p(q) = -\sum_{c=1}^{C} q(y_c) \cdot \log(p(y_c)) \tag{2.14}$$

$$H_p(q) = -\frac{1}{N}\sum_{i=1}^{N} y_i \cdot \log(p(y_i)) + (1 - y_i) \cdot \log(1 - p(y_i)) \tag{2.15}$$

2.4.5 Objectness

Object detectors produce box proposals in image areas they deem to possibly contain an object. It is therefore of interest to score the boxes based on how probable it is that they actually contain an object. This score is called objectness. In YOLOv5 it is based on the parameters *Multi-scale saliency*, *Colour contrast*, *Edge density* and *Superpixel straddling*, where the higher the score is in these underlying parameters the more probable a box is considered to contain an object, see [57].

Using objectness reduces detection time as it leads to an effective way of making relevant box proposals. At the same time it allows multi-class detection as it is a generalised algorithm not developed to work on a specific class.

2.4.6 Non-Maximum Suppression

Object detection algorithms output a large number of bounding boxes. These need to be sorted through in order to find the boxes that a) contain an object according to what one is trying to detect and b) best enclose that object. In order to do this an algorithm called NMS is used in YOLOv5. It is based on the confidence and IoU of each bounding box, see [58].

At its core the NMS algorithm is greedy and loops over all boxes for all classes. It takes the box with the highest confidence and removes it from the list of proposed boxes and then calculates the IoU score between it and every other proposed box. If the IoU is over

a set threshold the box with the lower confidence score is removed, see Equation (2.16). This process is then repeated with the list of remaining proposed boxes until it is empty, thus yielding a new list of boxes considered to be correct, see [59].

$$s_i = \begin{cases} s_i, & \text{IoU}(M, b_i) < N_t \\ 0, & \text{IoU}(M, b_i) \geq N_t \end{cases} \tag{2.16}$$

This algorithm is not perfect, for example it is highly time consuming ($O(n^2)$) and therefore other versions of the algorithm exist. The time issue can be bettered by removing box proposals with low confidence score before running the algorithm. Another issue is that it is poorly adapted to cases where two (or more) instances of the same class exist in the image and overlap. There is a high chance that their boxes will overlap enough for the IoU to be above the threshold resulting in only one instance of the class being detected.

2.4.7 The COCO Data Set

The COCO data set is a commonly used data set for training object detectors, and was the data set primarily used for evaluating YOLOv5 in the development phase. The data set consists of annotated images of complex everyday scenes with common objects. The composition is as follows:

- 328k images
- 2.5 million labelled instances
- 91 object classes

For a more extensive analysis of the data set and the composition of it, see [60].

2.4.8 Model Fitness

YOLOv5 uses a fitness function to evaluate how close the found solution is to the optimal solution. The function is used to decide for which epoch the model should be saved as the "best one". It takes the performance metrics *precision, recall, AP@0.5* and *AP@0.5:0.95* into account. The default setting is weighted so that the score is based 10% on the AP@0.5 and 90% on the AP@0.5:0.95.

2.4.9 Learning Rate

When updating the weights during back propagation it is of interest to control *how much* the weights are updated. This is done by controlling the step size, known as the learning rate. The learning rate is typically a small positive value between 0.0 and 1.0, which the estimated error is scaled with. It is important to choose an appropriate value since it controls how fast the model adapts to the problem. A smaller value would need more training epochs since it changes less for each epoch, while a larger value would adapt faster. Despite this, a large value could result in too fast an adaptation, making the model converge to a sub-optimal solution too quickly and oscillate around the optimal solution,

whereas a smaller value could cause the model to get stuck or just take very long time. An illustration showing the step size to find the minimum loss is shown in Figure 2.7. It is not possible to analytically calculate the optimal learning rate, thus it must be found trough trial and error.

Finding the Optimal Learning Rate
The learning rate will interact with several other parameters in the model and the interactions may not be linear. Smaller learning rates may need more training epochs and larger may need fewer. Smaller values have also shown to be better suited with smaller batch sizes, because of the noisy estimates from the gradient error. Traditionally the learning rate is set to a value between 10^{-6} and 1.0, but a good starting point seems to be 0.1 or 0.01.[61]

Learning Rate Schedule
An approach to be considered instead of using a fixed learning rate is to allow it to change over time. This is referred to as the learning rate schedule. The simplest schedule would be to initially have a large learning rate to allow large changes in the beginning and then linearly decrease it for fine-tuning. The schedule could also be chosen to decrease for a fixed number of epochs, and then fixed to give more time for fine-tuning, see [61]. In YOLOv5 two values are used, one denoted *lr0* that is the initial learning rate, and one denoted *lrf* that decides how much the learning rate should decrease per epoch. The final learning rate, for the last epoch, is then $lr0 \cdot lrf$. The default values in YOLOv5 are 0.1 and 0.01.

2.5 Improvement Techniques

There are several ways to improve the performance of networks. Some of the techniques, like augmenting the data, are used to generate more data from the existing data for the network to learn from, while some help the network to learn more efficiently or accurately. These different techniques could cause a longer training time or a higher memory allocation, but generally do not affect the inference speed for detection.

2.5.1 Image Augmentation

Neural networks are heavily reliant on the quality and amount of data available to train on. Contrary to this, medical data often is limited in its availability, especially when it comes to data annotated by professionals. It is therefore of interest to use this data as efficiently as possible. A way of doing this is with image augmentation. Image augmentation takes the existing images and applies transformations that alter the images thus generating new data from the network's perspective. To the human eye it can seem like augmented data adds very little new information, but even small changes can help improve object locating and minimise overfitting. Some of the augmentation techniques considered in this thesis are described below.

Mosaic

Mosaic is a data augmentation algorithm that mixes and puts several images next to each other in a single image. This is done to take the images out of their context, allowing the network to work on a different context in each layer, also reducing the batch size. This may also reduce the risk of overfitting and has been shown to give a higher AP for several detectors on the COCO data set, see [49].

HSV Augmentation

HSV, short for Hue-Saturation-Value, is a colour model aimed to be more logical to the way the human eye perceives colour compared to the traditional RGB (Red-Green-Blue) model. The model has a cylindrical geometry and is easiest to understand visually, see Figure 2.19. The conversion to HSV from the traditional RGB scale is rather complex, but is explained in [62]. When used for image augmentation the H, S and V values are each set to a range of how many per cent they are minimally/maximally to be altered. This results in images we perceive as lighter, darker or differently coloured.

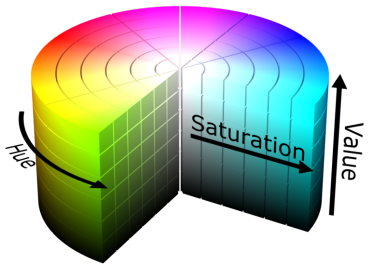


Figure 2.19: The HSV colour cylinder [63].

Positional Augmentations

There are several positional augmentations available. The ones used in this thesis are described in [64] as below:

- **Flip:** A flip can be made either horizontally or vertically. A flip horizontally is the same as a mirrored image of the original image. A vertical flip could be described as a 180 degree rotation followed by a horizontal flip. When added as an augmentation, the probability of the images flipping is added as the argument.
- **Shear:** Shear augmentation is an image transformation that slants the shape of the image and can be applied on the either the x- or y-axis. In other words, it transforms a rectangular image into a parallelogram. The argument added for shear is a +/- degree value.
- **Scale:** The image can be scaled inwards or outwards. If the image is scaled outwards the image size will be larger than the original, why a common thing to do is to cut out a part from the new image with equal size to the original image. When scaling inwards, resulting in a smaller image size, the framework makes an assumption about the image, filling the lack of information as it seem fitting. The scale is added as an augmentation based on a +/- gain value.

2.5.2 Focal Loss

The binary cross-entropy equation has a weakness when the network is unevenly confident in its predictions of different classes, as is often the case when working with an imbalanced data set. In these cases it may be advantageous for less confident class predictions to generate a much larger loss than is the case in binary cross-entropy, leading to training focusing more on improving these predictions instead of the already good predictions. A way of handling this was introduced in [65], presenting the so-called focal loss shown in Equation (2.17). It introduces both a term α that adjusts class weights and a term γ that regulates how much the predicted class probabilities, p_t, contributes to the loss. A large γ will alter the loss function so that less confident predictions (e.g. $p_t < 0.6$) contribute a lot more to the loss than more confident predictions (e.g. $p_t > 0.8$).

$$\mathrm{FL}(p_t) = -\alpha_t(1 - p_t)^\gamma \log(p_t) \tag{2.17}$$

2.5.3 Image Sizes

At CellaVision, it has previously been shown that a larger input image size increases the detection performance on PLTs. When larger original images are not available, the larger image size can be obtained by upsampling the image. Larger input image sizes would result in more pixels, letting the smaller objects become larger and therefore easier for the network to detect. During subsampling and convolutions, see Figure 2.11, some information is lost due to the mapping to feature maps. Having a larger image size yields more pixels and a lower risk of information loss. It is important to use the same image size when using it for detection as was used for training the network. To increase the image size is also suggested by the developer of YOLOv5 when having smaller object, see [66].

2.5.4 Class Weights

Class weights are used when having imbalanced data. They give all classes equal importance and prevent the network from predicting the more frequent class more often only because of it having most samples in the training data. During training, the total loss that is computed for each batch is replaced by a weighted sum, meaning that the samples contribute to the loss proportionally to the weight for the class of the sample, see [67]. One way of doing this is to assign weights for the classes inversely proportional to the frequency of the class, meaning the class(es) occurring less frequently contributes more to the total loss.

2.5.5 Image Weights

'Image weights' is used during training to give images with lower AP from previous training epochs higher likelihood to be selected in the next epoch of the training. In YOLOv5 the images are weighted by the inverse AP from the previous epoch according to the developer of YOLOv5, see [68].

2.6 Current Algorithms at CellaVision

The current PLT application at CellaVision has two separate algorithms for finding RBCs and PLTs. The RBCs are found using a method called segmentation, where the algorithm assigns each pixel a class and therefore both finds the object and the shape of it. The algorithm can only find single cells if they do not touch another cell in the image, i.e. when pixels of one class are completely surrounded by pixels of another class. If the cells are overlapping, the algorithm instead finds the aggregation and estimates how many RBCs are in it based on the size of the aggregation. The detached cells are classified, based on their size, as RBCs, PLTs, WBCs or discardable. All cells not classified as RBCs are not included in the analysis. The PLTs on the other hand are found using a YOLOv5 object detector, with a model called *nano v6* which is available in the YOLOv5 repository. The image size trained on is 1088×704. The average combined inference time for the current algorithms, based on 30 images, is 300ms/image where 211ms is for the segmentation algorithm and 79ms for the PLT detection.

Since two separate algorithms are used, they can be evaluated separately. Henceforth, the current methods used in CellaVision's systems will be referred to as RBC_CV and PLT_CV.

3 Method

3.1 Data Gathering

When starting this thesis project, a large data set of blood smear images with PLTs annotated as the result of an existing PLT detector at CellaVision was available, however no equivalent data set existed for RBCs. Since the PLT annotations as well as the underlying images were a great asset to this project, those images were decided to be used as our data set. To not allocate to much memory on the server, these images were down sampled to half of the original size (1920×1200), resulting in images with the size 960×600. It was immediately clear that there was not enough time to manually annotate the RBCs in all these image. Therefore, an approach to train a good enough RBC detector to, within a reasonable amount of time, pre-annotate the RBCs in these images and then complete missed/incorrect annotations manually was initiated.

An iterative approach was used where images were manually annotated, and then networks trained on these images used to pre-label the remaining images more quickly and accurately. This process was repeated until the training, validation and test set were large enough to start evaluating RBC networks. Once all test, validation and a fraction of the training images had been labelled, an attempt was made to run detection on both RBCs and PLTs simultaneously. Since the data set did not seem to be large enough for accurate PLT detections, more RBCs needed to be annotated to be able to use more of the labelled PLT data set. Because of the cumbersome task of manually completing the training set annotations, a pseudo labelling approach was considered to complete the RBC annotations. This approach would also be an interesting topic of discussion since there is no shortage of blood smear images at CellaVision in contrast to the amount of manually annotated images. Successful pseudo labelling techniques would mean the possibility of training future networks on very large amounts of data.

3.1.1 Pseudo Labelling RBCs

In order to develop the best network for pseudo labelling RBCs in the remaining training image data, different parameters and models were evaluated and later combined based on performance. The metrics used for evaluating performance were F1 score, AP@0.5 and AP@0.5:0.95. Inference time was not considered as the pseudo labelling network was not intended for use on CellaVision systems. As a base, the networks explored were pre-trained with weights from the COCO data set with the *low* augmentation settings. The parameters explored were image sizes, YOLOv5 model size and image augmentations. The image size used (unless exploring more image sizes) was 960×608 since image dimensions need to by dividable by 32 in YOLOv5.

In the end, a YOLOv5 *small* model pre-trained on the COCO data set operating on the image size 960×608 with the *high* level of augmentation from YOLOv5's options was

chosen as it had the highest score for all three performance metrics. This network was then used to run detection on all the remaining unannotated images and the subsequent bounding box coordinates for each image were saved. To ensure that all images had been labelled, one image showing the detected boxes from each smear image region were viewed manually. The smallest annotation data files from the detection were also checked to have a reasonable amount of RBC annotations, ensuring that the network had not failed to detect in any of the images. The RBC annotation files were then merged with the corresponding PLT annotations, ultimately forming a data set with both manually and pseudo labelled RBCs and manually labelled PLTs.

In order to evaluate the usefulness of pseudo labelling, three networks were run on one training set containing manually labelled data only and on one training set containing manually and pseudo labelled data. One network detecting only RBC, one detecting only PLT (no pseudo labelled data here but it gives a comparison of the effect more data has) and one detecting both classes. Since the network including pseudo labelled data improved PLT detection performance without decreasing RBC detection performance, the labelled + pseudo labelled training set was decided to be the training set used going forward with the project.

The composition and analysis of the final data set can be viewed in Section 4.

3.2 Testing Individual Parameters

Once the training data set had been established, the development of the RBC + PLT detector was initiated. The approach used was to first test different models and hyper parameters individually to get an idea of which factors have the largest positive impact on performance. All models were trained for 300 epochs and evaluated using an algorithm described in Section 3.2.1. The idea was to later combine the parameters in the hope of further improvement. The IoU threshold between predicted and ground truth boxes used when evaluating models at this stage was set to 0.3.

When evaluating performance for the individual parameters, precision, recall and F1 score on the validation set were considered. The networks were evaluated for RBC + PLT detection as well as for individual performance on RBCs and PLTs. This was done to be able to compare the found detector to existing RBC and PLT detection algorithms/networks at CellaVision and for the sake of identifying imbalances and weaknesses in our network. For example, as the training set was heavily imbalanced, see Figure 4.2) a very good RBC detection would mean a very good overall performance even if the detection on PLTs performed much worse. At this point, AP@0.5 and AP@0.5:0.95 were not considered, since it would be of more importance for CellaVision to find cells, and find the right cells, than for the findings to have higher IoU scores than 0.5. Instead precision, recall and F1 score for the confidence where the networks had their maximum F1 score during training were used.

3.2.1 Performance Evaluation Algorithm

CellaVision's systems capture small images that later on are put together to form a larger image. The images are put together with an overlap, meaning the cells detected in the edges are often only partially visible and are not used since the entire cell is visible in another image. The data set in this thesis was made up of these small images, and it was clear early on that performance was significantly lower near the edges. Therefore the precision, recall and F1 score were calculated only including cells that did not touch the edges to give a fairer picture of how performance on CellaVision's system would be. This is also how the performance is evaluated for the networks used by CellaVision today.

The performance metrics calculations were made with a custom algorithm. After models had finished training and the best model selected according to the fitness function, see Section 2.4.8, a detection on the validation set was run for the confidence score that had given the highest F1 score during training. These detections were evaluated against the ground truth. For RBCs and PLTs the precision, recall and F1 score were calculated for each class individually as well as combined. The macro F1 score between the classes was also calculated.

The performance on detecting large and giant PLTs were not considered for the validation set since there were very few instances of each class, especially giants, in the set (673 large and 15 giant, see Figure 4.3.

3.2.2 Model Sizes

The YOLOv5 repository comes with several different models differing in network size. A larger network should be able to handle more information and, in theory, perform better than smaller networks. On the other hand, smaller networks often have significantly lower training and inference time. To investigate the effect model size had on the data, the models named *nano, small, medium* and *large* were tested, without any pre-trained weights or augmentations other than mosaic.

Based on their performance and low training and inference time, testing proceeded only using the *nano* and *small* model sizes.

3.2.3 Pre-Trained Weights

It is possible to train networks using the best weights of an already trained network. The creators of YOLOv5 have trained the network on the COCO data set and made these weights available in the YOLOv5 repository. To see if the performance would be affected by these or other pre-trained weights, the ones from the YOLOv5 repository, the best PLT detector at CellaVision and a PLT network created by us were tested. The latter came from training the network from Section 3.1.1 on PLTs instead of RBCs. This network is called PLT_SoTi for the remainder of the report.

3.2.4 Augmentations

The performance slightly increased for the RBC detector used for pseudo labelling when adding image augmentations. It was also clear that when adding more data to the training set, i.e. the data where RBCs had been pseudo labelled, the performance increased. Since augmentations yield more data, augmentations for our combined detector was evaluated at this early stage. The augmentations tested were three augmentation settings files provided by the YOLOv5 repository: *low, medium* and *high*, as well as the augmentations used to train PLT_CV.

3.3 Testing Combined Parameters

Before leaving the phase of testing individual parameters, higher IoU thresholds of 0.5 and 0.7 were tested. This resulted in slightly lower performance, but did not change which parameter settings had performed best. Together with our supervisors at CellaVision, an IoU of 0.5 was decided to be used in further tests. This because whilst it is important to find all the cells, it is still important that they are boxed well enough to be able to make further analyses.

When testing individual parameters to see which ones had most influence on the performance, it was clear that different types of augmentation affected the performance most. Because of this, further augmentation testing was prioritised. Since previous testing did not show a better performance for the larger models, the upcoming improvements were decided to be tested on the *small* model size. The *small* model does not take as long time as the *medium* or *large* models to train, but it is still more complex than the *nano* network and should be able to make use of stronger augmentations and hyper parameters. In addition to this, since the networks seemed to benefit from using pre-trained weights in form of how fast the network learned in the first epochs, there was no obvious downside to using pre-trained weights when moving on with the augmentations. The COCO weights were used as the PLT_CV weights were only applicable for a *nano v6* model, as mentioned in Section 3.2.3. Additionally the COCO weights had better overall performance compared to the weights from our network trained on PLTs only.

3.3.1 Further Augmentations

To find the optimal augmentation types and settings, the optimal HSV augmentation values was first sought. Values in between those used in two best performing models during the latest augmentation tests were used as a starting point. This resulted in setting the HSV augmentation values to (0.01, 0.4, 0.4). These settings gave equal F1 scores to PLT_CV's augmentations. For the sake of exploring alternatives to what was already used at CellaVision we proceeded using the HSV settings (0.01, 0.4, 0.4) and kept these for all future networks.

Next we investigated positional augmentations. The flip up/down probability was set to 0.5 in two networks based on its presence in the augmentation files used in Section 3.2.4.

In one of these network an image shear set to 0.1 was also added in hopes of compensating for the fact that cells are not uniformly shaped.

3.3.2 Continued Testing of Hyper Parameters

Since the PLT detection performance, especially the recall, seemed most difficult for the networks to increase, improving these scores was focused on going forward. As the fitness function used so far was strongly weighted towards AP and not precision and recall, see 2.4.8, how changing the weights of the fitness function would influence the results was considered. The weights were changed to be based 50% on the recall, 40% on the precision and 10% on the AP@0.5:0.95. These weights were kept for all further networks.

Finally, a number of hyper parameters and settings available in YOLOv5 were tested. The parameters were chosen based on their availability, use at CellaVision and recommendations in machine learning forums. They are all listed below and the subsequent combinations of them presented in Table 3.1:

- Optimiser = 'Adam'
- 'Image weights' = True
- Anchors = 5 or 7
- Class Weights [RBC, PLT] = [1, 20]
- Focal loss ($\gamma = 1.5$)

Table 3.1: Different hyper parameter combinations tested.

Test no	Flip	Shear	Fitness	'Adam'	Img w.	Anchors	Cls w.	Focal loss
1	✓							
2	✓	✓						
3			✓					
4			✓	✓				
5			✓		✓			
6			✓			5		
7			✓			7		
8			✓				✓	
9			✓					✓
10			✓		✓	5		

3.3.3 Image Size

Because of the uneven ratio between PLTs and RBCs, extra focus was put on improving PLT detections. With support from section 2.5.3, increasing the image size could increase the performance. Because of this two networks where image size was increased to 1152×720 (120%) and 1344×840 (140%) were tested.

3.3.4 Final Network Selection

All results were evaluated and the best performing network based on the macro F1 score was selected. At this point we concluded the exploration of networks and moved on to evaluating performance on the test set. In the remainder of the thesis this network, called '5' in Table 3.1, will be referred to as YOLOv5_SoTi.

3.4 Final Testing

To evaluate our network without bias, the final network selected was tested on a test set, meaning a set that has never been seen by the networks. Since there was an indication of larger image sizes resulting in better performance for PLTs and considering it being successful at the company before, the two bigger image sizes (120% and 140% of the image size 960×608) were also evaluated on the test set. The performance metrics were calculated as described in 3.2.1. The inference times for the networks with different image sizes were tested with the help of our supervisor at CellaVision.

3.4.1 Testing Subsets

To investigate if YOLOv5_SoTi was performing better or worse on any type of stain, the test set was divided into the three subsets with the different stains: WG, MGG and Wright. Also, slides containing deviating cell types, according to morphology experts at CellaVision, were divided into three subsets based on the presence of: Giant PLTs, Abnormal RBCs (based on shape/size/colour) and Malaria/parasites.

3.5 Current Algorithms at CellaVision

For the sake of reference, the current detection algorithms/networks for RBCs and PLTs at CellaVision were evaluated. As PLT_CV is also based on YOLOv5 it was evaluated in the same way as the networks developed during this thesis, described in section 3.2.1. In addition to this, precision, recall and F1 score for large and giant PLTs was also evaluated. RBC_CV on the other hand is a segmentation algorithm used for counting and was therefore not evaluated using precision, recall or F1 score. The number of RBCs from our network as well as from the current segmentation algorithm were compared with the ground truth number of RBCs in each region of the slides. All cells touching image edges were neglected at this point as in every other performance evaluation. A visual comparison of the detections was also made to be able to evaluate strengths and weaknesses among the different methods.

4 Data Set Analysis

Despite there being a plethora of models, hyper parameters and techniques for improving results in machine learning tasks, the most important factor is the data. To understand why a network performs as it does it is important to understand the underlying data. Reports often neglect this, ignoring important aspects of the data set used such as variability and imbalance. Therefore, a break-down and simple analysis of the data used will be presented. The analysis will show aspects of the data which has been considered when developing the models presented in Section 5.

The images used to train, validate and test our models were were captured from slides collected from different haematology laboratories in different countries using different staining techniques. The stains used were either May Grünwald Giemsa, Wright Giemsa or Wright (used in 50%, 26% and 3% of the slides, respectively). The remaining 21% of the slides had one of these stains, but were not marked with which one. The slides were prepared at laboratories in Sweden, Denmark, the Netherlands, New Zealand, Belgium, Canada or the US. All images were colour images in the BMP format, originally of size 1900×1200, but down sampled to 960×600. Due to YOLOv5 needing dimensions to be dividable by 32, the images were padded to an image size of 960×608 during training and testing.

The training data consists of 3880 images (from 168 slides) taken from different regions on blood smear slides. The slides had varying number of regions, and the regions varied in size meaning there was a varying number of images from each region in each slide. A few background images, i.e. images from regions not containing cells, were added to the training set following YOLOv5 guidelines. The training set's composition is shown in Figure 4.1. The validation and test sets were manually labelled and split into 481 (11 slides) and 948 (22 slides) images respectively . The ratio between RBCs and PLTs in the different sets was very similar (approximately 20:1) but with the training set containing a slightly higher percentage of PLTs, see Figure 4.2. A similar breakdown for PLT sizes is shown in Figure 4.3, which also shows similar balance between the categories for the three data sets.

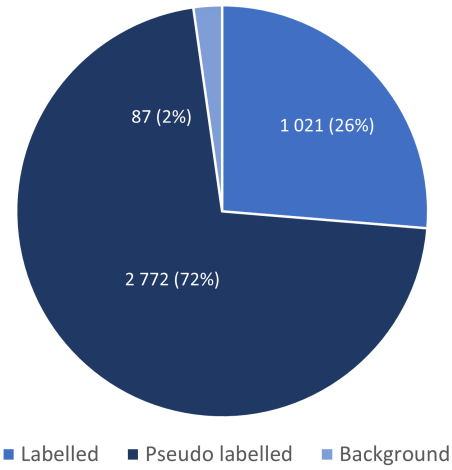


Figure 4.1: The ratio between labelled, pseudo labelled and background images in the training set.

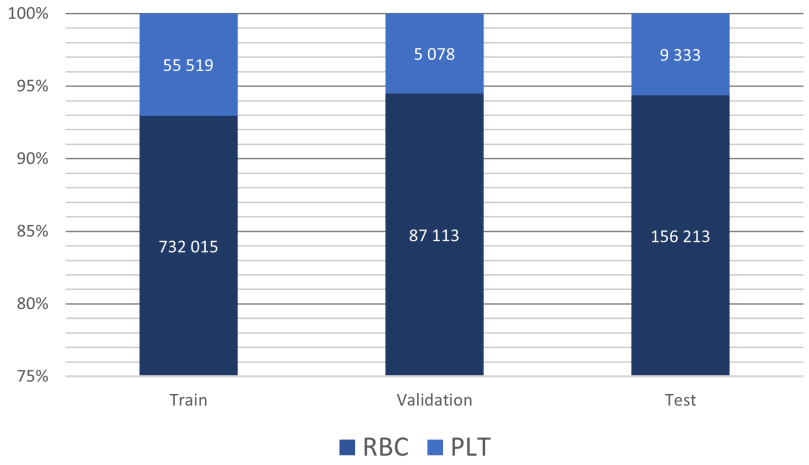


Figure 4.2: The number of RBC and PLT annotations and the ratio between them in each data set.

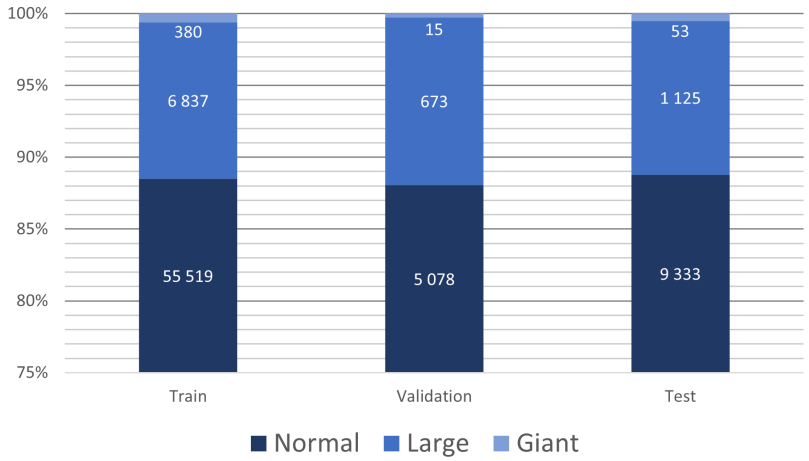
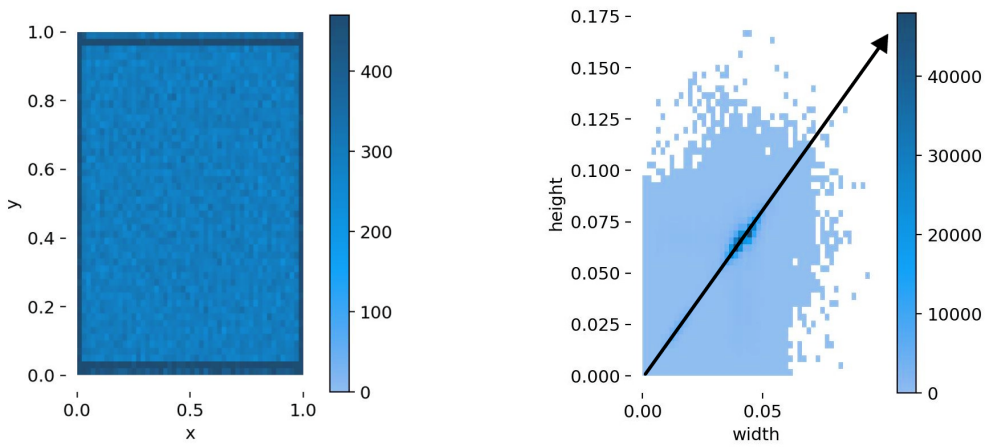


Figure 4.3: The number of normal, large and giant PLTs and the ratio between them in each data set.

The bounding boxes in the training set are well distributed over all coordinates of the images as can be seen in Figure 4.4a. The vast majority of these boxes are square and similar in size. Figure 4.4b shows an accumulation of boxes around 0.05% of image width and 0.07% image height representing normal RBCs, and a barely visible accumulation around 0.015% width and 0.025% representing PLTs. It is also clear from Figure 4.4b that there is a large variation in box dimensions although they are not frequent.



(a) (b)

Figure 4.4: Ground truth bounding box heat map histograms. Colour bars indicate number of boxes. (a) shows box locations in the images. (b) shows box dimensions as a percentage of image width and height. Arrow shows when the boxes' pixel width = pixel height, due to the fact that the underlying images are not square.

5 Results

The results are divided into four main sections. The first, Section 5.1, shows the results from developing the network used for pseudo labelling RBCs. The second, Section 5.2, shows the results from developing the network that detects both RBCs and PLTs. The third, Section 5.3, shows the results from using YOLOv5_SoTi on the test data set. The final one, Section 5.4, shows the results of separately comparing YOLOv5_SoTi to the current algorithms at CellaVision.

5.1 RBC Model Selection

This section presents results from developing the RBC detection network used for pseudo labelling RBCs. The final table shows a comparison between training without contra training with pseudo labelled data. All networks presented in this section were evaluated on the validation set.

5.1.1 Image Sizes

The *nano* YOLOv5 model pre-trained with the COCO weights and *low* augmentation settings trained on 20%, 40%, 60%, 80%, 100%, 120% and 140% of the image size 960×608. Smaller image sizes resulted in worse performance across all metrics whilst increasing image size did not improve performance except for AP@0.5, see Table 5.1.

Table 5.1: YOLOv5n model trained on different image sizes and evaluated on the image size 960×608, for RBCs only.

Image size	F1 score	AP@0.5	AP@0.5:0.95
192×128	0.876	0.906	0.626
384×256	0.958	0.972	0.771
576×352	0.969	0.981	0.791
768×480	**0.970**	0.981	0.789
960×608	**0.970**	0.982	**0.802**
1152×736	**0.970**	**0.983**	**0.802**
1344×832	0.967	0.981	0.777

5.1.2 Model Sizes

The performance of different YOLOv5 model sizes pre-trained with the COCO weights and *low* augmentation settings trained on the image size 960×608. It shows that the *small* network had the highest F1 and AP@0.5 score while the *medium* network had the highest AP@0.5:0.95 score, see Table 5.2.

Table 5.2: Different YOLOv5 models trained on the image size 960×608, for RBCs only.

Model	F1	AP@0.5	AP@0.5:0.95
YOLOv5n	0.970	0.982	0.802
YOLOv5s	**0.973**	**0.986**	0.806
YOLOv5m	0.971	0.985	**0.810**
YOLOv5l	0.966	0.975	0.783

5.1.3 Augmentations

Three different levels of augmentations from the YOLOv5 repository tested on the *small* model size pre-trained with the COCO weghts and trained on the image size 960×608. The *low* and *high* augmentations are provided in the YOLOv5 repository. The *custom* settings were the same as *low* but with HSV augmentations set to (0.4, 0.25, 0.25). The *high* augmentation level had the best or equal best score across all metrics, see Table 5.3.

Table 5.3: YOLOv5 *small* model trained on the image size 960×608 with different levels and types of augmentations.

Model	F1	AP@0.5	AP@0.5:0.95
low	**0.973**	0.986	0.806
high	**0.973**	**0.987**	**0.812**
custom	0.968	0.978	0.788

5.1.4 Evaluation of Pseudo Labelling

Table 5.4 shows results from YOLOv5 *small* networks with COCO weights and the *high* augmentation settings trained on the image size 960×608. The performance on PLTs clearly increased, whilst the performance on RBCs did not decrease for any metric when training on the manually + pseudo labelled data set.

Table 5.4: YOLOv5 *small* networks trained on the image size 960×608 with different training sets and detection classes. The data set including pseudo labels contains both manually and pseudo labelled RBCs, but only manually labelled PLTs.

Class	Training set	F1	AP@0.5	AP@0.5:0.95
RBC	Manually labelled	0.973	**0.986**	0.806
	Manually + Pseudo labelled	**0.974**	**0.986**	**0.815**
PLT	Manually labelled	0.944	0.968	0.657
	Manually + Pseudo labelled	**0.954**	**0.981**	**0.698**
Both	Manually labelled	0.953	0.969	0.719
	Manually + Pseudo labelled	**0.962**	**0.982**	**0.737**

5.2 RBC + PLT Model Selection

This section presents results from developing a network detecting both RBCs and PLTs. In total, the results of six different tests are shown. All networks presented in this section were trained on the training data set containing pseudo labelled RBCs and evaluated on the validation set.

5.2.1 Model Sizes

Table 5.5 shows the performance for different YOLOv5 model sizes without pre-trained weights or augmentations trained on the image size 960×608. For all three classes the *nano* model size performed best or equal best based on F1 score. Detection on RBCs showed both precisions and recalls over 0.98 whereas recall for PLTs was much lower.

Table 5.5: Different YOLOv5 model sizes trained on the image size 960×608 without pre-trained weights or augmentations.

Class	Model	Precision	Recall	F1(@IoU 0.3)	Macro F1
RBC	YOLOv5n	0.982	**0.982**	**0.982**	
	YOLOv5s	**0.983**	0.980	0.981	
	YOLOv5m	0.982	**0.982**	**0.982**	
	YOLOv5l	0.982	**0.982**	**0.982**	
PLT	YOLOv5n	0.960	**0.842**	**0.897**	
	YOLOv5s	**0.974**	0.811	0.885	
	YOLOv5m	0.970	0.777	0.863	
	YOLOv5l	0.967	0.796	0.873	
All	YOLOv5n	0.980	**0.973**	**0.977**	**0.940**
	YOLOv5s	**0.982**	0.970	0.976	0.933
	YOLOv5m	**0.982**	0.970	0.976	0.923
	YOLOv5l	0.981	0.970	0.976	0.928

5.2.2 Pre-trained Weights

Table 5.6 shows the results for models without augmentations with pre-trained weights from the COCO data set, PLT_SoTi, and PLT_CV trained on the image size 960×608. The weights from the COCO data set as well as from PLT_SoTi are applied to a *small* YOLOv5 while the PLT_CV are applied to a *nanov6* model. The overall F1 score was equal between the COCO and PLT_CV weights, with the former having the best F1 for RBCs and the latter the best F1 for PLTs, see Table 5.6. Using the PLT_CV weights yielded the best Macro F1 score.

Table 5.6: YOLOv5 *small* and *nano v6* models trained on the image size 960×608, pre-trained with different weights without any augmentations.

Class	Pre-trained weights	Precision	Recall	F1(@IoU 0.3)	Macro F1
RBC	COCO	**0.985**	0.982	**0.983**	
	PLT_SoTi	0.984	**0.983**	**0.983**	
	PLT_CV	0.982	0.982	0.982	
PLT	COCO	0.967	0.830	0.893	
	PLT_SoTi	**0.977**	0.766	0.858	
	PLT_CV	0.960	**0.885**	**0.921**	
All	COCO	**0.984**	0.972	**0.978**	0.938
	PLT_SoTi	**0.984**	0.969	0.976	0.921
	PLT_CV	0.980	**0.976**	**0.978**	**0.951**

5.2.3 Image Augmentations

Table 5.7 presents results from different image augmentation settings evaluated on the *nano* model size without pre-trained weights and trained on the image size 960×608. The best overall F1 score came from the *medium* augmentation settings and the Macro F1 scores were equal for *medium*, *high* and PLT_CV.

Table 5.7: YOLOv5 *nano* model trained on the image size 960×608 with different image augmentations.

Class	Augmentation	Precision	Recall	F1(@IoU 0.3)	Macro F1
RBC	Low	0.984	**0.982**	0.983	
	Medium	**0.990**	**0.982**	**0.986**	
	High	0.987	**0.982**	0.985	
	PLT_CV	0.986	**0.982**	0.984	
PLT	Low	0.952	0.945	0.949	
	Medium	0.959	0.950	0.954	
	High	**0.967**	0.945	**0.956**	
	PLT_CV	0.959	**0.952**	0.955	
All	Low	0.982	**0.980**	0.981	0.966
	Medium	**0.988**	**0.980**	**0.984**	**0.970**
	High	0.982	**0.980**	0.983	**0.970**
	PLT_CV	0.984	**0.980**	0.982	**0.970**

5.2.4 Further Image Augmentations

Different augmentations applied to a YOLOv5 *small* model with pre-trained weights from the COCO data set trained on the image size 960×608. It should be noted that the IoU threshold here has been raised from 0.3 to 0.5 compared to previous testing. The PLT_CV augmentation settings and our HSV augmentations gave equal best F1 and macro F1 scores across all classes, see Table 5.8.

Table 5.8: YOLOv5 *small* model trained on the image size 960×608, with pre-trained weights from the COCO data set with different image augmentations.

Class	Augmentation	Precision	Recall	F1(@IoU 0.5)	Macro F1
RBC	PLT_CV_aug	0.989	**0.983**	**0.986**	
	HSV	0.989	**0.983**	**0.986**	
	HSV+flip	**0.991**	0.982	**0.986**	
	HSV+flip+shear	0.986	0.982	0.984	
PLT	PLT_CV_aug	0.970	0.953	**0.961**	
	HSV	0.967	**0.955**	**0.961**	
	HSV+flip	**0.975**	0.936	0.956	
	HSV+flip+shear	0.974	0.936	0.955	
All	PLT_CV_aug	0.988	**0.981**	**0.984**	**0.973**
	HSV	0.988	**0.981**	**0.984**	**0.973**
	HSV+flip	**0.990**	0.979	**0.984**	0.971
	HSV+flip+shear	0.985	0.979	0.982	0.969

5.2.5 Continued Testing of Hyper Parameters

Performance from adding/changing different hyper parameters to a YOLOv5 *small* model pre-trained with the COCO weights with HSV augmentations and the new fitness function, trained on the image size 960×608. Adding only 'Image weights' yielded the best macro F1 and PLT F1 scores, and equal best overall F1 scores, see Table 5.9.

Table 5.9: YOLOv5 *small* model trained on the image size 960×608, pre-trained with the COCO data set with different hyper parameters and hyper parameter settings added.

Class	Parameter(s)	Precision	Recall	F1(@IoU 0.5)	Macro F1
RBC	Fitness function	**0.991**	0.983	**0.987**	
	Optimiser	0.988	**0.986**	0.985	
	Image weights	0.990	0.983	0.985	
	Anchors = 5	**0.991**	0.983	**0.987**	
	Anchors = 7	0.988	0.983	0.985	
	Class Weights	0.988	0.983	0.985	
	Focal loss	**0.991**	0.983	**0.987**	
	Image weights & anchors = 5	0.989	0.983	0.986	
PLT	Fitness function	0.964	0.962	0.963	
	Optimiser	0.964	0.940	0.952	
	Image weights	0.970	**0.968**	**0.969**	
	Anchors = 5	0.967	0.964	0.965	
	Anchors = 7	**0.972**	0.955	0.963	
	Class Weights	0.955	0.937	0.946	
	Focal loss	0.964	0.959	0.962	
	Image weights & anchors = 5	0.969	0.964	0.967	
All	Fitness function	0.989	**0.982**	**0.985**	0.975
	Optimiser	0.986	0.980	0.983	0.969
	Image weights	0.989	**0.982**	**0.985**	**0.978**
	Anchors = 5	0.989	**0.982**	**0.985**	0.976
	Anchors = 7	0.987	0.981	0.984	0.974
	Class Weights	0.986	0.980	0.983	0.966
	Focal loss	**0.990**	0.981	**0.985**	0.974
	Image weights & anchors = 5	0.988	**0.982**	**0.985**	0.976

5.2.6 Image Sizes

YOLOv5_SoTi trained and tested on image sizes 1152×736 and 1344×832 (120% 140% of the image size 960×608). The image size 960×608 performed best on RBC, overall and Macro F1 scores, see Table 5.10. The larger image sizes had the equal best F1 score for PLTs.

Table 5.10: YOLOv5_SoTi trained and tested on different image sizes, on the validation set.

Class	Image Size	Precision	Recall	F1(@IoU 0.5)	Macro F1
RBC	960×608	**0.990**	**0.983**	**0.985**	
	1152×736	0.985	**0.983**	0.984	
	1344×832	0.985	**0.983**	0.984	
PLT	9604×608	0.970	0.968	0.969	
	1152×736	**0.972**	0.969	**0.971**	
	1344×832	0.970	**0.972**	**0.971**	
All	960×608	**0.989**	**0.982**	**0.985**	**0.978**
	1152×736	0.985	**0.982**	0.983	0.977
	1344×832	0.984	**0.982**	0.983	0.977

5.3 Final Testing

This section presents results from testing the network that performed best on the validation data, YOLOv5_SoTi, on the test set or subsets of it.

5.3.1 Image Sizes

YOLOv5_SoTi trained and tested on image sizes 960×608, 1152×736 and 1344×832. The image size 960×608 performed best on RBC, overall and Macro F1 scores, see Table 5.10. The image size 1344 had the best F1 score for PLTs and a higher PLT recall than the image size 960×608, see Table 5.11. The inference time was significantly higher for the larger image sizes.

Table 5.11: YOLOv5_SoTi trained and tested on different image sizes, on the test set.

Class	Image Size	Precision	Recall	F1(@IoU 0.5)	Macro F1	Inference time
RBC	960×608	**0.989**	**0.983**	0.**986**		
	1152×736	0.985	**0.983**	0.984		
	1344×832	0.985	**0.983**	0.984		
PLT	960×608	**0.961**	0.957	0.959		
	1152×736	0.954	0.964	0.959		
	1344×832	0.951	**0.971**	**0.961**		
All	960×608	**0.987**	0.981	**0.984**	**0.972**	**150ms/image**
	1152×736	0.983	**0.982**	0.982	0.971	220ms/image
	1344×832	0.982	**0.982**	0.982	**0.972**	250ms/image

5.3.2 Stains

YOLOv5_SoTi tested on subsets containing the images with the stains WG, MGG and W. Performance was best or equal best on the MGG set for all F1 scores, especially for PLT F1 and Macro F1, see Table 5.12.

Table 5.12: YOLOv5_SoTi performance on subsets with different stains.

Class	Test subset	Precision	Recall	F1(@IoU 0.5)	Macro F1	Images (n)
RBC	W	**0.994**	0.978	**0.986**		
	WG	0.988	0.981	0.979		
	MGG	0.989	**0.982**	**0.986**		
PLT	W	0.969	0.950	0.959		
	WG	0.967	0.963	0.965		
	MGG	**0.979**	**0.989**	**0.984**		
All	W	**0.993**	0.977	0.985	0.973	48
	WG	0.986	0.979	0.983	0.975	424
	MGG	0.989	**0.983**	**0.986**	**0.985**	145

5.3.3 Deviating Cell Types

YOLOv5_SoTi tested on subsets containing the images with giant PLTs, abnormal RBCs and Malaria/parasites is shown in Table 5.13. It should be noted that the subset denoted 'Giants PLTs' consists of slides where more giant PLTs have been noted than normal, and does not contain all the giant PLTs in the test set. The subset with abnormal RBCs refers to abnormal shape, size and/or colour. The results below are not a comparison and therefore do not have the best performances marked in bold.

Table 5.13: YOLOv5_SoTi performance on subsets with deviating cell types.

Class	Test subset	Precision	Recall	F1(@IoU 0.5)	Macro F1	Images (n)
RBC	Giant PLT	0.991	0.986	0.989		
	Abnormal RBC	0.980	0.975	0.978		
	Malaria/parasites	0.989	0.981	0.985		
PLT	Giant PLT	0.955	0.982	0.969		
	Abnormal RBC	0.967	0.949	0.958		
	Malaria/parasites	0.940	0.948	0.944		
All	Giant PLT	0.989	0.986	0.988	0.979	111
	Abnormal RBC	0.978	0.972	0.975	0.968	170
	Malaria/parasites	0.987	0.980	0.983	0.965	143

5.4 Current Algorithms at CellaVision

This section presents results that separately compare the performance of YOLOv5_SoTi and the algorithms currently used at CellaVision. All results in this section are based on the test set.

5.4.1 RBC Count

Comparing YOLOv5_SoTi as an RBC counter to RBC_CV per slide showed 3.8 percentage points lower mean error, 6 percentage points lower standard deviation and 43.4 percentage points lower max error, as can be seen in 5.14. Comparing the performance per region

show similar numbers with 2.6 percentage points lower mean error, 4.95 percentage points lower standard deviation and 24.1 percentage points lower max error.

Table 5.14: Comparison between YOLOv5_SoTi and RBC_CV for counting RBCs. Calculated per individual image and per slide region. Best (lowest) percentage in bold.

	RBC Counter	Mean error (%)	Std error (%)	Max error (%)
Per image	YOLOv5_SoTi	**1.4**	**1.3**	**7.6**
	RBC_CV	5.2	7.3	51.0
Per region	RBC_SoTi	**0.9**	**0.65**	**3.1**
	RBC_CV	3.5	5.6	27.2

Plots of the RBC detection counts of YOLOv5_SoTi and RBC_CV against the true counts for all individual images can be seen in Figure 5.1 and for slide image regions in Figure 5.2. The plots show that YOLOv5_SoTi is the closest to the ground truth and that it results in less outliers (unusually large errors).

Figure 5.1: RBC count predicted by YOLOv5_SoTi (blue) and RBC_CV (orange) plotted against the true number of RBCs in each *individual image* (948 images) in the test set. Green line shows perfect count for reference.

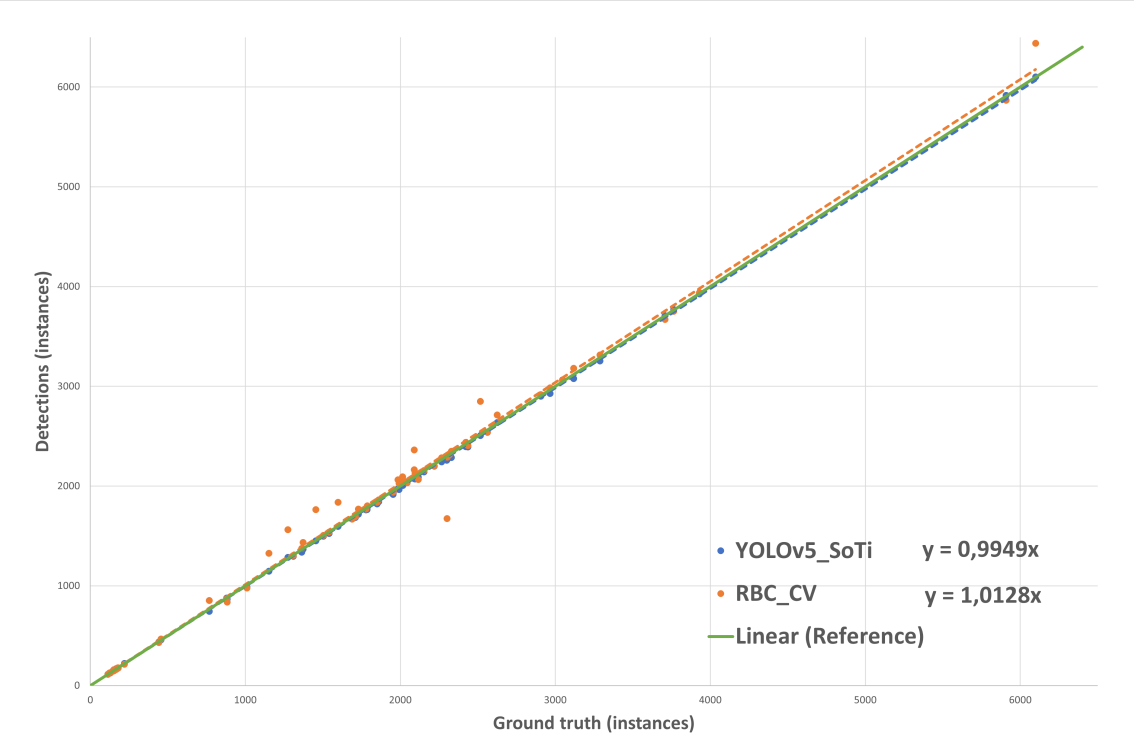
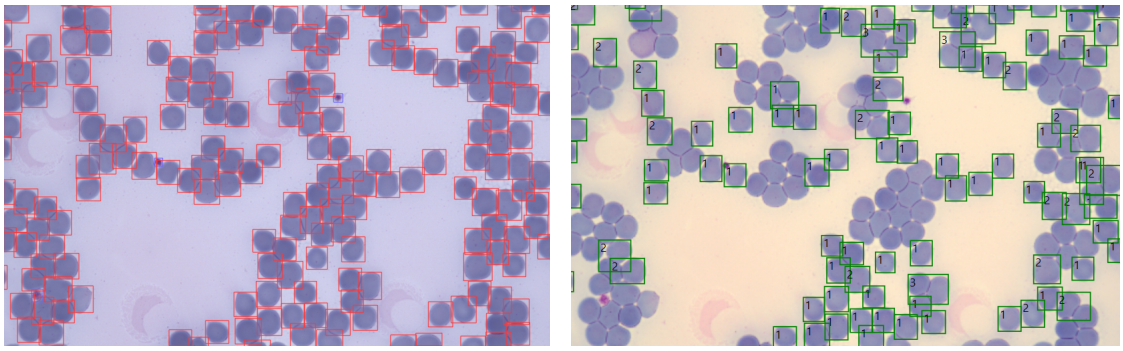


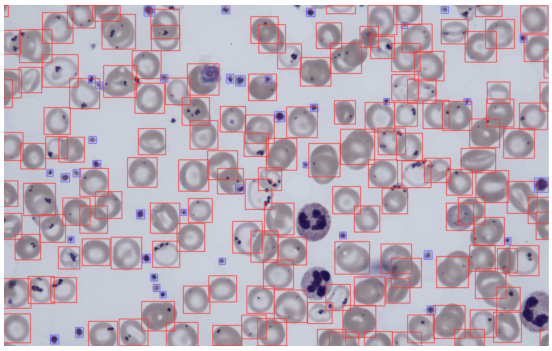
Figure 5.2: RBC count predicted by YOLOv5_SoTi (blue) and RBC_CV (orange) plotted against the true number of RBCs in each *slide region* (70 regions from a total of 22 slides) in the test set. Green line shows perfect count for reference.

Images of the worst RBC count using RBC_CV next to the count from YOLOv5_SoTi on the same area are shown in Figure 5.3, and the reverse shown in Figure 5.4.

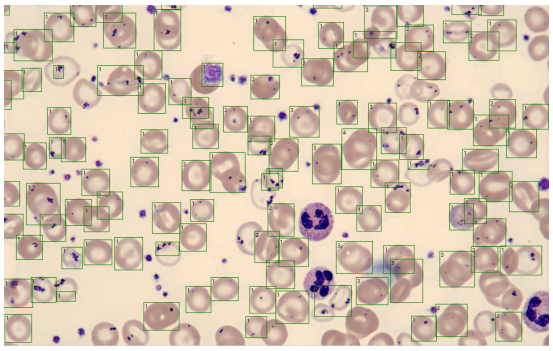


(a) YOLOv5_SoTi (0.04% error in entire region). (b) RBC_CV (27% error in entire region).

Figure 5.3: Comparison of the two algorithms on a region where RBC_CV performed worst. YOLOv5_SoTi have both RBCs (red boxes) and PLTs (blue boxes) detected. The difference in image colour is due to (b) being modified by CellaVision software, but the underlying images are identical.



(a) YOLOv5_SoTi (3.1% error in entire region).



(b) RBC_CV (15% error in entire region).

Figure 5.4: Comparison of the two algorithms on a region where YOLOv5_SoTi performed worst. YOLOv5_SoTi have both RBCs (red boxes) and PLTs (blue boxes) detected. The difference in image colour is due to (b) being modified by CellaVision software, but the underlying images are identical.

5.4.2 PLT Detection

Performance on the test set for PLT_CV and YOLOv5_SoTi shows PLT_CV as the best PLT detector based on F1 score and recall on all PLTs, see Table 5.15. Evaluating performance on only large and giant PLTs separately yielded the same type of results, where the F1 score was best for PLT_CV in both cases, see Table 5.15. It should be noted that YOLOv5_SoTi is trained and tested on image size 960×608 and PLT_CV on image size 1088×704.

Table 5.15: Comparison between YOLOv5_SoTi and PLT_CV for detecting PLTs of different sizes.

PLT Class	Detector	Precision	Recall	F1
All PLT	YOLOv5_SoTi	**0.961**	0.957	0.959
	PLT_CV	0.952	**0.981**	**0.966**
Large PLT	YOLOv5_SoTi	0.789	0.662	0.720
	PLT_CV	**0.790**	**0.684**	**0.733**
Giant PLT	YOLOv5_SoTi	0.475	**0.422**	0.447
	PLT_CV	**0.654**	0.405	**0.500**

6 Discussion

Similarly to the results section, the discussion is divided into four main sections. The first, Section 6.1, discusses the process and results of developing the RBC pseudo labelling network as well as pseudo labelling as an area of improvement. The second, Section 6.2, discusses the process and results of developing a network detecting both RBCs and PLTs and the decisions made to select YOLOv5_SoTi as the final network. The third, Section 6.3, discusses the results of testing YOLOv5_SoTi on the test data and what conclusions can be drawn from them. It also presents separate discussions for the class imbalance issue and how RBC recall may have affected the development process as well as ways of improving it. The fourth, Section 6.4, discusses how YOLOv5_SoTi performed compared to the current algorithms at CellaVision and whether it is an improvement, based on several different aspects. The fifth and final section simply states the ethical considerations made during this thesis.

6.1 RBC model selection

The initial RBC detector was developed as a tool for more efficiently labelling the images used in our data sets, but ended up being used as a pseudo labelling network. Due to this initial purpose the RBC detector did not undergo as extensive experimentation and testing as the later RBC + PLT detector. However, as it ended up being an integral part of our work its development is discussed below.

Looking at Table 5.1 it can be seen that training on image sizes did little to improve performance with the only improvement being for AP@0.5 by merely 0.01 when training on 120% of the image size 960×608. A clear drop in performance is seen when decreasing the image size trained on. This is expected as information is lost when down-sampling an image. Although, sometimes down-sampling can increase performance as it may remove noise and help the network generalise like for FPNs, see [51] and Section 2.4.2. The results from this test should be taken lightly though, as we made the mistake of testing on the image size 960×608 for all the networks. Testing the networks on the same image size as they had been trained on would probably improve the results.

Little improvement was made when testing different YOLOv5 model sizes, although a slight improvement was made across all metrics when using the *small* or *medium* models, comparing Table 5.1 and 5.2. The rather small improvements here indicate the advantages of larger models' abilities to handle more information, but it is likely that our training data was not large enough to make significant use of this. Testing model sizes with image augmentations applied to the training data would possibly have have shown the advantage of larger models.

Lastly, applying different image augmentations to the *small* model size only improved AP@0.5 and AP@0.5:0.95 by a small margin, comparing Table 5.2 and 5.3. We find this quite surprising and rather interesting, considering that significant improvements to PLT detection were later made when applying the same type of augmentations. Possible reasons for this is that the augmentations used were the wrong ones or set to too high or low levels for RBC detection, and that RBC and PLTs need different types of augmentation. It is also possible that the RBCs not found and falsely classified were due to issues such as overlap which may be difficult to fix with augmentations.

6.1.1 Pseudo Labelling

The use of pseudo labelling in this thesis is an interesting topic which perhaps was not explored as extensively as it could have. It was primarily used as a means of accessing all PLT annotation data and we were therefore satisfied with RBC detection performance not decreasing when adding pseudo labelled RBC data to the training set. As our work progressed the performance was significantly worse on PLT and focus was therefore placed on improving PLT detection, slightly neglecting the topic of improving RBC detection. Since 72 % of the RBCs in our training set were pseudo labelled, see Figure 4.1, further exploration and experimentation within the topic might have been a key to improving RBC detection. Semi-supervised learning and subsequently pseudo labelling is a scientific topic in itself with a lot of research presenting improvement techniques not explored here.

When using a network to pre-annotate data to later manually complete the annotations as we initially did, a refined RBC detection network could be useful for others wanting to efficiently annotate images. A mistake made was having too low an IoU threshold (0.6) during the validation when training the network, possibly being the reason for boxes needing to be adjusted for more cells than what would be considered efficient. This is seen in Table 5.3 where the AP@0.5 is 0.987 and the AP@0.5:0.95 0.812 indicating that quite a few boxes have an IoU in the lower range of 0.5:0.95. How large an IoU is necessary depends on the aim. For the sake of simply counting red blood cells the IoU is of next to no importance as long as the count is correct. However in applications where the user wants to view each individual cell, as is the case in some CellaVision applications, a high IoU could be crucial for getting a full view.

When the network is used for pseudo labelling, a higher IoU in both training and detection could also be argued to be important. If the entire cell is not within the box or, reversely, extra surrounding objects included in the box there is a loss or distortion of information. The effects of this may propagate through training and result in a worse performance than would be attained with accurate boxes in the training set. When networks are developed to perform a task previously performed by humans we want them to closely imitate how humans approach the problem. If the boxes are different from those manually annotated, the network no longer trains on what one expects it to and might form biases which are hard to see, especially if performance is still considered good. This may have long-term effects if the network fails to generalise and one needs to understand why.

6.2 RBC + PLT Detector

The initial evaluations of testing parameters individually showed small to no changes in performance when training and testing on different model sizes, see Table 5.5, and pre-trained weights, see Table 5.6. When applying augmentations PLT recall increased remarkably, from a highest score of 0.885 to 0.952, see Table 5.7, using PLT_CV's augmentations. The precision for RBCs increased slightly, but the recall remained at 0.982. Why the augmentation had more effect on PLT performance may have several reasons, where the number of instances might be a crucial one. Augmentations add variation to the images, in this case mostly regarding hue, saturation and value. This could lead to a more robust network that is not as dependent on the colour being exactly as in the training data, and for the network to be more confident in its detections. The fact that the recall increased while the precision decreased could also strengthen the theory of the network being more confident. More confident predictions would result in more detections over the confidence threshold, including false positives. The RBC recall did not change significantly regardless of the alterations to the model.

Adding a flip up/down to get more instances was wrongly assumed to increase the performance. The assumption was that it would not change anything in the images, only flip them up and down and therefore generate more data. The precision improved slightly for all classes but the recall decreased, especially for PLTs which decreased from 0.955 to 0.936, see Table 5.8. The reason for this is not clear, but might have to do with added generalisation of the over represented morphologies of PLTs. A flip could help if the training data does not have images with evenly distributed objects, meaning that the network learns that objects are more likely to be located in certain parts of the images. Our training data is very evenly distributed, since the cells have similar probability to be located anywhere on the slide, see Figure 4.4a, which might have given positional augmentations less impact in the training. Adding a shear would possibly help increase the recall, since the cells would get more variations in their shape. The shear neither increased nor decreased the recall significantly, and all other performance metrics were either unchanged or lowered. Out of all remaining tested hyper parameters the ones that improved the performance metrics significantly were anchors = 5 and 'Image weights'.

The image weights had most influence on improving the PLT performance, which was reflected in the highest macro F1 score, see Table 5.9. Since image weights allow images with the highest error to be more likely to be selected in the training, it is a reasonable cause for the increased performance. However, the overall performance did not improve as the precision, and therefore the F1 score, for RBCs decreased. This is interesting since the error during training is calculated for the combined classes, where RBC performance has more influence due to the superior number of instances. A possible reason for the decreased RBC performance combined with the increased PLT performance is that the images where RBCs had a high error contained RBCs that are difficult to detect. Another reason could be that the pseudo labelling of the images with bad performance for RBCs was inadequate and several false positives existed. This would lead to a low precision and a high error, and the network would be less confident in its findings when these images get a higher probability for being selected during training. Since pseudo labelling is a semi-supervised method where the user trusts a network trained on labelled data to make

predictions on unseen data, its actual performance cannot be evaluated. This adds uncertainty to the use of pseudo labels.

When combining 'Image weights' and anchors = 5, we discovered that the performance did not improve. This indicates that improvements from hyper parameters are not linear, meaning that two hyper parameters improving performance individually does not imply that they will improve the performance combined. It was still the network with image weights that performed best at this point, why this network was kept.

Increasing the image size resulted in an increase of all performance metrics for PLTs, while all metrics for RBCs were unchanged or decreased, see Table 5.10. The reason why the performance increased for PLTs but not for RBCs might have to do with the sizes of the objects. In the subsampling and convolution steps in a CNN, see Figure 2.11, some information is lost, why smaller objects may loose to much of information for the CNN to later detect. This together with the fact that the performance for detecting PLTs in larger images has previously been stated to increase at CellaVision, see Section 2.5.3, are examples of why this is reasonable. The size of a PLT is much smaller than for an RBC, indicating that a larger image size would benefit the detection for these more than for RBCs. An error made at this stage was to use the same down sampled images (960×608) for up sampling to the larger image sizes, since down sampling results in loss of information. A more correct way of testing different image sizes would be to down sample from the original image size (1920×1200) to the sought after image sizes, which would possibly yield even higher performance.

A deeper analysis of augmentations and different hyper parameters would be interesting for increasing performance. Learning rate was not tested in this thesis, which according to Section 2.4.9 could be an important hyper parameter to test. On the other hand, the learning rate used in YOLOv5 is already a type of learning rate schedule, and may therefore already be the optimal solution. Testing different hyper parameters is a very time-consuming task. To improve the performance of our model more hyper parameters, and especially the combinations of them, could be tested. As mentioned above, the impact each hyper parameter has on the performance is not linear, meaning that even if improvements were not seen from some hyper parameters, a combination of them might be the optimal choice. To explore this further, an algorithm that explores many different types of hyper parameters and combinations of them could be used.

6.3 Final Testing

Evaluating YOLOv5_SoTi on the test set yielded a lot of interesting results. Most prominently a rather small drop across our chosen performance metrics was seen compared to evaluating on the validation set. The macro F1 score only dropped by 0.006 and the overall F1 score by 0.001 (compare image size 960×608 in Table 5.10 and 5.11). There was, regrettably, a drop in PLT performance despite series of efforts to improve it. Conversely the RBC F1 score very marginally increased. This points to our validation data set being well selected, which is further confirmed by its similarity in composition shown in Figure

4.2 and 4.3. We consider the performance to be very good, especially considering that the test set was selected by CellaVision prior to our thesis work, and considered to include "difficult" and diverse blood smear images.

Also training and testing YOLOv5_SoTi for different image sizes on the test set did not improve neither overall F1 or macro F1 score. It did however improve the PLT recall significantly for image size 1344, see Figure 5.11, confirming that an increased image size can have a positive impact on PLT detections.

Dividing the test set into subsets based on the stains used in the images proved interesting as there was a rather large difference in performance between the subsets. The MGG subset outperformed the others in macro F1 by a 0.01 margin, see Figure 5.12, which can be seen as rather large considering previous testing in this thesis. This primarily came from the performance on PLTs where the F1 score was the highest by almost a 0.02 margin and both the precision and recall the highest achieved in any test while developing YOLOv5_SoTi, see Figure 5.12. There may be several explanations for this. It could simply be that the network was more easily trained to MGG images because of factors such as contrast between cells and background. Considering the great performance on PLTs, it is an indication that the MGG stain does this for PLTs. It is also possible that the image slides considered "difficult" were more present in the worse performing stain subsets.

Dividing the test set into subsets based on the presence of giant PLTs, abnormal RBCs (regarding shape, size and colour) and Malaria/parasites was primarily done hypothesising that these subsets would be difficult to detect on. Surprisingly, performance on the 'Giant PLT' subset was very good even for PLT detection, see Table 5.13. This is difficult to explain especially considering how poor detection is for large and giant PLTs on the entire test set, see Figure 5.15, but could be that the image quality happens to be unusually high in this subset. Despite there being a high presence of giant PLTs in the images in this subset, there were only 111 images in the subset, and therefore not many giant PLTs overall. This may lead to rather unreliable metrics. Less surprisingly, the presence of abnormal RBCs decreased performance significantly across all metrics for RBC detection, compare Table 5.11 and 5.13. The presence of Malaria/parasites did not affect RBC performance much but lowered PLT performance, compare Table 5.11 and 5.13, which could be explained by this image category being rare and that parasites/inclusions can have similar appearance to PLTs.

6.3.1 Class Imbalance

A lot of effort was put into improving performance on PLT detection as it was always trailing behind the performance on RBC detection. The efforts were successful, increasing the PLT F1 score on the validation set from 0.897, see Table 5.5, to 0.971, see Table 5.10. This was however still below the performance on RBCs.

It was identified early on that there was a heavy imbalance between RBCs and PLTs in the training set, see Figure 4.2. Large class imbalances notoriously complicate classifica-

tion/detection on the under represented class(es) in neural network tasks. Our efforts to weight the classes' contributions to the loss by using 'class weights' was unsuccessful, but the use of 'image weights' showed a noticeable improvement, see Table 5.9. The effects of imbalance in PLT subclasses shown in Figure 4.3 can also be seen in Table 5.15.

Unfortunately, assessing the RBC/PLT imbalance in the training set is very difficult as RBCs will always be more present than PLTs in blood smear images. A possible solution briefly discussed with our supervisors but not implemented, would be to cut out PLTs from images and then add them in large numbers on top of the original images, thus creating artificial data where the imbalance is much smaller.

6.3.2 Improving RBC Recall

An interesting aspect that was present throughout the RBC + PLT network development process was the RBC recall. Viewing Table 5.5 - 5.11 one can see that it is 0.982 or 0.983 for every single test except two. The RBC recall was only higher than this when replacing SGD with Adam as the optimiser used, see Table 5.9. A recall of 0.983 should be considered a very good score, meaning that the network only fails to find and sufficiently well box 1.7% of RBCs. Although impressive, the fact that recall barely ever changed indicates that the same RBCs are consistently missed which points towards a weakness in the training set. If this is the case we suggest two inter-related causes:

Imbalance in the training set. RBCs can, just like PLTs, be divided into sub-categories based on morphology. Some of these morphologies differ largely in size and shape from the norm. Figure 4.4b shows this as the vast majority of cells fall within a small range of sizes with relatively square shape. At the same time there is a clear presence of infrequent but very different sizes and shapes. This indicates an imbalance. Another indication is that the RBC recall dropped to 0.975 when testing only on images from slides containing a high presence of abnormal RBCs.

The pseudo labelling network's weaknesses and biases propagating throughout our development. Since the network developed for pseudo labelling did not go through as extensive attempts of improvement as the RBC + PLT detector and was not even evaluated based on recall, the RBC recall issue was not even noticed. If it was present at this stage, pseudo labelling will likely have made under-represented RBC morphologies even more difficult to detect.

Lastly, a possible cause unrelated to RBC morphology is the difficulty of detecting overlapping objects. It can be seen that YOLOv5_SoTi sometimes struggles with overlapping RBCs in Figure 5.4b. This may be improved with augmentations that exaggerate edges. NMS could also be a reason as the algorithm may throw away overlapping detections, see Section 2.4.6. We were aware of this but failed to implement variations of NMS which are said to improve detection of overlapping objects, such as soft NMS described in [69].

6.4 Comparison with current algorithms at CellaVision

Since the RBC algorithm used at CellaVision today is a segmentation algorithm and the PLT algorithm is an object detector, the comparisons will be discussed separately.

6.4.1 RBC Count

Table 5.14 shows that YOLOv5_SoTi has 3.8 percentage points lower mean error on image level and 2.6 lower on region level, compared to RBC_CV. The lower standard deviation of 6 percentage points on image level and 4.95 percentage points on region level indicate YOLOv5_SoTi having less outliers both on image and region level. Additionally, YOLOv5_SoTi has a 43.4 percentage points lower max error on image level and 24.1 percentage points on region level. Figure 5.1 shows that RBC_CV predicts both too many and too few RBCs in the individual images, which is rare for YOLOv5_SoTi. RBC_CV also has more extreme outliers, indicating that YOLOv5_SoTi is more robust. Since the region errors shown in Figure 5.2 are made up of the images in Figure 5.1, the over estimations by RBC_CV may sometimes cancel out under estimations, thus improving results on the regions. This can also be seen in Table 5.14 where the mean error, standard deviation and max error are all lower for the entire regions than the individual images. Although the error on the whole region is what matters from an application point of view, the large individual image errors should be seen as a weakness from a scientific point of view. Figure 5.3 shows images from the worst performing region for RBC_CV. Comparing Figure 5.3b and 5.3a clearly shows that YOLOv5_SoTi found more of the RBCs in this image than RBC_CV, indicating an improved performance on these types of slides containing RBC aggregations and/or this specific stain. Figure 5.4 shows images of the worst performing region for YOLOv5_SoTi. It is clear that the overlapping RBCs in this region are hard for the object detector to find, see Figure 5.4a. Despite this being the worst performing region for YOLOv5_SoTi, it still has a 11.9 percentage points more accurate prediction than RBC_CV.

Overall we consider YOLOv5_SoTi to outperform RBC_CV. As RBC_CV is considered good enough for clinical use, YOLOv5_SoTi should be too for RBC counting.

6.4.2 PLT Detection

As can be seen in Table 5.15, YOLOv5_SoTi has better precision than PLT_CV, but lower recall for all PLTs. The F1 score is also lower for our detector. This means that PLT_CV finds more of the PLTs, but that it has more false positives than YOLOv5_SoTi. How to evaluate which model is -the best one- depends on the goal. Only looking at the F1 score, PLT_CV is the best performing one, as well as when it is most important to find many of the PLTs. On the other hand, if it is more of interest that the findings are true positives, YOLOv5_SoTi is the best performing one. To get the recall higher on our model, an idea would be to lower the confidence threshold during detection. This would most likely result in more findings, where both true and false positives would be included. In that case, the precision would decrease and the recall increase.

When comparing the performance for large and giant PLTs, the metrics are higher for PLT_CV in all categories except for the giant PLT recall, see Table 5.15. There is not a big difference between the two networks' performances, except for the precision on giant PLTs. The metrics on large and giant PLTs are hard to compare, partly because of the low number of instances. In the test set there are 1152 large PLTs, and only 53 giant PLTs, see Figure 4.3. In addition to this, our evaluation algorithm is very sensitive to the sizes. For example, if a bounding box for a PLT in the ground truth file is 4µm, and in the detection file only 3.999µm, the large PLT would be considered as a false negative. The same would apply to the border between large and giant PLTs. A way to improve the detection of large and giant PLTs would be to gather more data, since giant PLTs are especially under represented in the training data as well.

6.4.3 Inference Time

When working with clinical applications like this one, performance versus speed is something that always needs to be considered. PLT_CV has an inference time of 89ms/image and RBC_CV of 211ms/image, with a total of 300ms/image. YOLOv5_SoTi with image sizes 960×608, 1152×736 and 1344×832 had the inference times 150, 220 and 250ms/image respectively. Based on these inference times, all three image sizes tested with YOLOv5_SoTi have a lower inference time than today's algorithms. RBC_CV and PLT_CV work simultaneously, meaning that they have to share the processor power. This in combination with the segmentation algorithm being slower results in the slower analysis. The advantage with YOLOv5_SoTi is that both analyses are done simultaneously, meaning that there is only one process running at a time, resulting in a lower inference time. Depending on what is considered most valuable in terms of performance (precision or recall) and inference speed, different models or image sizes would be considered the best one.

6.5 Ethical Considerations

The blood samples used in this master's thesis did not contain any patient data for the sake of patients' security and integrity.

7 Conclusion

In summary, this master thesis has shown it possible to achieve good results with a multi-class object detector, detecting both RBCs and PLTs. The performance for counting RBCs proved to be an improvement compared to the segmentation algorithm currently used at CellaVision, as a result of more accurate and robust predictions. Meanwhile, the performance for detecting PLTs was slightly lower than for the current algorithm. The inference time for YOLOv5_SoTi was lower than the combined inference time of CellaVision's RBC and PLT detection algorithms.

8 Future Work

There are several possible improvements and experiments that could be performed if allowed to continue with the work presented in this thesis:

Deeper analysis of what types of cells or stains the network performs poorly on would be interesting. If this could be isolated, these types should be added to the training data to improve the data diversity and balance, and possibly increase performance.

A more extensive evaluation of the pseudo labelling would also be interesting. If the pseudo labelling lacked in performance for some types of RBCs, it could possibly be improved. It would also be interesting to compare the network used for the initial pseudo labelling and YOLOv5_SoTi on RBC performance. If YOLOv5_SoTi performed better than the pseudo labelling network it could be used for generating new, more accurate, pseudo labels. This approach could be repeated until the best performance on RBCs, and thus the best pseudo labelling network, is achieved.

If the inference time would turn out to be too high, all model and hyper parameters for YOLOv5_SoTi could be applied and modified for a nano network, possibly in combination with the exploration of an algorithm that explore different combinations of parameters.

Lastly, a further development of YOLOv5_SoTi could be to teach the network to classify different types of cell sub classes. RBCs can for example be classified based on size, colour and shape, as well as the presence of parasites, such as malaria.

Bibliography

[1] Johns Hopkins Medicine. *Facts about blood* [Online]. Retrieved March 8 2022.

[2] Encyclopaedia Britannica. *Red blood cells* [Online]. Retrieved March 8 2022.

[3] H. Aliyu, M. Rayak, R. Sudirman. (2019, April). "Normal and abnormal red blood cell recognition using image processing". *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 14, issue 1, pp. 96-100.

[4] Encyclopaedia Britannica. *White blood cells* [Online]. Retrieved March 10 2022.

[5] Encyclopaedia Britannica. *Platelets* [Online]. Retrieved March 10 2022.

[6] Johns Hopkins Medicine. *What Are Platelets and Why Are They Important?* [Online]. Retrieved March 21 2022.

[7] Mayo Clinic. (2022, April). *Thrombocytopenia (low platelet count)* [Online]. Retrieved March 21 2022.

[8] International Council for Standardization in Haematology Expert Panel on Cytometry, International Society of Laboratory Hematology Task Force on Platelet Counting. (2001, March). "Platelet counting by the RBC/platelet ratio method. A reference method.". *American journal of clinical pathology*, vol. 115, issue 3, pp 460–464.

[9] American Society of Haematology. (2016, December). *Giant platelets* [Online]. Retrieved April 27 2022.

[10] CellaVision. *CellaVision peripher blood application* [Online]. Retrieved March 22 2022.

[11] F. Mokobi. (2020, September). *Romanowsky Stains- Principle, Types, Applications* [Online]. Retrieved March 22 2022.

[12] B. Marr. (2021, July). *Deep Learning Vs Neural Networks – What's The Difference?* [Online]. Retrieved March 23 2022.

[13] H. Maheshwari. (2021, September). *Everything you need to know about "activation functions" for deep learning models* [Online]. Retrieved March 29 2022. Permission to use image given by author.

[14] L. Hardesty. (2017, April). *Explained: Neural networks* [Online]. Retrieved March 24 2022.

[15] Data base camp. (2021, November). *Artificial Neural Networks – Basics* [Online]. Retrieved March 25 2022.

[16] S. Sharma. (2017, September). *Activation Functions in Neural Networks* [Online]. Retrieved March 25 2022.

[17] J. Jordan. (2017, July). *Neural networks: activation functions.* [Online]. Retrieved April 25 2022.

[18] AfterAcademy. (2019, December). *Mastering Activation Functions in Neural Networks* [Online]. Retrieved April 25 2022.

[19] A. Amidi & S. Amidi. (2018). *Deep Learning cheatsheet* [Online]. Retrieved June 11 2022. Permission to use image given by author.

[20] S. Doshi. (2019, January). *Various Optimization Algorithms For Training Neural Network* [Online]. Retrieved April 25 2022.

[21] J. Brownlee. (2019, January). *Loss and Loss Functions for Training Deep Learning Neural Networks* [Online]. Retrieved April 25 2022.

[22] J. Brownlee. (2019, January). *Gentle Introduction to the Adam Optimization Algorithm for Deep Learning* [Online]. Retrieved May 22 2022.

[23] IBM Cloud Education. (2020, October). *Gradient Descent* [Online]. Retrieved May 24 2022.

[24] Dive Into Deep Learning. *Forward Propagation, Backward Propagation, and Computational Graphs* [Online]. Retrieved May 23 2022.

[25] Goodfellow et al. "Chapter 6.5 Back-Propagation and Other Differentiation Algorithms". in *Deep Learning*. MIT Press, 2016, pp 200.

[26] R. Potter. (2019, November). *What is Data Annotation and What are its Advantages?* [Online]. Retrieved February 15 2022.

[27] J. Bhattacharyya. (2020, October). *Pseudo Labelling – A Guide To Semi-Supervised Learning* [Online]. Retrieved February 24 2022.

[28] IBM Cloud Education. (2020, August). *Supervised Learning* [Online]. Retrieved March 29 2022.

[29] IBM Cloud Education. (2020, September). *Unsupervised Learning* [Online]. Retrieved March 29 2022.

[30] DataRobot. (2020, August). *Semi-supervised Learning* [Online]. Retrieved March 29 2022.

[31] V. Subramanyam. (20121, January). IOU (Intersection over Union). Retrieved May 24 2022.

[32] N. Harikrishnan. (2019, December). *Confusion Matrix, Accuracy, Precision, Recall, F1 Score* [Online]. Retrieved March 15 2022.

[33] Wang, J., Zhang, J., An, Y. et al. (2016, August). "Biomedical event trigger detection by dependency-based word embedding", *BMC Med Genomics*, vol. 9, issue 45.

[34] J. Hui. (2018, March). *mAP (mean Average Precision) for Object Detection* [Online]. Retrieved March 21 2022.

[35] IBM Cloud Education. (2021, March). *Overfitting* [Online]. Retrieved March 29 2022.

[36] Iglesias, L.L., Bellón, P.S., del Barrio, A.P. et al. "A primer on deep learning and convolutional neural networks for clinicians". *Insights Imaging*, vol 12, issue 117, 2021. Available via license: CC BY 4.0

[37] J. Brownlee. (2020, April). *How Do Convolutional Layers Work in Deep Learning Neural Networks?* [Online]. Retrieved March 30 2022.

[38] D. Batista. (2018, March). *Convolutional Neural Networks for Text Classification* [Online]. Retrieved May 24 2022.

[39] J. Brownlee. (2019, July). *A Gentle Introduction to Pooling Layers for Convolutional Neural Networks* [Online]. Retrieved March 30 2022.

[40] Y-P. Huang, S.-Y. Lee. (2021, May) "An Effective and Reliable Methodology for Deep Machine Learning Application in Caries Detection". Available via license: CC BY 4.0

[41] V. Mallawaarachchi. (2017, July). *How to define a Fitness Function in a Genetic Algorithm?* [Online]. Retreived March 31 2022.

[42] A. Lohia, K. D. Kadam, R. R. Joshi, A. M. Bongale. (2021, Feb). "Bibliometric Analysis of One-stage and Two-stage Object Detection" v. *University of Nebraska-Lincoln*

[43] R. Girshick, J. Donahue, T. Darrell, J. Malik. (2014, Oct). "Rich feature hierarchies for accurate object detection and semantic segmentation Tech report (v5)" [Online]. *UC Berkeley.*

[44] R. Girshick. (2015, Sep). "Fast R-CNN" [Online]. *Microsoft Research.*

[45] S. Ren, K. He, R. Girshick, J. Sun. (2016, Jan). "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks" [Online]. *Microsoft Research* .

[46] J. Redmon, S. Divvala, R. Girshick, A. Farhadi. (2016, May). "You Only Look Once: Unified, Real-Time Object Detection" [Online]. *University of Washington, Allen Institute for AI, Facebook AI Research.* Permission to use images given by author.

[47] J. Redmon, A. Farhadi. (2016, Dec). "YOLO9000: Better, Faster, Stronger" [Online]. *University of Washington, Allen Institute for AI* .

[48] J. Redmon, A. Farhadi. (2018, Aug). "YOLOv3: An Incremental Improvement" [Online]. *University of Washington* .

[49] A. Bochkovskiy, C. Wang, H. M. Liao. (2020, April). *YOLOv4: Optimal Speed and Accuracy of Object Detection* [Online].

[50] Editorial team at Towards AI. (2020, July). *YOLO V5—Explained and Demystified* [Online]. Retrieved April 26 2022.

[51] J. Hui. (2018, March). *Understanding Feature Pyramid Networks for object detection (FPN)* [Online]. Retrieved April 26 2022. Available via license: CC BY 4.0

[52] X. Renjie. (2021, February). "A Forest Fire Detection System Based on Ensemble Learning", *Forests*, vol 12, pp. 217.

[53] MathWorks. (2022). *Anchor Boxes for Object Detection* [Online]. Retrieved April 26 2022.

[54] R. Hamid et al. (2019, June). "Generalized Intersection over Union", *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR).*

[55] H. Gva. (2019, June). *PR-172: Generalized Intersection over Union: A Metric and A Loss for Bounding Box Regression* [Online]. Retrieved May 24, 2022.

[56] D. Godoy. (2018, Nov). *Understanding binary cross-entropy / log loss: a visual explanation* [Online]. Retrieved March 15, 2022.

[57] B. Alexe, T. Deselaers, V. Ferrari. "Measuring the Objectness of Image Windows", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, issue 11, pp. 2189 - 2202, Nov 2012.

[58] U. Almog. (2020, Oct). *Object Detection With Deep Learning: RCNN, Anchors, Non-Maximum-Suppression* [Online]. Retrieved March 16 2022.

[59] K. Sambasivarao. (2019, Oct). *Non-maximum Suppression (NMS)* [Online]. Retrieved March 16 2022.

[60] T. Lin, M. Maire, S. Belongie, L. Bourdev, R. Girshick, J. Hays, P. Perona, D. Ramanan, C. L. Zitnick, P. Dollar. (2015, Feb). "Microsoft COCO: Common Objects in Context" [Online]. *European Conference on Computer Vision*

[61] J. Brownlee. (2019, Aug). *How to Configure the Learning Rate When Training Deep Learning Neural Networks* [Online]. Retrieved April 7 2022.

[62] K. Alsaif, A. Albasha. "Color Image Enhancement Based on Contourlet Transform Coefficients" [Online]. *Australian Journal of Basic and Applied Sciences*, vol. 7, issue 8, pp. 207-213, Jan 2013.

[63] Wikipedia. HSV color solid cylinder. Retrieved May 24 2022. Available via license: Wikimedia Commons.

[64] M. Sharma. (2020, May). *Image Augmentation with skimage — Python* [Online]. Retrieved May 10 2022.

[65] T. Lin, P. Goyal, R. Girshick, K. He, P. Dollar. (2018, Feb). "Focal Loss for Dense Object Detection" [Online], *Facebook AI Research (FAIR).*

[66] G. Jocher. (2022). *Tips for Best Training Results* [Online]. Retrieved May 10 2022.

[67] Peltarion. *Class weights* [Online]. Retrieved May 10 2022.

[68] G. Jocher. (2020, Oct). *Issue 1238, img-weights* [Online] . Retrieved May 10 2022.

[69] Bodla, Navaneeth & Singh, Bharat Chellappa, Rama & Davis, Larry. (2017, August). "Soft-NMS — Improving Object Detection with One Line of Code". *Conference: 2017 IEEE International Conference on Computer Vision (ICCV).*

A Full YOLOv5 Architecture

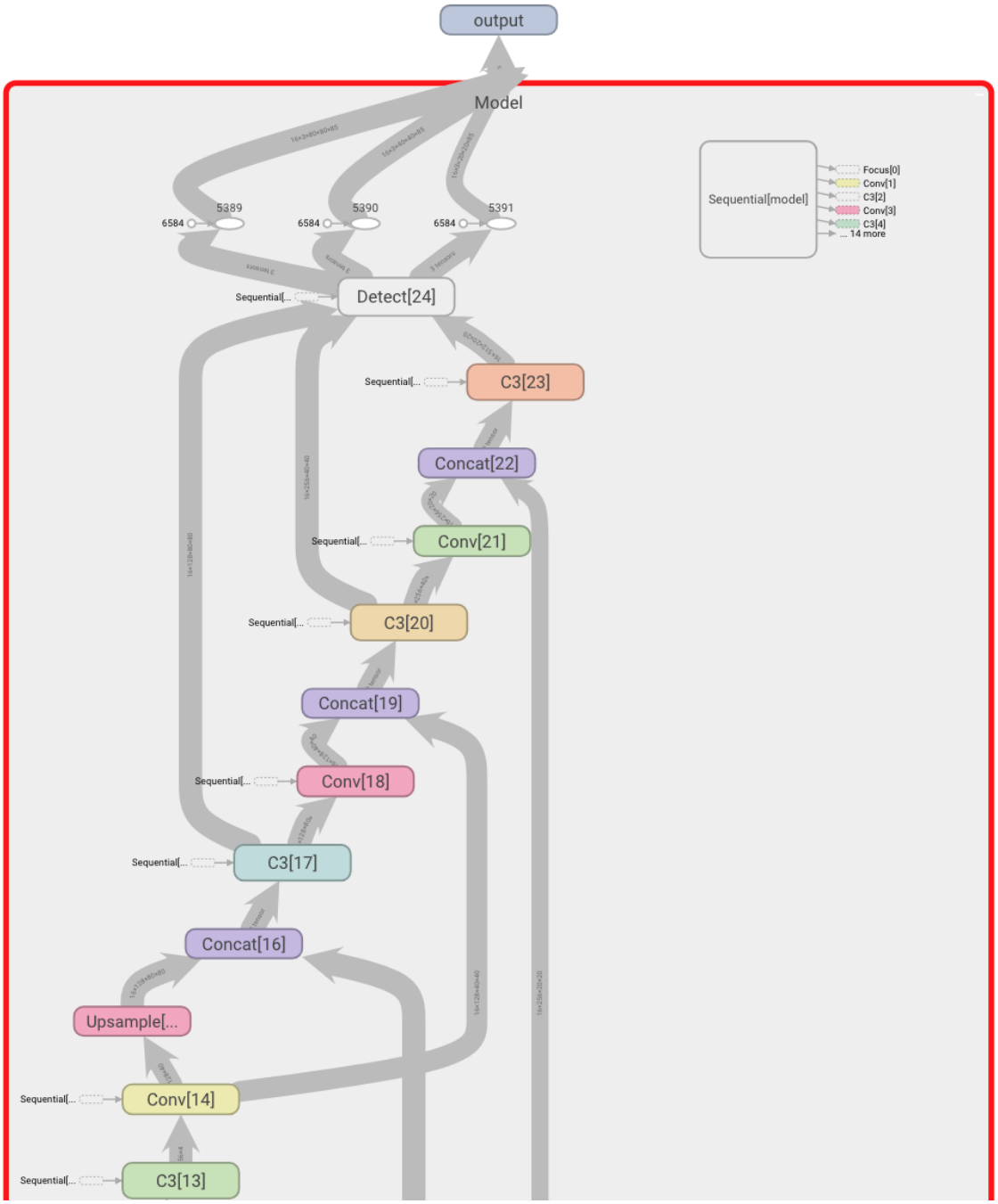


Figure A.1: Top half of full YOLOv5 architecture.

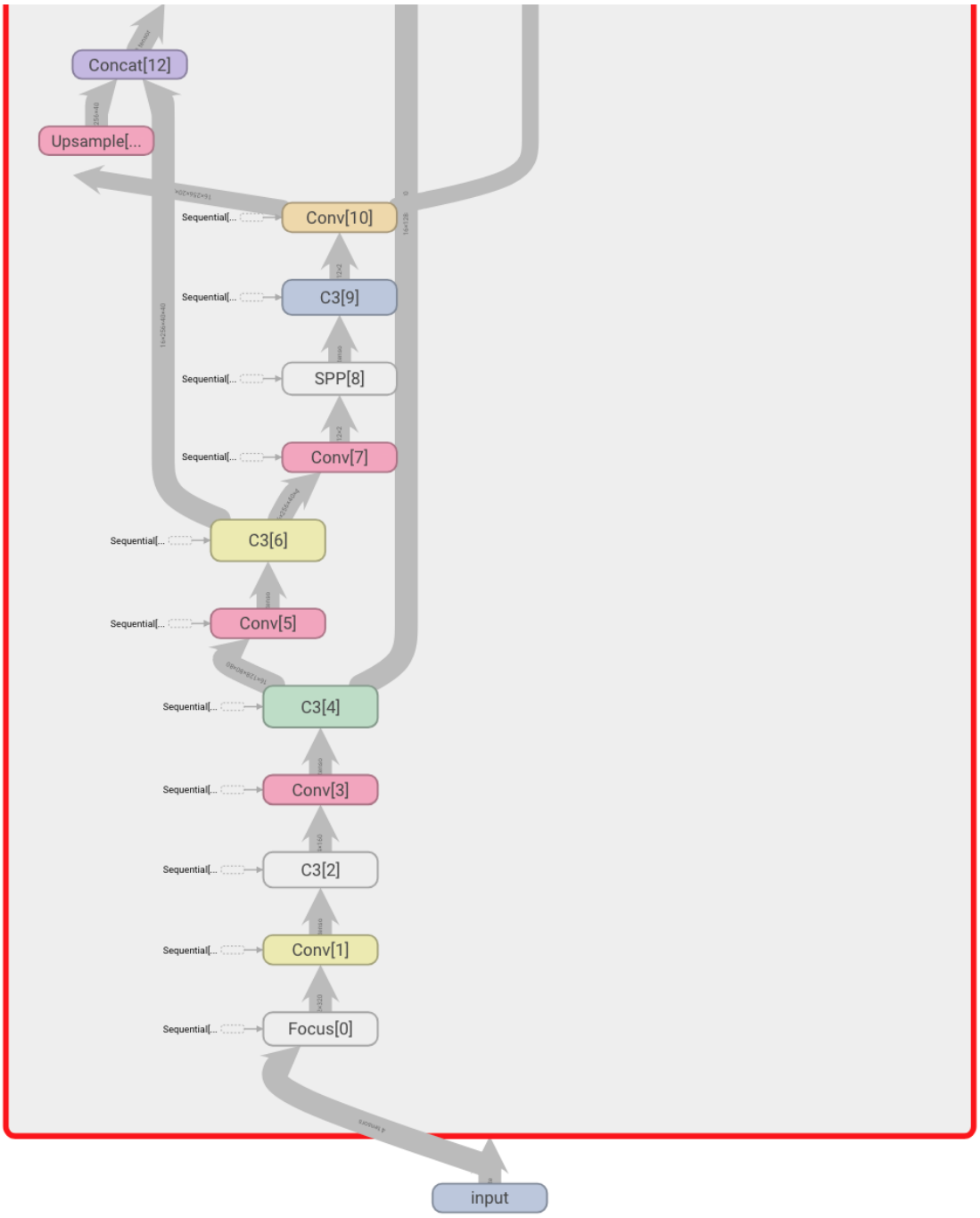


Figure A.2: Bottom half of full YOLOv5 architecture.

Master's Theses in Mathematical Sciences 2022:E20
ISSN 1404-6342

LUTFMA-3469-2022

Mathematics
Centre for Mathematical Sciences
Lund University
Box 118, SE-221 00 Lund, Sweden

http://www.maths.lth.se/