REPUBLIC OF TURKEY
YILDIZ TECHNICAL UNIVERSITY
DEPARTMENT OF COMPUTER ENGINEERING

DETERMINATION OF MOTILE / NON-MOTILE SPERM CELLS BY DEEP LEARNING-BASED DIVIDING TECHNIQUES

15011702 — Özgür KAN
16011905 — Oussama MIZAOUI

SENIOR PROJECT

Advisor
Assist. Prof. Dr. Hamza Osman İLHAN

December, 2020

TABLE OF CONTENTS

LIST OF ABBREVIATIONS iv

LIST OF FIGURES v

LIST OF TABLES vi

1 Introduction 1

2 Literature Review 2

2.1 Using Deep Learning to Streamline Intracytoplasmic Sperm Injection in Cancer Patients [1] . . . 2

2.2 A preliminary study of sperm identification in microdissection testicular sperm extraction samples with deep convolutional neural networks [2] 3

2.3 DeepSperm: A robust and real-time bull sperm-cell detection in densely populated semen videos [3] . . . 3

2.4 Segmentation of COVID-19 pneumonia lesions: A deep learning approach [4] . . . 4

3 Feasibility Study 5

3.1 Technical Feasibility . . . 5

3.1.1 Software Feasibility . . . 5

3.1.2 Hardware Feasibility . . . 6

3.1.3 Time Feasibility . . . 6

3.2 Economic Feasibility . . . 7

3.2.1 Hardware Costs . . . 7

3.2.2 Software Costs . . . 7

3.2.3 Employee Costs . . . 7

3.3 Legal Feasibility . . . 8

4 System Analysis 9

4.1 Use-Case Scenario . . . 9

4.2 Performance Metrics . . . 10

4.3 General Flowchart . . . 10

5 System Design **11**
5.1 YOLO Architecture 11
5.1.1 YOLOv1 11
5.1.2 YOLOv2 12
5.1.3 YOLOv3 12
5.1.4 YOLOv4 13
5.1.5 YOLOv5 13
5.2 R-CNN Architecture 15
5.2.1 Fast R-CNN Architecture 15
5.2.2 Faster R-CNN Architecture 16
5.2.3 Mask R-CNN Architecture 16
5.3 Detectron2 Architecture 17
5.4 EfficientDet Architecture 18
5.5 SSD - Single Shot MultiBox Detector Architecture 19

6 Application **20**
6.1 YOLOv5 Application 21
6.2 Mask R-CNN Application 22
6.3 SSD Application 23

References **24**

LIST OF ABBREVIATIONS

ICSI	Intracytoplasmic Sperm Injection
R-CNN	Region Based Convolutional Neural Networks
Fast R-CNN	Fast Region Based Convolutional Neural Networks
Faster R-CNN	Faster Region Based Convolutional Neural Networks
Mask R-CNN	Mask Region Based Convolutional Neural Networks
SSD	Single Shot Detector
YOLO	You Only Look Once
IoU	Intersection over Union
GPU	Graphics Processing Unit
CuDNN	NVIDIA® CUDA® Deep Neural Network library™

LIST OF FIGURES

Figure 3.1 Gantt Diagram 6
Figure 4.1 Use case diagram 9
Figure 4.2 General flowchart 10
Figure 5.1 YOLO Network Design [5] 12
Figure 5.2 Object Detection Architecture [8] 13
Figure 5.3 Comparision YOLOv5 Models [9] 13
Figure 5.4 Model Storage Size (MB) [9] 14
Figure 5.5 Mask R-CNN Framework [10] 16
Figure 5.6 Detectron2: Generalized R-CNN Models [11] 17
Figure 5.7 EfficientDet Architecture [12] 18
Figure 5.8 SSD Architecture 19
Figure 6.1 The best result in the YOLOv5 model 21
Figure 6.2 The worst result in the YOLOv5 model 21
Figure 6.3 The best result in the Mask R-CNN model 22
Figure 6.4 The worst result in the Mask R-CNN model 22
Figure 6.5 The best result in the SSD model 23
Figure 6.6 The worst result in the SSD model 23

LIST OF TABLES

Table 3.1 Hardware price information 7
Table 3.2 Software price information 7
Table 3.3 Employee Costs in this project 7
Table 6.1 Number of labels for each video 20

1
Introduction

In this chapter, the problem will be introduced and information about the current state of the problem will be mentioned.

Computer-aided diagnosis has been in our lives for the past few decades. The system's responsibility escalated lately with new method findings and it started to being used in vast areas. From detecting breast cancers to Alzheimer's disease, it has been a huge help for the doctors and we can say that it's becoming a necessity to utilize.

On the other hand, with the emerging technology, the focus on deep learning has increased in every field, especially on computer-aided diagnosis recently with the increased trust on deep learning techniques.

Sperm detection is one of the most important areas where computer-aided systems are being used, since the sperms are really small and there are mostly a lot of them to detect. These make detecting sperms and checking for other issues such as sperm motility or morphology infeasible for people.

In computer-aided systems, object detection is utilized to detect sperms. In this project, we are looking to find an answer for which of the most popular object detection methods works best on sperm detection by using different metrics to calculate the differences.

Within the scope of this project, different deep learning based segmentation techniques will be applied to detect sperm objects on 13 tagged sperm videos. These methods are selected as YOLO, SSD, Mask RCNN, Faster RCNN, Detectron, EfficientDet, which are referred to as "Two Level Stage" and "Feature Pyramid Detectors".

2
Literature Review

In this chapter, previously developed projects on object detection will be introduced.

2.1 Using Deep Learning to Streamline Intracytoplasmic Sperm Injection in Cancer Patients [1]

In this project, the results of deep learning methods have been investigated in order to facilitate intracytoplasmic sperm injection in cancer patients. ICSI, a single sperm is taken within a thin glass needle and injected into each of the eggs in the laboratory. In practice, a small number of sperm is required and the ability of sperm to penetrate the egg is increased by the ICSI technique. The sperm cell selected in this method, should be motile and of good quality. This sperm selection is done by an expert and it takes time. By using deep learning methods, they aimed to do this job quickly and with high accuracy. This project uses YOLO(v3) and Faster R-CNN as object detection models. The dataset they use includes 1110 pictures and one or more sperm cells in each picture. These sperm cells are labeled as normal and abnormal. To expand the dataset, data augmentation techniques has been utilized. After the data augmentation process, they had 4440 tagged images.

They used 120 epochs and various training parameters while training the YOLO model. Each epoch took an average of 90 seconds. The best mAP value they got in the YOLO model was 0.37. In the Faster R-CNN model, they determined 50 epochs and the epoch length as 1000. Each epoch took an average of 720 seconds. The best mAP value they got in this model was 0.56.

According to these results, the Faster R-CNN model has a higher mAP value than the YOLO model in their education. However, they came to the conclusion that the YOLO model was too fast in terms of training time to the Faster R-CNN model. They suggested that the data set could be increased for future studies and better results could be obtained if they had more normal sperm cell labels.

2.2 A preliminary study of sperm identification in microdissection testicular sperm extraction samples with deep convolutional neural networks [2]

In this project, a new computer-aided sperm analysis system is introduced which also utilizes deep learning to achieve near human-level performance on testicular sperm extraction(TESE), trained on a custom dataset. The dataset comprises 702 de-identified images from testicular biopsy samples of 30 patients. All of the images were normalized and passed through glare filters and diffraction correction. The data were split 80%, 10% and 10% into training, validation and test sets, respectively.

The deep learning model has been used in this model, employs MobileNetV2 and SSD. The MobileNet is a feature extraction network for fast predictions on mobile devices. It's an updated version composed of a fully convolutional layer with thirty-two 3x3 filters prepended to 19 residual bottleneck layers. SSD on the other hand, is an object detection network which makes the use of the convolutional layers of VGG16, appending additional convolutional layers, and extracting ffeature maps at each layer for prediction. The feature maps exist in a variety of sizes which allows the predictions to be made in various scales.

The dataset is labeled by embryologists. So the model was benchmarked against embryologists' performance on the detection task, to compare the performance of the suggested model against human error rates. The proposed deep learning CASA system achieved a mAP of 0.741 with an average recall AR of 0.376 on the presented dataset.

2.3 DeepSperm: A robust and real-time bull sperm-cell detection in densely populated semen videos [3]

In this project, a study was conducted to detect bull sperm in videos. Bull sperm cells are dense, small and difficult to detect because they move very quickly. In this study, they tested the method they developed as well as popular object detection methods. They called the method they developed themselves DeepSperm. They compared the DeepSperm method with the popular object detection methods YOLO (v3) and Mask R-CNN.Its main goals were limited accuracy, high computational cost, and limited annotated training data.

As for the dataset, they used 6 bull sperm videos varying between 15 seconds and 124 seconds in length. The dataset has been splitted, 80% for training and 20% percent for testing. In total, there were 18,882 sperm cells in the training dataset, 4,728 in the validation dataset, and 3,174 in the test dataset.

At the end of the training, they calculated the mAP value for Mask R-CNN as 64.77 and for YOLO as 67.78.In the DeepSperm method, which is their method, they calculated the mAP value as 84.54.In addition, they found that the Mask R-CNN method was slower than the YOLO method in their study.Mask R-CNN used 11.7 GB ram while YOLO used 7.4 GB ram.

As a result, they said that the YOLO method is better than the Mask R-CNN method in terms of accuracy and speed. However, the DeepSperm method they recommended obtained better results than these two methods.

2.4 Segmentation of COVID-19 pneumonia lesions: A deep learning approach [4]

In this study, Detectron2, one of the deep learning methods, was used to detect pneumonia lesions caused by COVID-19 disease, which is still effective today. The most effective method to detect COVID 19 disease earlier and more precisely is a lung CT scan.

They used lung CT scans of 55 COVID19 patients and 41 healthy individuals to train the model. They used a total of 2469 CT scan slices containing 1402 manually segmented abnormal and 1067 normal slices from these lung CT scans. They used scans of 1224 lung CT slices from 18 COVID19 patients and 9 healthy individuals to test the model.

As a result, they found its accuracy to be 0.95, its sensitivity to 0.92, its specificity to 0.96 and IoU metric as 0.86. They achieved an acceptable success rate in this study. They suggested that, thanks to this study, abnormal conditions could be detected quickly in chest CT scans of patients, and any changes that occurred in patients could be easily followed.

3
Feasibility Study

Feasibility Studies are one of the most important steps to get started on a project. These studies and its sub-branches will directly contribute to the success and efficiency of the project. While doing these studies, we will examine the technical details to be used in our project. As a result of the feasibility studies, the costs of our project will be determined approximately.

3.1 Technical Feasibility

In this section, the technical feasibility of the planned project will be examined under sub-headings.

3.1.1 Software Feasibility

To perform well in deep learning operations, one would need to utilize the GPU to its limits. To be able to do that, we employed Nvidia's CUDA which is a parallel computing architecture and CuDNN library for optimizing back and forward convolution, pooling and normalization for detection. To do the deep learning operations we decided to subscribe to Google Colab to use its powerful GPU servers.

In this project, Python is used as the main programming language since it's supported by Google Colab and it offers a variety of libraries, fully supported by the developers and it has a large community in deep learning field.

3.1.2 Hardware Feasibility

Machine learning operations, especially deep learning operations requires a well-performing GPU. The amount of computers needed for this project is two. The computer's features are;

1. Intel(R) Xeon(R) CPU @ 2.20GHz
2. Nvidia Tesla T4
3. 12 GB of RAM

3.1.3 Time Feasibility

Time planning is important in terms of the planned and systematic progress of the project. In addition, by looking at this plan in the future, the remaining works in the project can be determined, preventing the disruption of the project's end date and having an idea about the general course of the project. In this section, the time feasibility analysis of the project has been made and it is expressed as a Gantt diagram in 3.1.

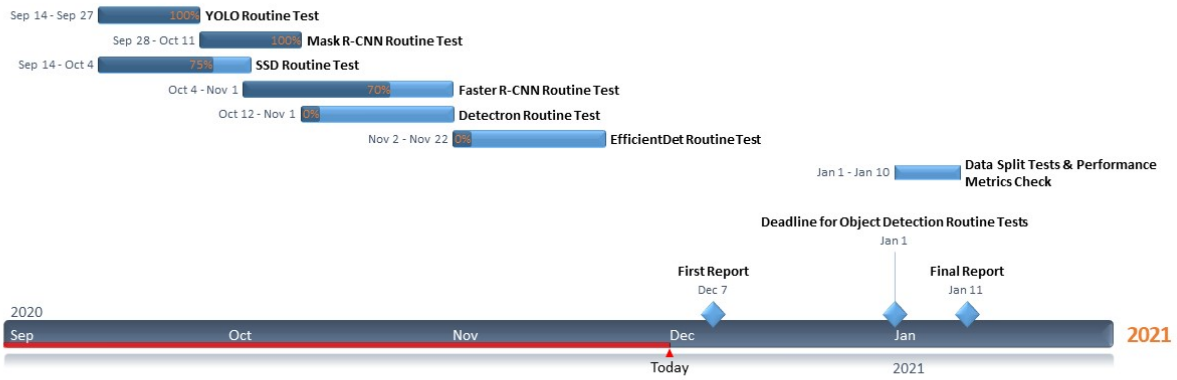

Figure 3.1 Gantt Diagram

3.2 Economic Feasibility

The cost of the project is examined in three main titles.

3.2.1 Hardware Costs

The hardware cost of the project is as in 3.1.

Table 3.1 Hardware price information

Name	Brand	Quantity	Cost (TL)
Computer	Dell G3 3590	1	8579 TL
Computer	MSI GE60 0ND	1	2500 TL
		Sum	11079 TL

3.2.2 Software Costs

Every object detection method that we have used in this project is free and open source. We will use Google Colab for three months. The software costs of the project is as in 3.2.

Table 3.2 Software price information

Name	Functionality	Quantity	Mon. Price (TL)	Price (TL)
Ubuntu 18.04	OS	2	Free	Free
Gedit	Text Editor	2	Free	Free
Object Detection Algs.	Methods	-	Free	Free
Google Colab Subs.	Server	2	156 TL	468 TL
			Sum	468 TL

3.2.3 Employee Costs

The hardware cost of the project is as in 3.3.

Table 3.3 Employee Costs in this project

Name	Weekly Working Time	Project Time	Salary	Total Cost
Oussama Mizaoui	13 Hours	3 Months	3000TL	9000TL
Özgür Kan	13 Hours	3 Months	3000TL	9000TL
			Sum	18000TL

3.3 Legal Feasibility

Most of the programs used in the project are open source codes. Personal data set was not used in this project. Also this project is supported by Yildiz Technical University.

4
System Analysis

4.1 Use-Case Scenario

1. The dataset should have only two labels. Sperm and Non-Sperm.
2. The user arranges the dataset and gives it as an input for the model to the system.
3. The user selects an object detection model to train.
4. The system processes the images and its labels and passes each labeled image to the selected object detection model.
5. After training, the user gives a video as an input for the system. The system will run the trained model on the video and it will return a new video with the detected sperms covered by a bounding box. In the new video, the percentage of the estimates will be shown as well.

The use-case scenario diagram is provided at Figure 4.1 below.

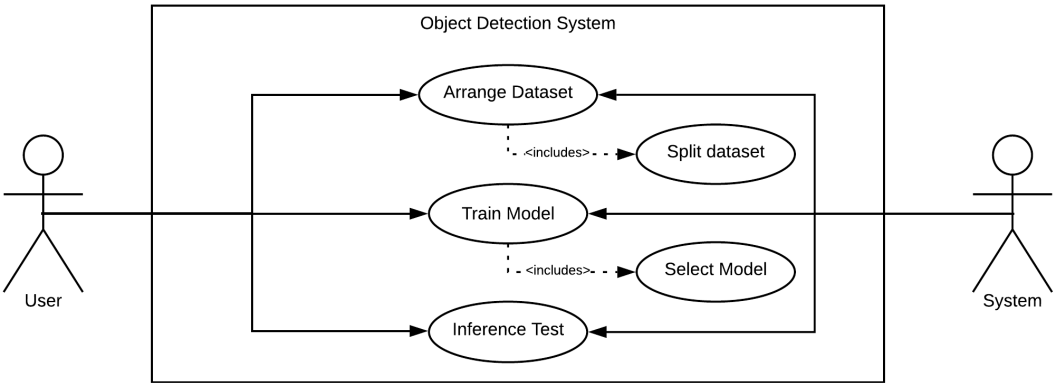

Figure 4.1 Use case diagram

4.2 Performance Metrics

The data-set will be divided in three different ways;

1. %20 Train - %80 Test
2. %50 Train - %50 Test
3. %80 Train - %20 Test

At the end of this training, evaluations will be made using mAP values and the training time. The object detectors will be trained with the split data-set. At the end of these evaluations, the most successful video, in the object detection algorithm we use, will only be used for training, and the remaining videos will be used for testing. Thus, it will be observed how the videos that are not in training will degrade during the test phase.

The object detectors will be compared based on average training time and mAP values.

4.3 General Flowchart

The general flowchart is provided at Figure 4.2 below.

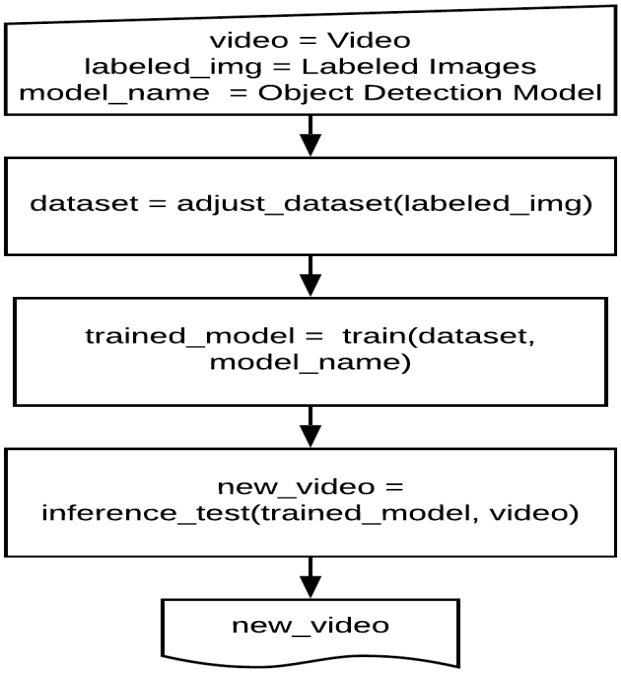

Figure 4.2 General flowchart

5
System Design

In this section, detailed information about the object detection algorithms we use in the project will be given.

5.1 YOLO Architecture

5.1.1 YOLOv1

YOLOv1 was announced by Joseph Redmon et al in an article published in May 2016.[5] It means "You Only Look Once". Many object detection algorithms require the input image to be passed through the object detection network multiple times, or two stages are required for object detection. In the YOLO algorithm, the input picture is passed through the object detection network only once. This is the main difference of YOLO from other networks and is faster than other models due to the single processing of the input picture.

YOLO divides the input image into a 7x7 grid. If the object coincides with the center of one of these grids, that grid is responsible for detecting that object. YOLO runs a classification and localization problem to each of the 7x7=49 grid cells simultaneously. The main problem here is that a grid cell can only detect one object. Another problem is that since it has 49 grids, it can detect a maximum of 49 objects in a picture. Another important problem is that if the object to be detected covers more than one grid, this object cannot be detected.Each cell predicts 2 bounding boxes and for each box the model gives a confidence score. This score shows how accurate the detected object is. Boxes with low confidence scores are not taken into account.

YOLO uses a single convolutional network to simultaneously predict multiple bounding boxes and classify probabilities for those boxes. YOLO uses only 1×1 shrink layers followed by a 3×3 convolutional layer. It has 24 convolutional layers followed by 2 fully connected layers. The final output of the network is the $7 \times 7 \times 30$ tensor.

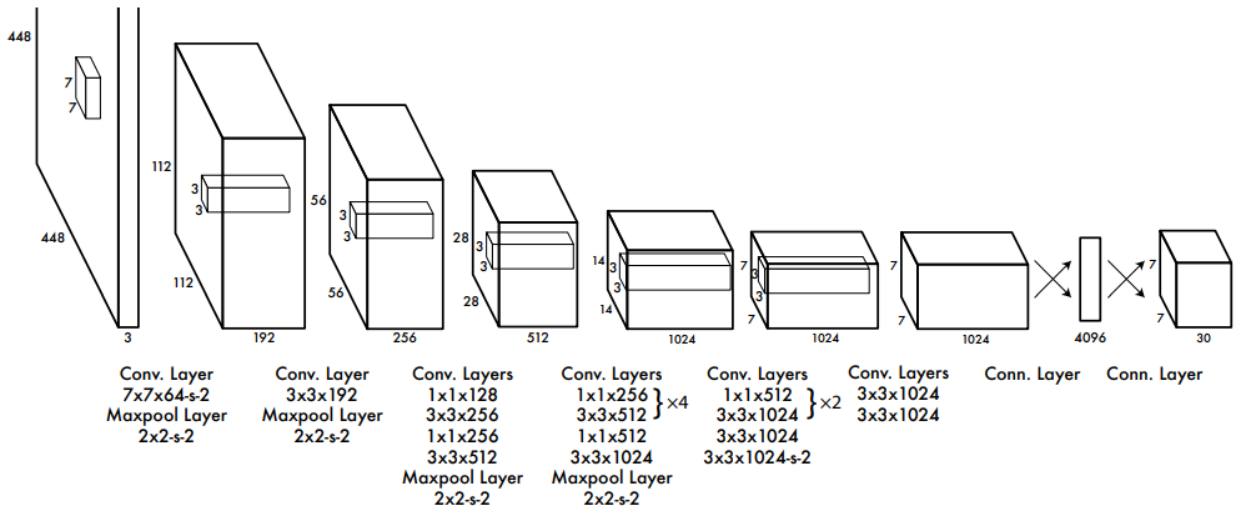

Figure 5.1 YOLO Network Design [5]

5.1.2 YOLOv2

Joseph Redmon and her colleagues introduced YOLOv2 in the article "YOLO9000: Better, Faster, Stronger" in December 2017.[6] In the second version of YOLO, they focused mainly on improving recall and localization while maintaining classification accuracy. By adding bulk normalization to all convolutional layers, they achieved a 2 percent increase in mAP. They set the resolution to 224x224 while training the network, and increased the resolution to 448 when testing.Detecting only one object in a single grid was a problem. To fix this problem, they used k bounding boxes to detect multiple objects. To predict k bounding boxes YOLOv2 used the idea of Anchor boxes. To encounter the problem of complexity and accuracy the authors propose a new classification model called Darknet-19 to be used as a backbone for YOLOv2.Darknet-19 has 19 convolutional layers and 5 maxpooling layers. The final output of the network is the 13x13x125 tensor.

5.1.3 YOLOv3

Joseph Redmon and Ali Farhadi released the new version with their article "YOLOv3: An Incremental Improvement" in April 2018.[7] YOLOv3 predicts an objectness score for each bounding box using logistic regression. YOLOv3 does not use a softmax; instead, it simply uses independent logistic classifiers for any class. During training, they used binary cross-entropy loss for the class predictions. It has worse performance on medium and large sized objects, but is more successful in detecting small objects. It uses a new network for feature extraction and is called Darknet-53. Performance drops significantly as the IOU threshold increases, indicating that YOLOv3 struggles to get the boxes perfectly aligned with the object, but it still faster than other versions.

5.1.4 YOLOv4

On April 23, 2020, Alexey Bochkovskiy and her colleagues announced YOLOv4 with the article "YOLOv4: Optimal Speed and Accuracy of Object Detection".[8] The difference of YOLOv4 from YOLOv3 is only the backbone. Because the YOLOv3 has Darknet53 backbone. But the YOLOv4 has CSPDarknet53. All the other things are the same when it's compared with YOLOv3.

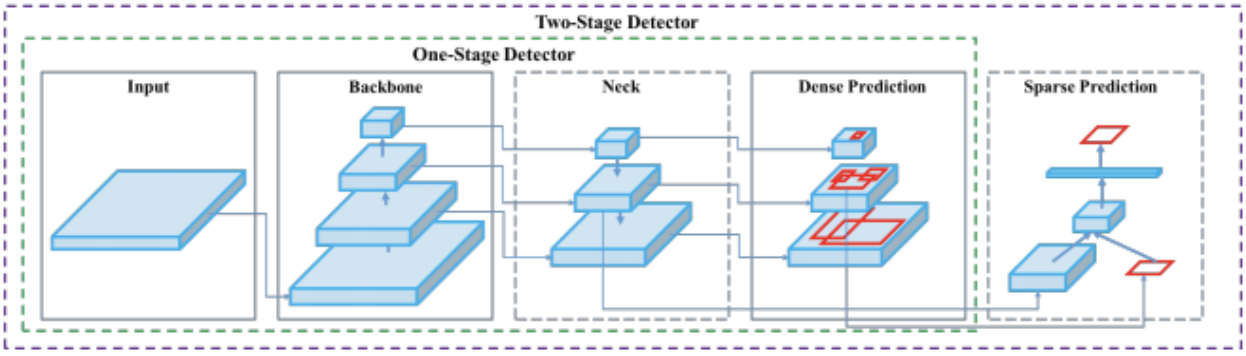

Figure 5.2 Object Detection Architecture [8]

5.1.5 YOLOv5

On May 29, 2020, Glenn Jocher created a repository called YOLOv5, which does not contain any model code, and added a continued commitment message "YOLOv5 greetings" to the YOLOv3 application on June 9, 2020. Jocher's YOLOv5 implementation differs from previous versions in several important ways. Jocher has not (yet) published an article. While Jocher implements YOLOv5 natively in PyTorch, all previous models in the YOLO family take advantage of the Darknet. [9] There are 4 different models in the repository: YOLOv5s, YOLOv5m, YOLOv5l, YOLOv5x. The first is the smallest and least accurate, the last one is the largest with the greatest accuracy. All models run on PyTorch.

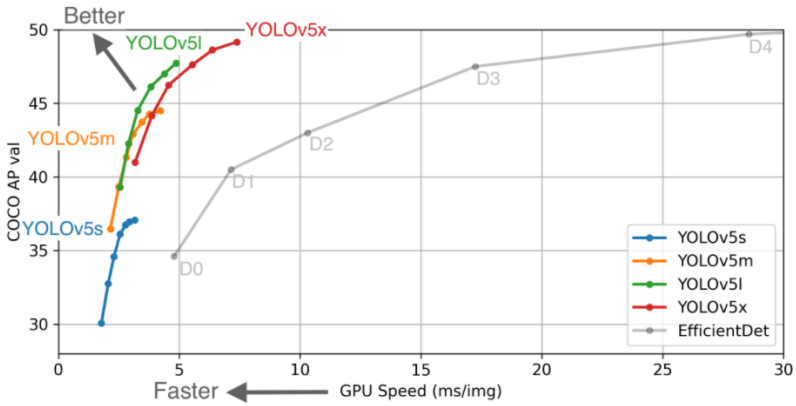

Figure 5.3 Comparision YOLOv5 Models [9]

YOLOv5 is extremely fast. A YOLOv5 Colab laptop running Tesla P100 achieves times of up to 0.007 seconds per image i.e. 140 frames per second. In contrast, YOLOv4 achieved 50 FPS after being converted to the same Ultralytics PyTorch library. Third, YOLOv5 is correct. In tests on the blood cell count and detection dataset, an average mean sensitivity (mAP) of about 0.895 was obtained after only 100 periods of training. It is rare to see such extensive performance improvements without any loss in accuracy.YOLOv5 is small. Specifically, a weights file for YOLOv5 is 27 megabytes. Our weights file for YOLOv4 (with Darknet architecture) is 244 megabytes. YOLOv5 is nearly 90 percent smaller than YOLOv4. This means YOLOv5 can be deployed to embedded devices much more easily.The largest YOLOv5 is YOLOv5x, and its weights are 367 MB.

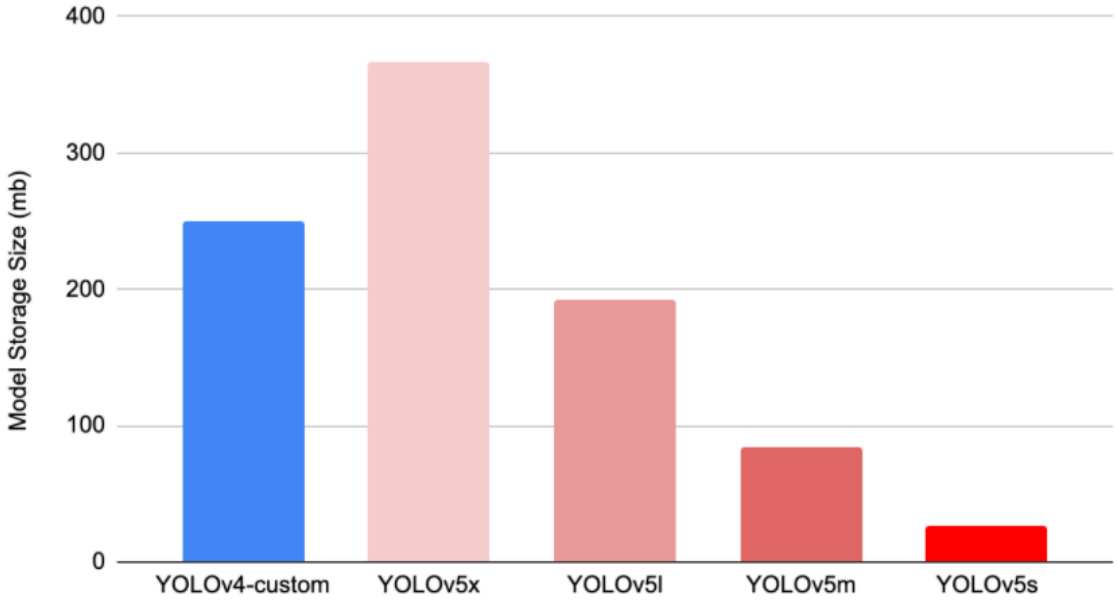

Figure 5.4 Model Storage Size (MB) [9]

As a result, in our project, we used YOLOv5, the latest version of YOLO. Among the YOLOv5 models, although the model size is high, we used the YOLOv5x model in our project because speed and accuracy are important parameters for us. The YOLOv5x model has been the best choice for us in terms of training speed.

5.2 R-CNN Architecture

R-CNN was announced by Ross Girshick in 2013 where he combines region proposals with CNNs hence the name comes from it. It combines two key insights;

1. One can apply high-capacity convolutional neural netwroks to bottom-up region proposals in order to localize and segment objects.
2. When labeled training data is scarce, supervised pre-training for an auxiliary task, followed by domain-specific fine-tuning, yields a significant performance boost.

It has a simple yet effective way to do detection. It takes an input image and it extracts around 2000 bottom-up region proposals. With them, it computes features for each proposal using a large CNN and it classifies each region using class-specific linear SVMs.

5.2.1 Fast R-CNN Architecture

Fast R-CNN have reduced the running time of R-CNN by basically solving couple of drawbacks that the former version had and adding to that, it also improved its speed and accuracy. It had these following advantages;

1. Higher detection quality (mAP) than R-CNN.
2. Training is reduced to be a single-stage, using a multi-task loss.
3. Training can update all network layers. With that the training time is reduced.
4. No disk storage is required for feature caching, this is good for memory saving.

It takes an input image and a set of object proposals. The network first processes the whole image with several convolutional and max pooling layers to produce a feature map. Then, for each object proposal a region of interest (RoI) pooling layer extracts a fixed-length feature vector from the feature map.

Each feature vector is fed into a sequence of fully connected layers that finally branch into softmax probabilities and per-class bounding box regression offsets as its output.

5.2.2 Faster R-CNN Architecture

Faster R-CNN was announced by Shaoqing Ren et al. in an article published in 2015. It's an upgraded version of R-CNN and Fast R-CNN. It has been realized with upgrading to Fast R-CNN that the region proposal computation is the main bottleneck. With this, a new region proposal network is introduced that shares full-image convolutional features with the detection network, thus enabling nearly cost-free region proposals. It merges RPN and Fast R-CNN into a single network by sharing their convolutional features.

5.2.3 Mask R-CNN Architecture

Mask R-CNN was announced by Kaiming He et. al. in an article published in 2018. It simply, extends Faster R-CNN by adding a branch for predicting an object mask in parallel with the existing branch for bounding box recognition and it does it with a small overhead to Faster R-CNN. The mask branch is a small FCN applied to each region of interest, predicting a segmentation in a pixel-to-pixel manner.

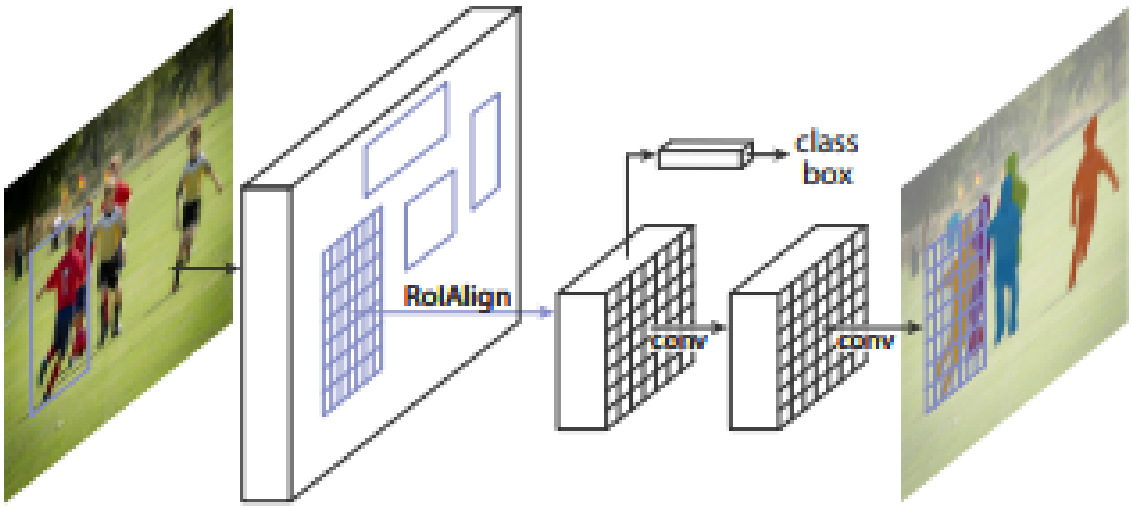

Figure 5.5 Mask R-CNN Framework [10]

5.3 Detectron2 Architecture

Detectron2 was introduced by the FAIR Facebook AI Research group in October 2019.[11] Detectron2 is a computer vision model written in PyTorch. It is a ground-up rewrite of the previous version, Detectron, and it originates from maskrcnn-benchmark.

Detectron2 is flexible and extensible, and able to provide fast training on single or multiple GPU servers. Detectron2 includes high-quality implementations of state-of-the-art object detection algorithms, including DensePose, panoptic feature pyramid networks and numerous variants of the pioneering Mask R-CNN model family also developed by FAIR.

Detectron2 contains improved versions of Faster R-CNN, Mask R-CNN, RetinaNet, and Denspose. Detectron2 also has new models including Cascade R-CNN, Panoptic FPN, and TensorMask. Detectron2 is utilized to perform keypoint detection, object detection or semantic segmentation. It registers the datasets in COCO JSON format.

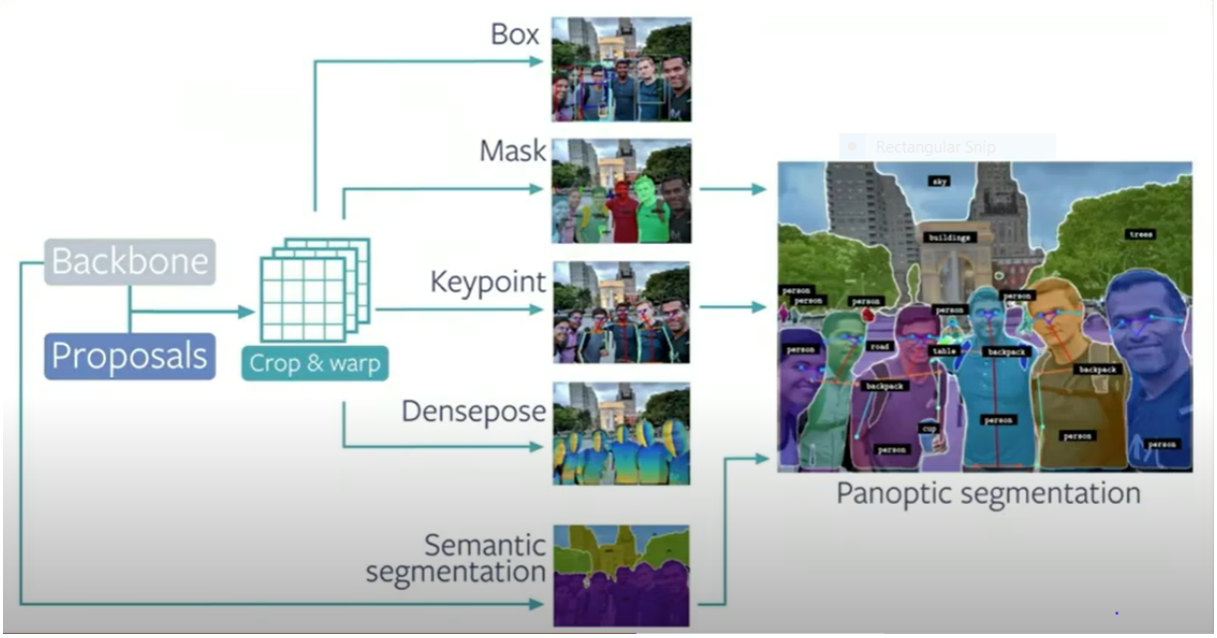

Figure 5.6 Detectron2: Generalized R-CNN Models [11]

5.4 EfficientDet Architecture

The EfficientDet architecture was introduced by Google Brain in July 2020 with the article "EfficientDet: Scalable and Efficient Object Detection".[12] EfficientDet built on top of EfficientNet, a convolutional neural network that is pretrained on the ImageNet image database for classification. EfficientDet pools and mixes portions of the image at given granularities and forms features which are passed through a NAS-FPN feature fusion layer. The NAS-FPN combines various features at varying granularities and passes them forward to the detection head, where bounding boxes and class labels are predicted.

EfficientDets was developed based on an improved backbone, a new BiFPN, and a new scaling technique. EfficientNets are used as backbone networks. BiFPN, a two-way feature network developed with rapid normalization, and providing easy and fast feature aggregation has been proposed. A single composite scaling factor was used to manage depth, width, and resolution for all backbone, feature, and prediction networks.

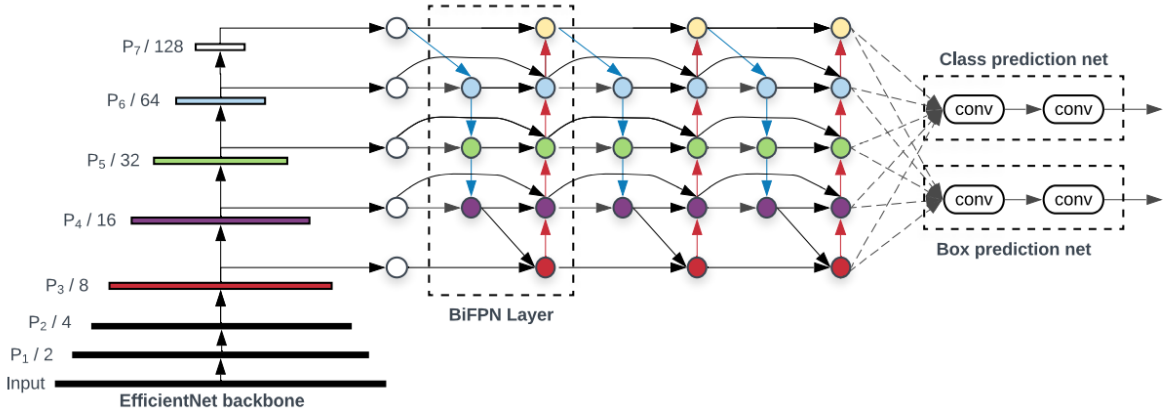

Figure 5.7 EfficientDet Architecture [12]

5.5 SSD - Single Shot MultiBox Detector Architecture

The SSD architecture was introduced by Wei Liu et al. in December 2016 with the article "SSD: Single Shot MultiBox Detector" [13].

This method is faster than YOLO and as accurate as slower techniques that produce region proposals. The approach, discretizes the output space of bounding boxes into a set of default boxes over different aspect ratios and scales per feature map location.

At prediction time, the network generates scores for the presence of each object category in each default box and produces adjustments to the box to better match the object shape. Additionally, the network combines predictions from multiple feature maps with different resolutions to naturally handle objects of various sizes.

SSD acts similarly to methods that require object proposals because it completely eliminates proposal generation so in a way, it's acting like R-CNN without region proposal. This makes it easy to train and straightforward to integrate into systems that require a detection component.

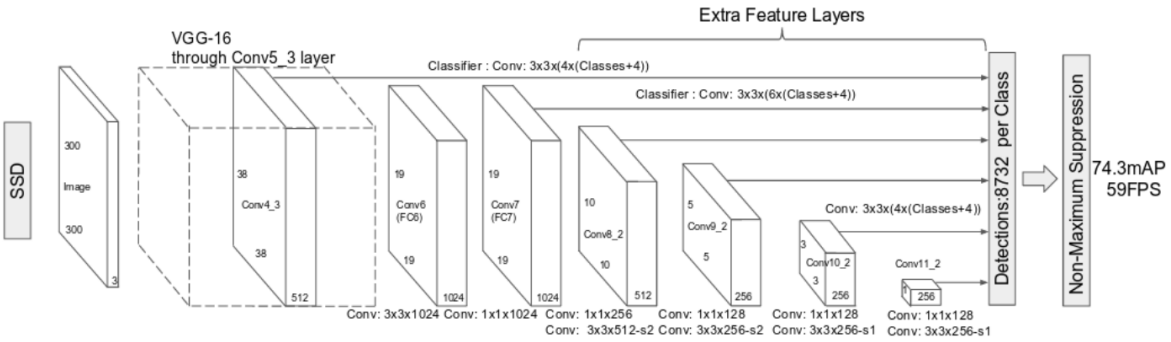

Figure 5.8 SSD Architecture

6
Application

In the application section, we have chosen 6 different methods for object detection. These methods are; YOLOv5, Mask R-CNN, Detectron2, Faster R-CNN, SSD and EfficientDet. We chose these methods because they are widely preferred for object detection. Some methods have recently been developed using new technologies and not enough studies have been done, we wanted to see how successful these methods would be in detecting small objects such as sperm. In addition, we wanted to determine which of these methods are faster and we used these methods in our study.

The dataset we used in our project was given to us by our consultant. There are 13 videos in this dataset. 13 videos were cropped into five frames per second and images extracted. These images have been previously tagged as sperm and not sperm by an arbitrary labeling tool. Since the image sizes are large, the image sizes are set to 640 x 640 in each architecture. Care has been taken not to shift the tagged data while doing this sizing. Also, since each architecture accepts different labeling formats, conversions have been made between these formats. Data augmentation or other sample multiplier methods were not used on the images.

Table 6.1 Number of labels for each video

Video1	2871
Video2	3362
Video3	4086
Video4	1269
Video5	3606
Video6	2161
Video7	2890
Video8	3211
Video9	11004
Video10	2864
Video11	2369
Video12	6600
Video13	2720

In our first scenario, trainings were carried out for each model with 20% train 80% test, 50% train 50% test and 80% train 20% test. In our second scenario, the video pictures that gave the best results in the first scenario were used only for training, while the remaining videos were used for testing and the results were recorded. In our last scenario, the video pictures with the worst result in the first scenario were used only for testing. All other videos were used for training and it was observed whether the result of the video with the worst result in the first case improved.

6.1 YOLOv5 Application

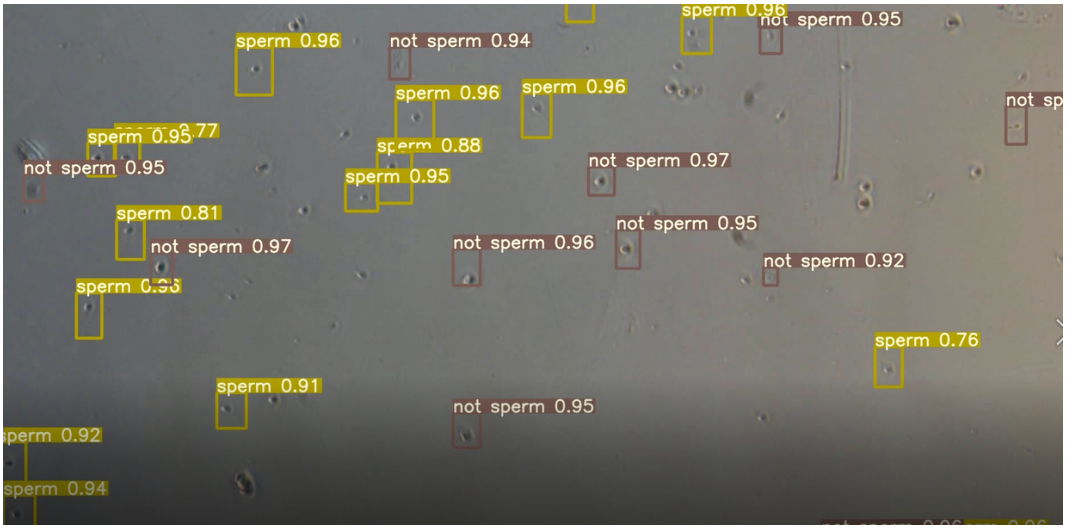

Figure 6.1 The best result in the YOLOv5 model

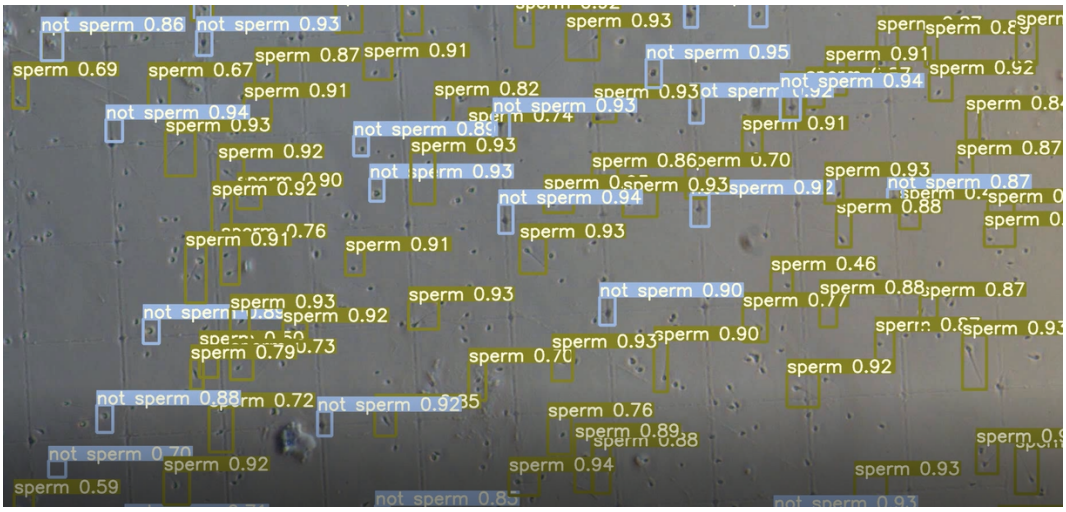

Figure 6.2 The worst result in the YOLOv5 model

6.2 Mask R-CNN Application

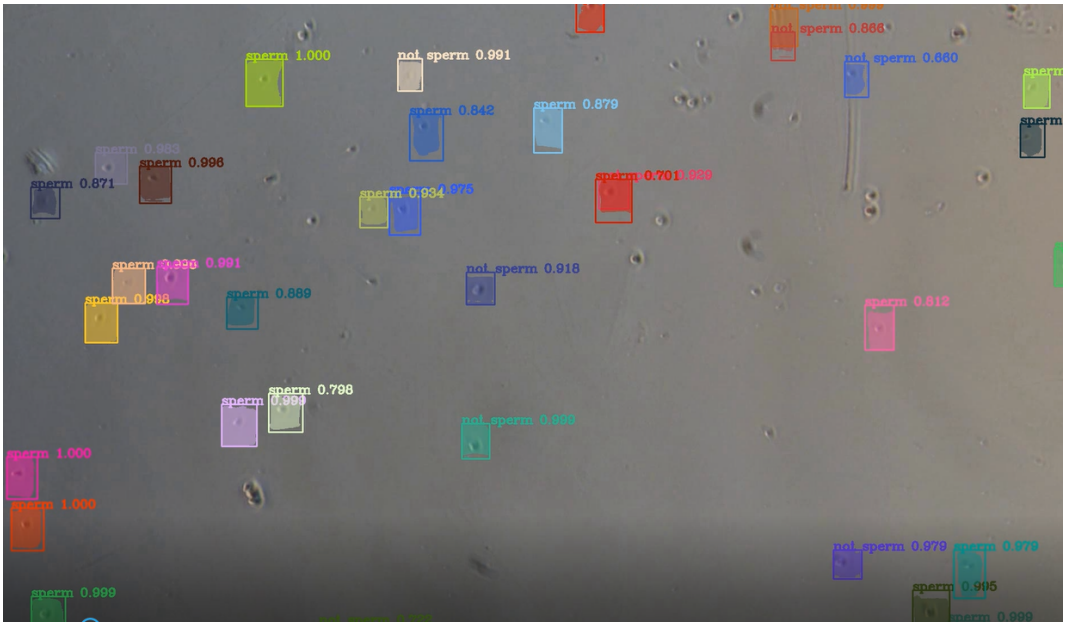

Figure 6.3 The best result in the Mask R-CNN model

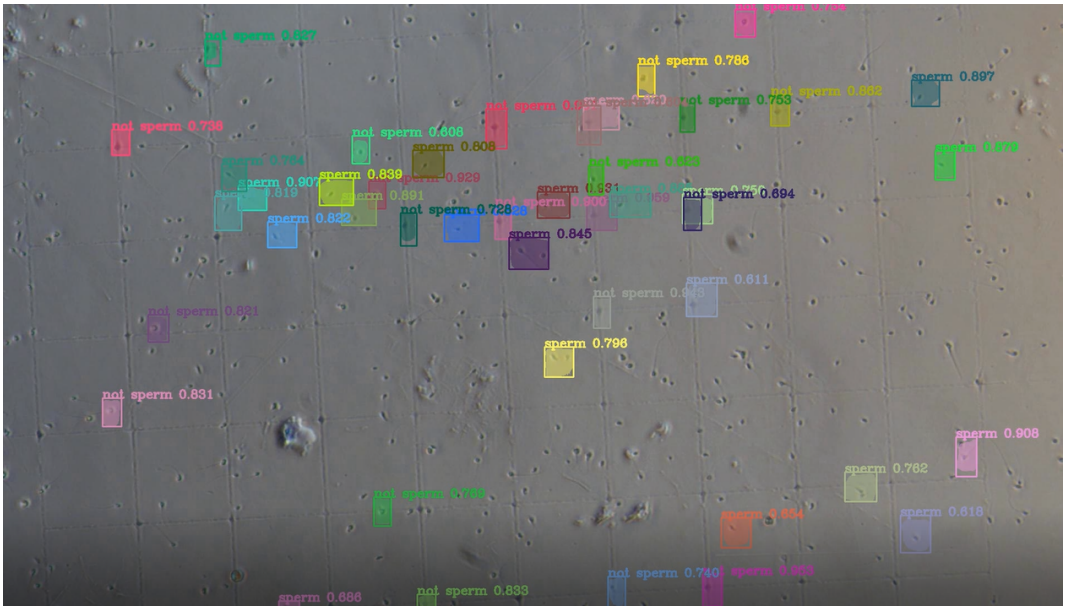

Figure 6.4 The worst result in the Mask R-CNN model

6.3 SSD Application

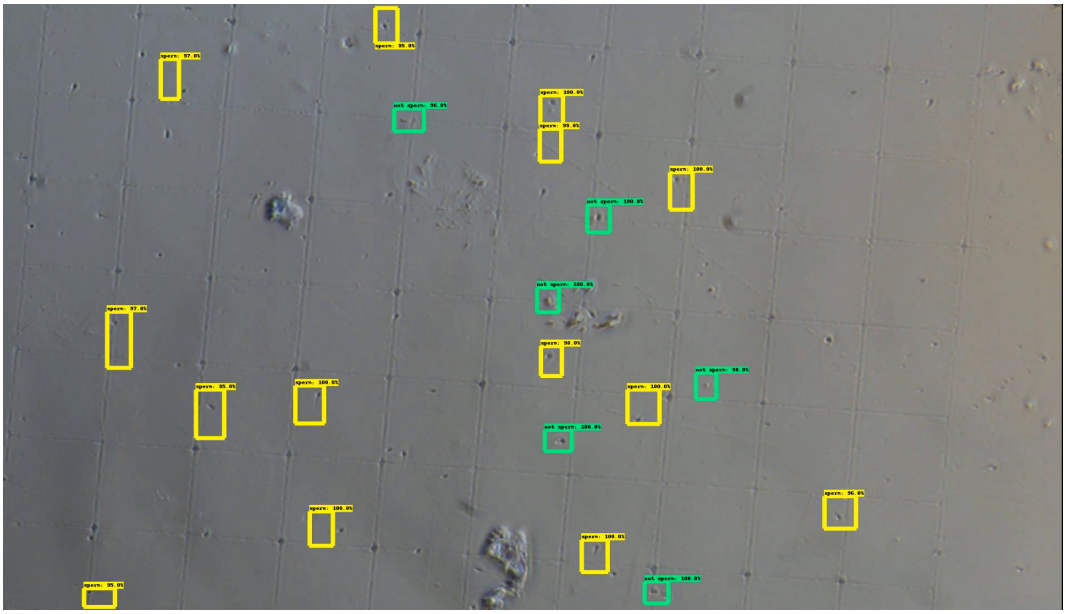

Figure 6.5 The best result in the SSD model

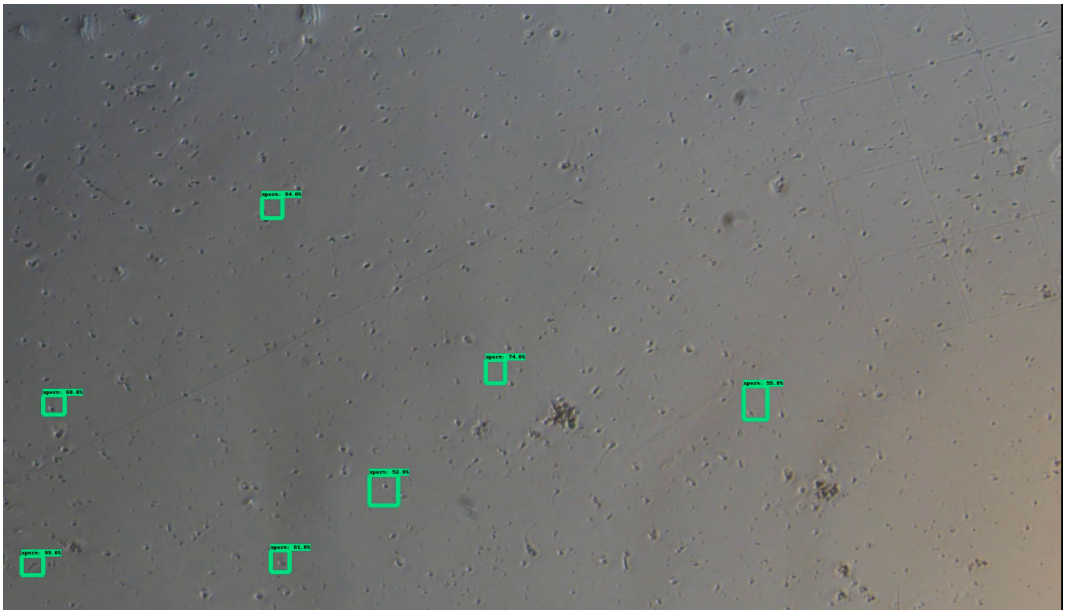

Figure 6.6 The worst result in the SSD model

Note: There are only 3 methods' results are given. Our trainings and studies on the other methods still continue. These methods will be added to the application section in the final report and a detailed analysis for all of them will be made.

References

[1] M. Aggarwal, V. S. Nair, and T. Sun, "Using deep learning to streamline intracytoplasmic sperm injection in cancer patients,"

[2] D. J. Wu, O. Badamjav, V. V. Reddy, M. Eisenberg, and B. Behr, "A preliminary study of sperm identification in microdissection testicular sperm extraction samples with deep convolutional neural networks," *Asian Journal of Andrology*, 2020.

[3] P. Hidayatullah, X. Wang, T. Yamasaki, T. L. Mengko, R. Munir, A. Barlian, E. Sukmawati, and S. Supraptono, "Deepsperm: A robust and real-time bull sperm-cell detection in densely populated semen videos," *arXiv preprint arXiv:2003.01395*, 2020.

[4] Z. Ghomi, R. Mirshahi, A. Fattahpour, S. Mohammadiun, A. Alavi Gharahbagh, A. Djavadifar, H. Arabalibeik, R. Sadiq, K. Hewage, *et al.*, "Segmentation of covid-19 pneumonia lesions: A deep learning approach," *Medical Journal of The Islamic Republic of Iran (MJIRI)*, vol. 34, no. 1, pp. 1216–1222, 2020.

[5] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2016.

[6] J. Redmon and A. Farhadi, "Yolo9000: Better, faster, stronger," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jul. 2017.

[7] A. Farhadi and J. Redmon, "Yolov3: An incremental improvement," *Computer Vision and Pattern Recognition, cite as*, 2018.

[8] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "Yolov4: Optimal speed and accuracy of object detection," *arXiv preprint arXiv:2004.10934*, 2020.

[9] G. Jocher, *Yolov5*, `https://github.com/ultralytics/yolov5`, 2020.

[10] K. He, G. Gkioxari, P. Dollar, and R. Girshick, "Mask r-cnn," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Oct. 2017.

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

[12] M. Tan, R. Pang, and Q. V. Le, "Efficientdet: Scalable and efficient object detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2020.

[13] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. Berg, "Ssd: Single shot multibox detector," in *ECCV*, 2016.