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 — Usame İSLAMOĞLU

SENIOR PROJECT

Advisor
Assist. Prof. Dr. Hamza Osman İLHAN

January, 2021

ACKNOWLEDGEMENTS

We would like to thank Yıldız Technical University Computer Engineering Department for providing us the opportunity to do such projects and all of our teachers for everything they have contributed to us throughout our education life.

We are grateful to have Assistant Prof. Hamza Osman İLHAN as our supervisor. He provided us with the necessary sources and guided us throughout the work we have done.

Thanks to everyone who provided access to the methods and packages used in this project and supported the development of open source code.

We dedicate this work to our esteemed families who supported and stood by us throughout our education life.

Özgür KAN
Usame İSLAMOĞLU

TABLE OF CONTENTS

LIST OF ABBREVIATIONS

ICSI	Intracytoplasmic Sperm Injection
CNN	Convolutional Neural Network
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
RPN	Region Proposal Network
SSD	Single Shot Detector
YOLO	You Only Look Once
IOU	Intersection Over Union
CPU	Central Process Unit
GPU	Graphics Processing Unit
CuDNN	NVIDIA® CUDA® Deep Neural Network library™
ROI	Region Of Interest
mAP	Mean average precision
SSD	Single Shot MultiBox Detector

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

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

Özgür KAN
Usame İSLAMOĞLU

Department of Computer Engineering
Senior Project

Advisor: Assist. Prof. Dr. Hamza Osman İLHAN

Object detection algorithms today make things easier in many areas. One of the most important of them is the object detection in health field. Object detection algorithms are gradually being used to detect the effect of COVID-19 disease on the lung. As a result of these algorithms, diagnosis and curing is significantly accelerated. With the emerging technology, the rate of infertility increases in humans with the effect of the increasing radiation. The IVF method is seen as a laborious and difficult way. In order to facilitate this way, healthy sperm cells must be detected and the process must be performed.

In this project, by using object detection algorithms to detect sperm cells, it has been investigated which of these algorithms give more reliable results and which gives faster results in terms of speed. In this research, algorithms that have appeared recently for object detection were preferred. These algorithms were selected as YOLOv5, Mask R-CNN, Detectron2, SSD and EfficientDet and were implemented in the project. Using these algorithms in our project, we were able to identify very small sperm cells and observe their results.

Keywords: object detection, sperm, deep learning

ÖZET

DERİN ÖĞRENME TABANLI BÖLÜTLEME TEKNİKLERİ İLE HAREKETLİ / HAREKETSİZ SPERM HÜCRELERİNİN TESPİTİ

Özgür KAN
Usame İSLAMOĞLU

Bilgisayar Mühendisliği Bölümü
Bitirme Projesi

Danışman: Dr. Ögr. Üyesi Hamza Osman İLHAN

Günümüzde nesne tespiti algoritmaları birçok alanda işleri kolaylaştırmaktadır. Bu alanların en önemlisi sağlık alanı olarak gösterilebilir. Nesne tespiti algoritmaları günümüzde pandemi halinde olan covid-19 hastalığının akciğer üzerindeki etkisini tespit etmek için yavaş yavaş kullanılmaya başlamıştır. Bu algoritmalar sonucunda teşhis ve takip önemli ölçüde hız kazanmaktadır.Gelişen teknolojiyle birlikte artan radyasyonunda etkisiyle insanlarda kısırlık oranı da artmaktadır. Tüp bebek yöntemi zahmetli ve zor bir yol olarak görülmektedir.Bu yolu kolaylaştırmak için sağlıklı sperm hücrelerinin tespit edilip işlemin gerçekleştirilmesi gerekmektedir.

Bu projede sprem hücrelerini tespit etmek için nesne tespti algoritmaları kullanılarak bu algoritmalardan hangilerinin daha güvenilir sonuçlar verdiği ve hız açısından hangisinin daha hızlı sonuçlar verdiği araştırılmıştır. Bu araştırmada nesne tespiti için oldukça yakın zamanlarda çıkan algoritmalar tercih edilmiştir. Bu algoritmalar YOLOv5, Mask R-CNN, Detectron2, SSD ve Efficient-Det olarak seçilmiştir ve projede uygulanmıştır. Projemizde bu algortimaları kullanarak oldukça küçük olan sperm hücrelerini tespit etmeyi ve sonuçlarını gözlemlemeyi başardık.

Anahtar Kelimeler: nesne tespiti, sperm, derin öğrenme

1
Introduction

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

Computer-aided diagnosis has been in our lives for the past few decades. The system's responsibility escalated lately with new inventions and it started 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 crucial 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 are applied to detect sperm objects on 12 tagged sperm videos. These methods are selected as YOLOv5, SSD, Mask RCNN, Detectron2, 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 is introduced.

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 dataset could be increased for future studies and better results could be obtained if they had more normal sperm cell labels.

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.

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.

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 have examined the technical details to be used in our project. As a result of the feasibility studies, the costs of our project are determined approximately.

3.1 Technical Feasibility

In this section, the technical feasibility of the planned project is 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 Figure 3.1.

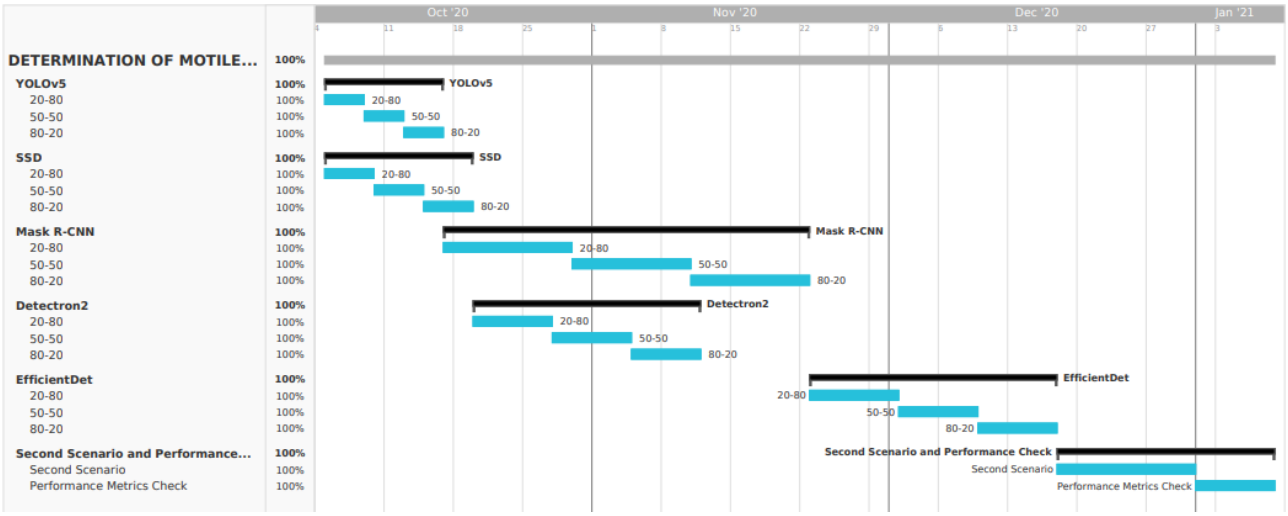


Figure 3.1 Gantt Diagram

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 Table 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 Table 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 Table 3.3.

Table 3.3 Employee Costs in this project

Name	Weekly Working Time	Project Time	Salary	Total Cost
Usame İslamoğlu	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 dataset was not used in this project. Also this project is supported by Yildiz Technical University.

4
System Analysis

Two different scenarios were used in our project. Detailed information about these scenarios is given below.

4.1 Scenario 1

Our first scenario is our basic scenario and it has been applied on 12 videos. In this scenario, each video was trained and tested with its own pictures. This scenario has been realized in 3 stages. In these 3 stages, the training and test videos of the datasets are divided at different rates. Our aim in this scenario is to evaluate the success of the samples taken from the patient's own video throughout the video.

4.1.1 %20 Train %80 Test (Scenario 1A)

In the first stage, %20 of the dataset was reserved for training and the remaining %80 for testing. This stage was performed for each video and the results were recorded.

4.1.2 %50 Train %50 Test (Scenario 1B)

In the second stage, %50 of the dataset was reserved for training and the remaining %50 for testing. This stage was performed for each video and the results were recorded.

4.1.3 %80 Train %20 Test (Scenario 1C)

In the third stage, %80 of the dataset was reserved for training and the remaining %20 for testing. This stage was performed for each video and the results were recorded.

4.2 Scenario 2

In the second scenario, 2 videos with the worst mAP value in the first scenario were selected as the target video. The remaining videos were used for training. Our aim in this scenario is to check whether the mAP values of these 2 videos with bad mAP values have improved.

4.2.1 %20 Train %80 Test (Scenario 2A)

In the first stage, %20 of the target video was reserved for training and the remaining %80 for testing. This stage was performed for each video and the results were recorded.

4.2.2 %50 Train %50 Test (Scenario 2B)

In the second stage, %50 of the target video was reserved for training and the remaining %50 for testing. This stage was performed for each video and the results were recorded.

4.2.3 %80 Train %20 Test (Scenario 2C)

In the third stage, %80 of the target video was reserved for training and the remaining %20 for testing. This stage was performed for each video and the results were recorded.

4.3 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.

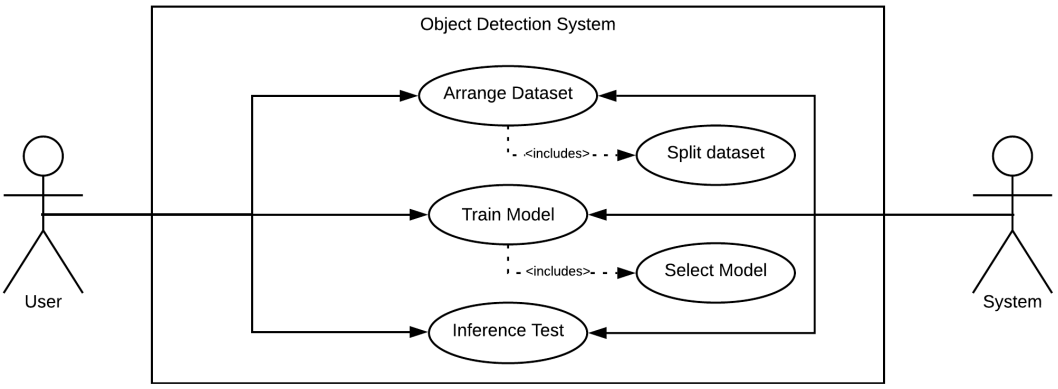


Figure 4.1 Use case diagram

4.4 General Flowchart

The general flowchart is provided at Figure 4.2 below.

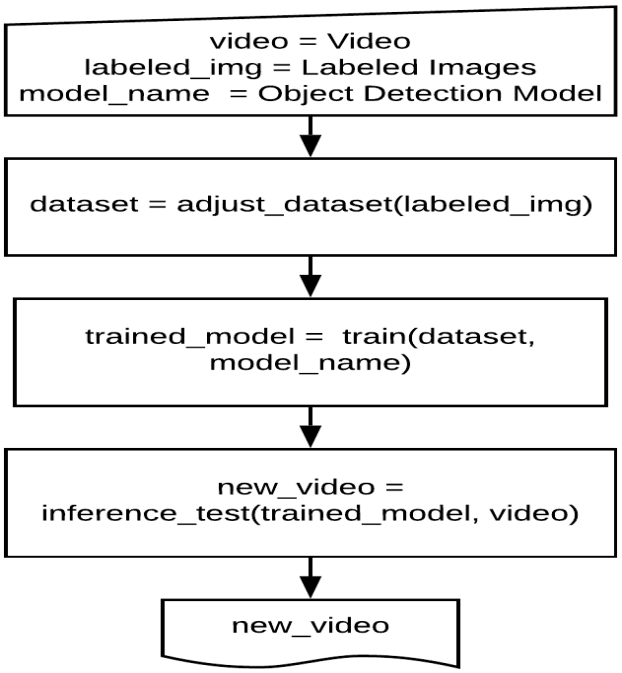


Figure 4.2 General flowchart

5
System Design

In this chapter, detailed information about the object detection algorithms we use in the project is 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 shows the YOLO architecture.

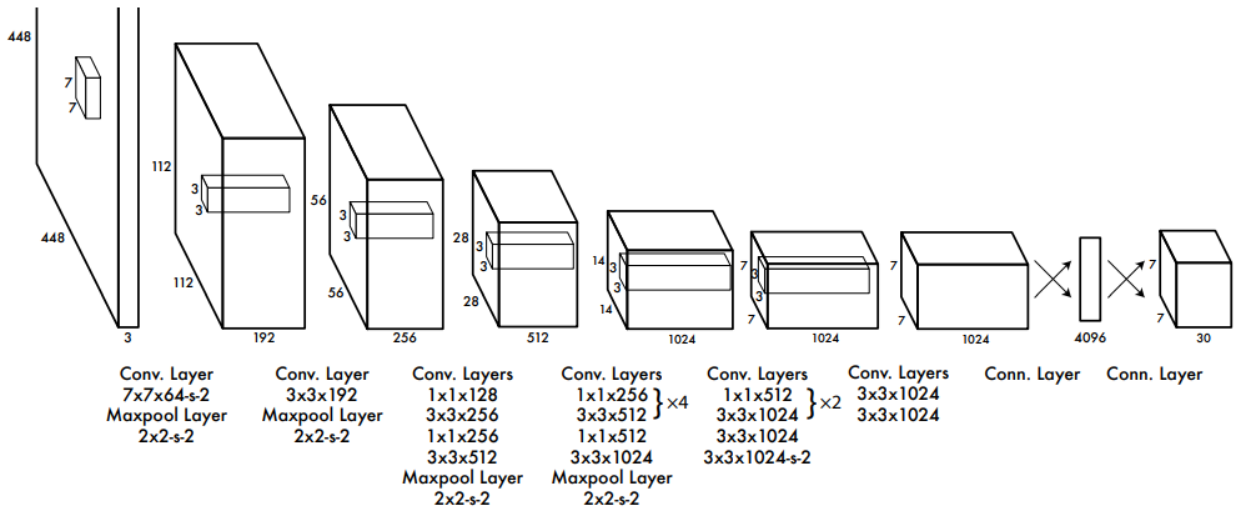


Figure 5.1 YOLO Network Design [5]

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 shows the object detection architecture.

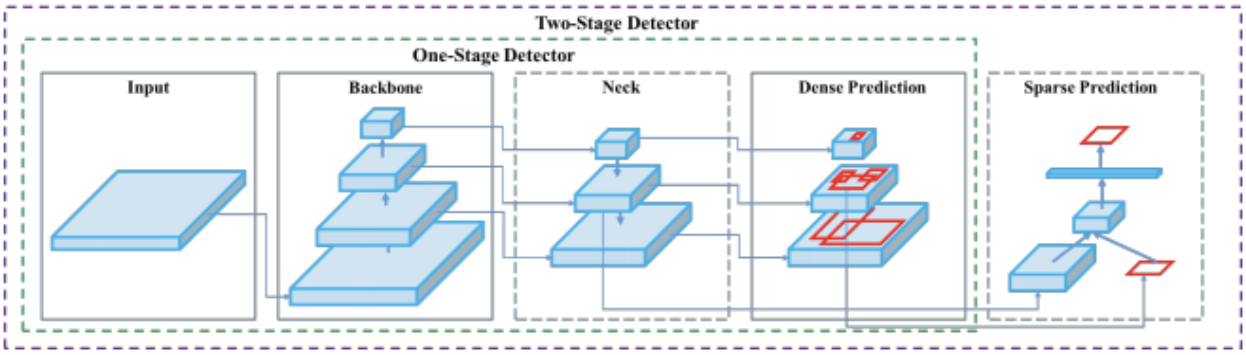


Figure 5.2 Object Detection Architecture [8]

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. [**YOLOv5**] 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 compares YOLOv5 architectures.

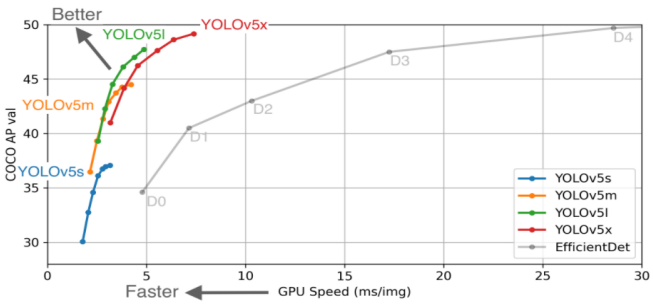


Figure 5.3 Comparision YOLOv5 Models [**YOLOv5**]

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 compares the sizes of the YOLOv5 models.

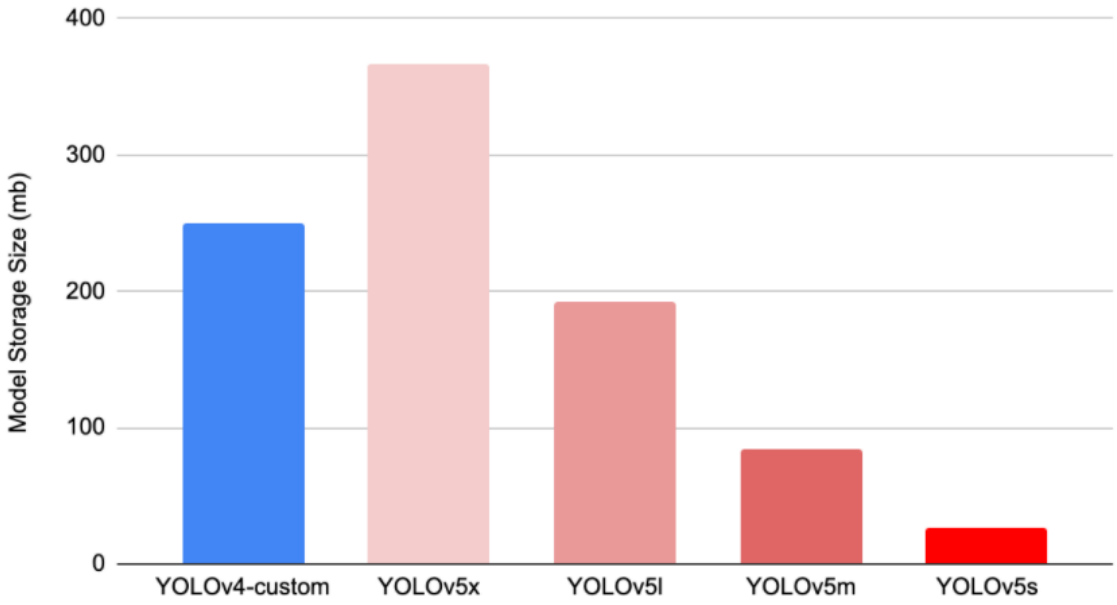


Figure 5.4 Model Storage Size (MB) [**YOLOv5**]

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

Region-based convolutional neural networks or regions with CNN features (R-CNNs) are a pioneering approach that applies deep models to object detection.R-CNN was announced by Ross Girshick in 2013 [9] 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.

Specifically, R-CNNs are composed of four main parts:

1. Selective search is performed on the input image to select multiple high-quality proposed regions. These proposed regions are generally selected on multiple scales and have different shapes and sizes. The category and ground-truth bounding box of each proposed region is labeled.
2. A pre-trained CNN is selected and placed, in truncated form, before the output layer. It transforms each proposed region into the input dimensions required by the network and uses forward computation to output the features extracted from the proposed regions.
3. The features and labeled category of each proposed region are combined as an example to train multiple support vector machines for object classification. Here, each support vector machine is used to determine whether an example belongs to a certain category.
4. The features and labeled bounding box of each proposed region are combined as an example to train a linear regression model for ground-truth bounding box prediction.

Although R-CNN models use pre-trained CNNs to effectively extract image features, the main downside is the slow speed. As you can imagine, we can select thousands of proposed regions from a single image, requiring thousands of forward computations from the CNN to perform object detection. This massive computing load means that R-CNNs are not widely used in actual applications. Figure 5.5 shows an R-CNN model.

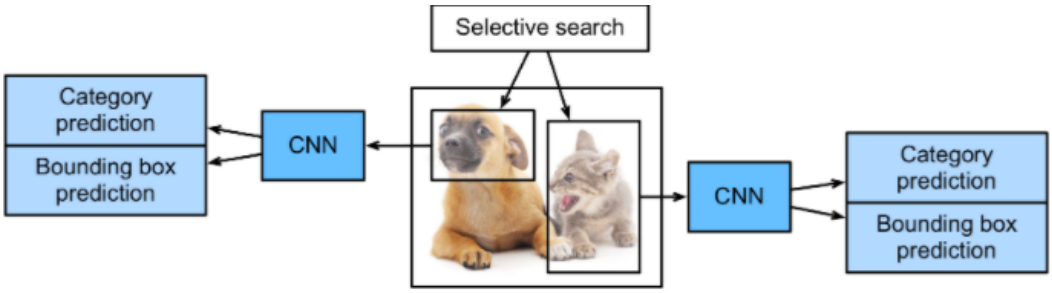


Figure 5.5 R-CNN Model

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 was announced by Ross Girshick and it builds on previous work. Compared to R-CNN, the running time is reduced by overcoming couple of drawbacks that the former version had and adding to that, it's speed and accuracy is also improved. It has 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. Figure 5.6 shows an Fast R-CNN model.

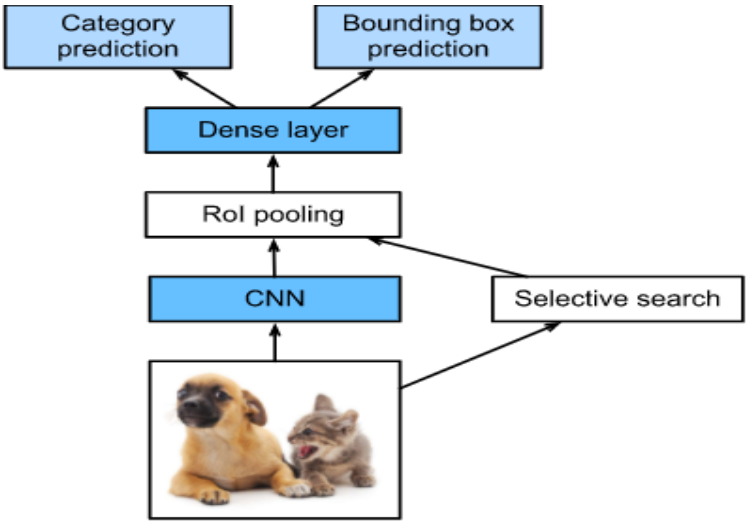


Figure 5.6 Fast R-CNN Model

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. Figure 5.7 shows an Fast R-CNN model.

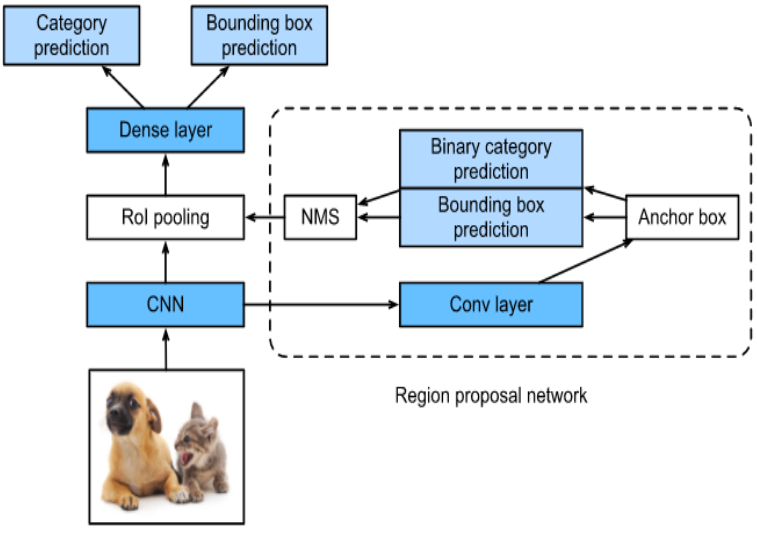


Figure 5.7 Faster R-CNN Model

5.2.3 Mask R-CNN Architecture

If training data is labeled with the pixel-level positions of each object in an image, a Mask R-CNN model can effectively use these detailed labels to further improve the precision of object detection. 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 5.8 shows an Mask R-CNN model.

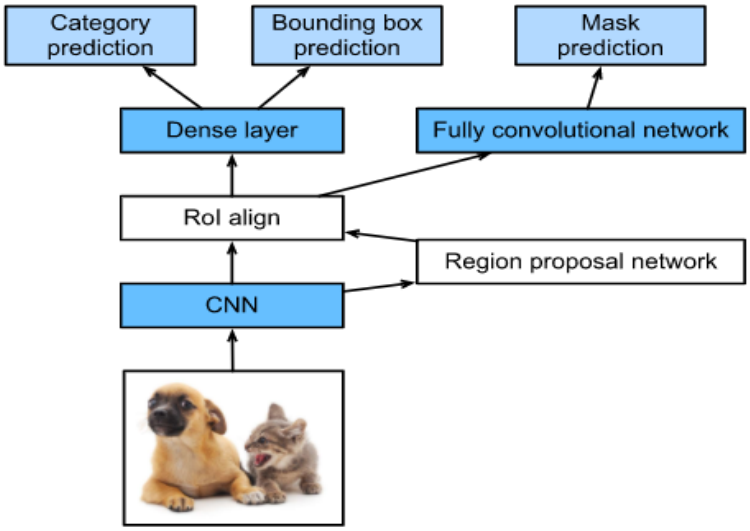


Figure 5.8 Mask R-CNN Model

5.3 Detectron2 Architecture

Detectron2 was introduced by the FAIR Facebook AI Research group in October 2019. [10] 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. Detectron2 architecture is shown in Figure 5.9.

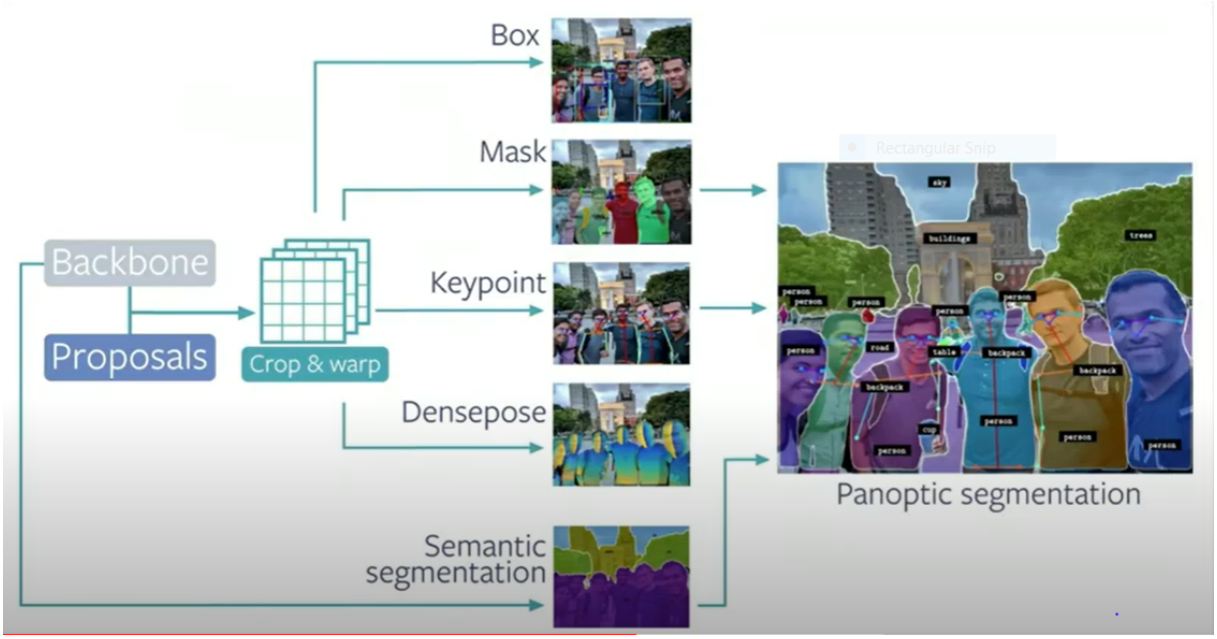


Figure 5.9 Detectron2: Generalized R-CNN Models [10]

5.4 EfficientDet Architecture

The EfficientDet architecture was introduced by Google Brain in July 2020 with the article "EfficientDet: Scalable and Efficient Object Detection". [11] 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 5.10 shows the EfficientDet architecture.

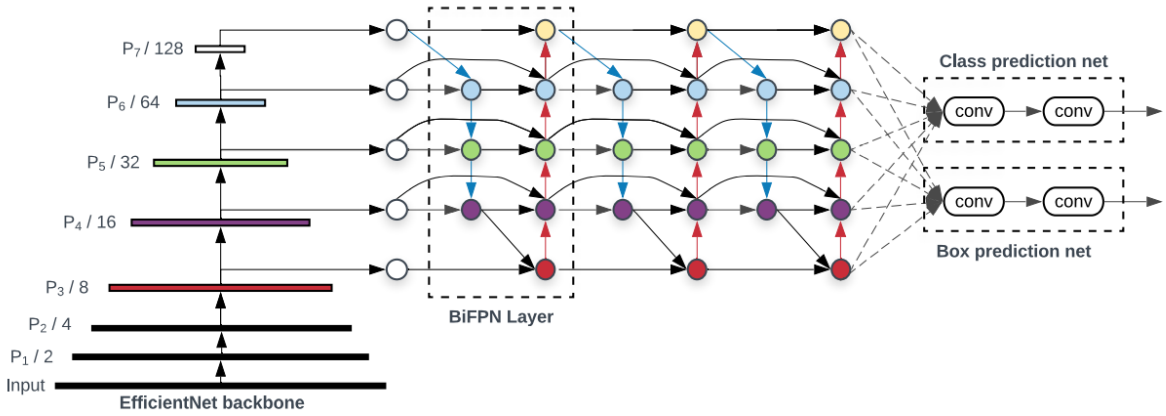


Figure 5.10 EfficientDet Architecture [11]

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" [12].

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. SSD architecture is shown in Figure 5.11.

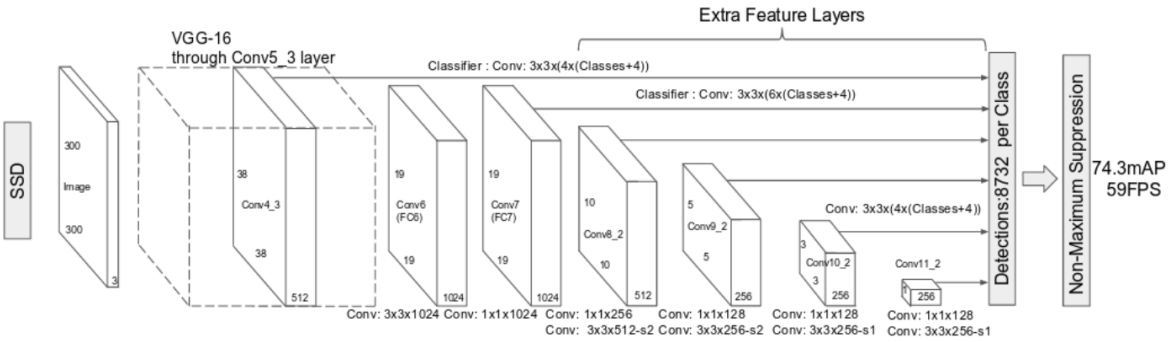


Figure 5.11 SSD Architecture

6
Application

In the application chapter, the followed procedure to make this project is presented.

We have chosen 5 different methods for object detection. All selected methods are trained using Google Colab. These methods are; YOLOv5, Mask R-CNN, Detectron2, 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.

6.1 Dataset Information

The dataset we used in our project was given to us by our consultant. There are 12 videos with different video lengths in this dataset. These videos were cropped into five frames per second and the images were extracted. These images were previously labeled as sperm, not sperm, using an arbitrary labeling tool. Since the image size is large, such as 1920x1200, the image sizes are set to 640 x 640 to use it in each architecture. While doing this sizing, the tagged area is also has been taken care of. Additionally, since each architecture accepts different labelling formats, conversions have been made between these formats. No data enhancement nor other sample multiplier methods were used on the dataset. The number of labels for each video is shown in detail in Table 6.1.

Table 6.1 The number of labels for each video

	SPERM	NOT SPERM	TOTAL	FRAME	VIDEO TIME(Second)
Video 1	**2510**	**361**	**2871**	**60**	**12**
Video 2	**3095**	**267**	**3362**	**51**	**10**
Video 3	**3798**	**151**	**3949**	**54**	**10**
Video 4	**1224**	**45**	**1269**	**13**	**2**
Video 5	**3443**	**163**	**3606**	**47**	**9**
Video 6	**2974**	**1950**	**4924**	**150**	**30**
Video 7	**15544**	**3597**	**19141**	**150**	**30**
Video 8	**13191**	**4201**	**17392**	**150**	**30**
Video 9	**7913**	**4800**	**12713**	**150**	**30**
Video 10	**4516**	**3000**	**7516**	**150**	**30**
Video 11	**5782**	**3000**	**8782**	**150**	**30**
Video 12	**4554**	**2250**	**6804**	**150**	**30**

6.2 YOLOv5 Application

In this section, the best and worst results of the YOLOv5 trained model are shown in Figure 6.1 and Figure 6.2.

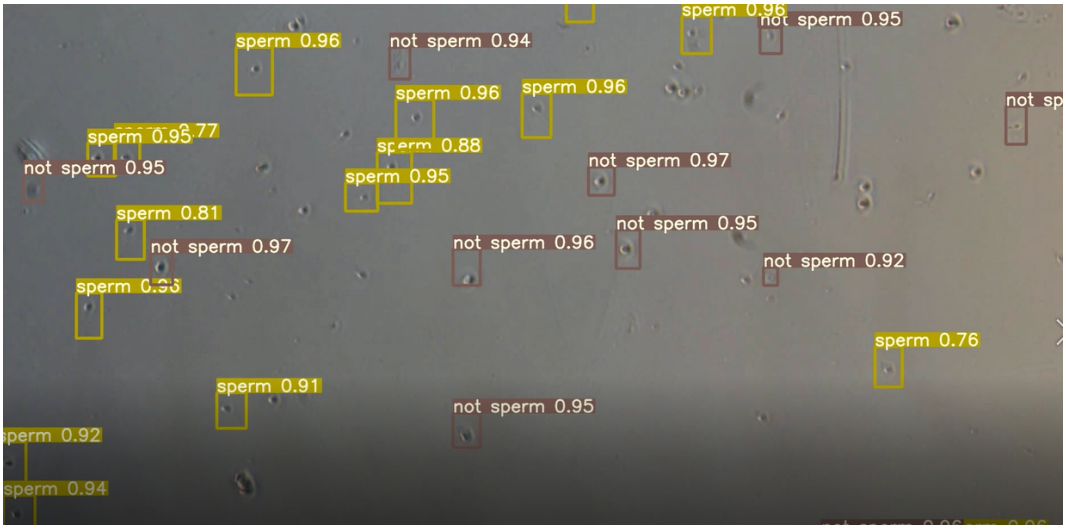


Figure 6.1 The best result in the YOLOv5 model

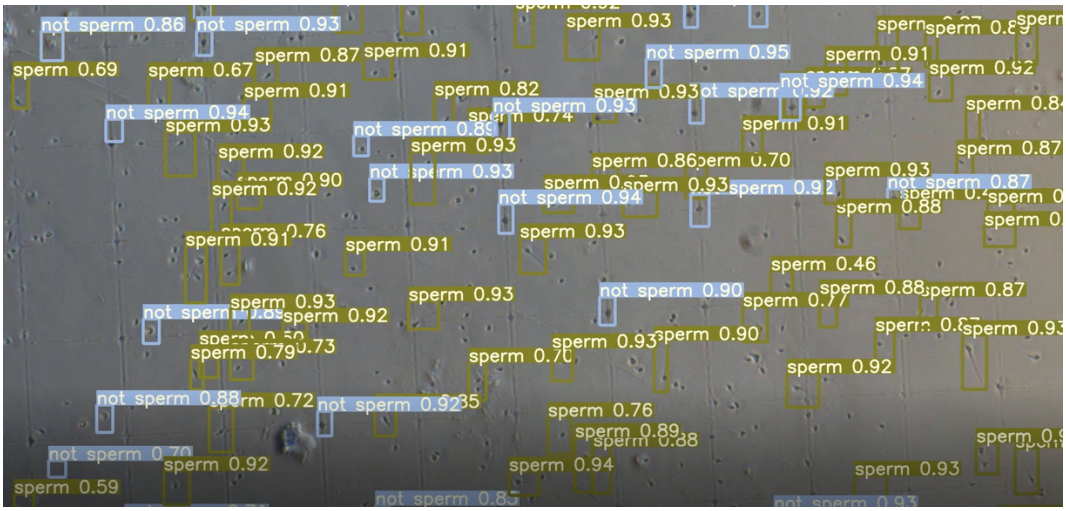


Figure 6.2 The worst result in the YOLOv5 model

6.3 Mask R-CNN Application

In this section, the best and worst results of the Mask R-CNN trained model are shown in Figure 6.3 and Figure 6.4.

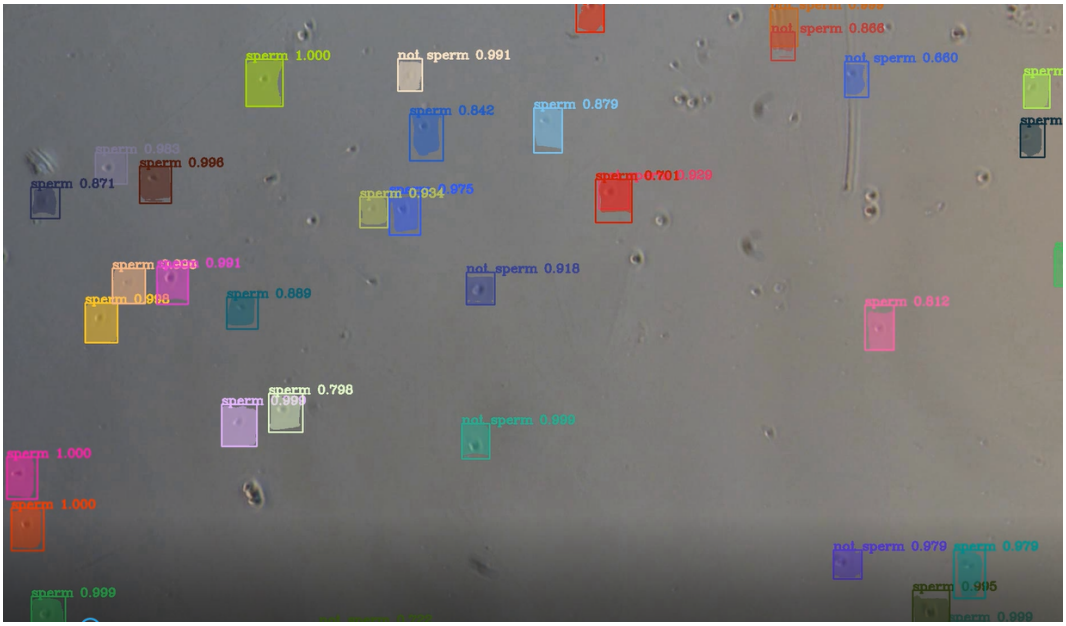


Figure 6.3 The best result in the Mask R-CNN model

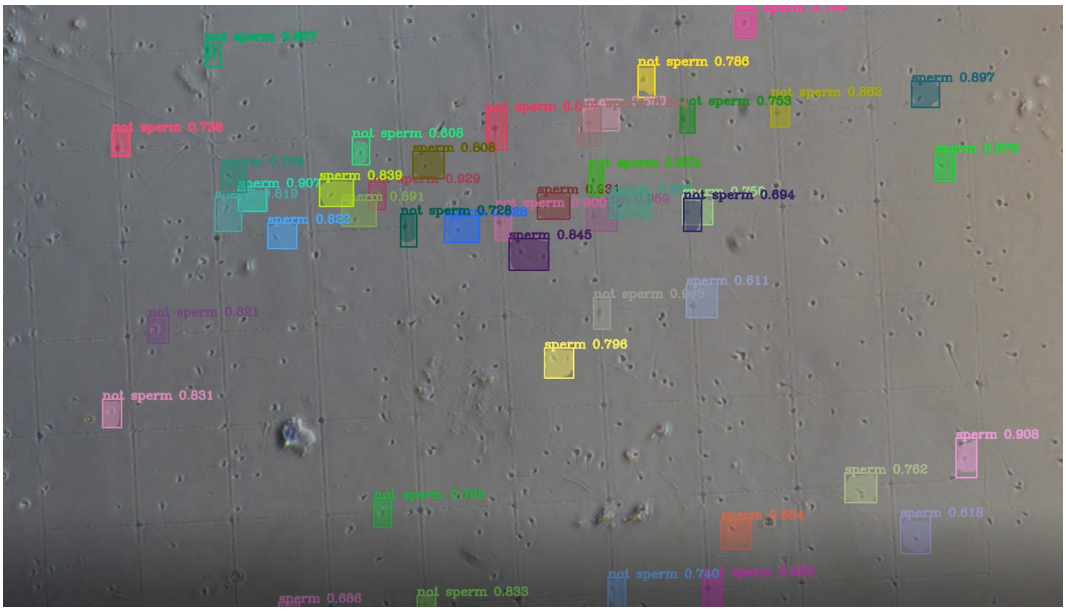


Figure 6.4 The worst result in the Mask R-CNN model

6.4 SSD Application

In this section, the best and worst results of the SSD trained model are shown in Figure 6.5 and Figure 6.6.

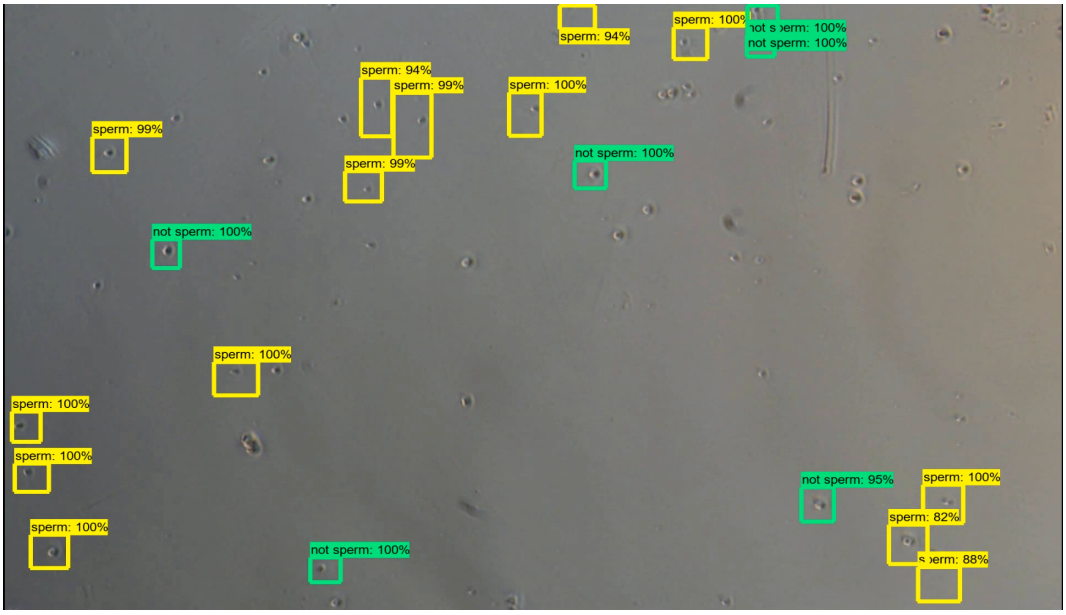


Figure 6.5 The best result in the SSD model

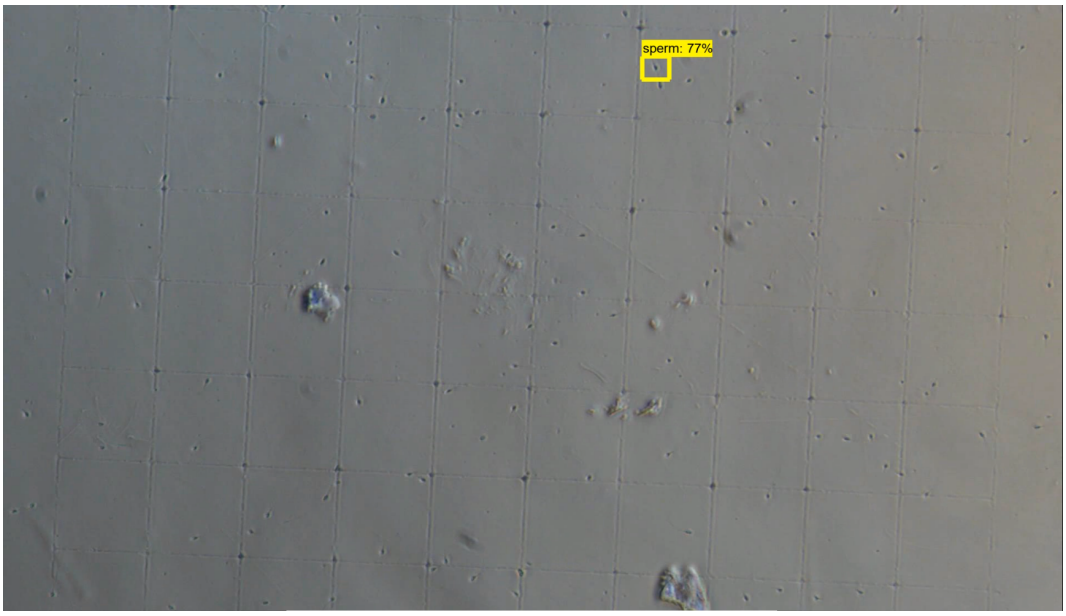


Figure 6.6 The worst result in the SSD model

6.5 DETECTRON2 Application

In this section, the best and worst results of the Detectron2 trained model are shown in Figure 6.7 and Figure 6.8.

Figure 6.7 The best result in the Detectron2 model

Figure 6.8 The worst result in the Detectron2 model

6.6 EfficientDet Application

In this section, the best and worst results of the EfficientDet trained model are shown in Figure 6.9 and Figure 6.10.

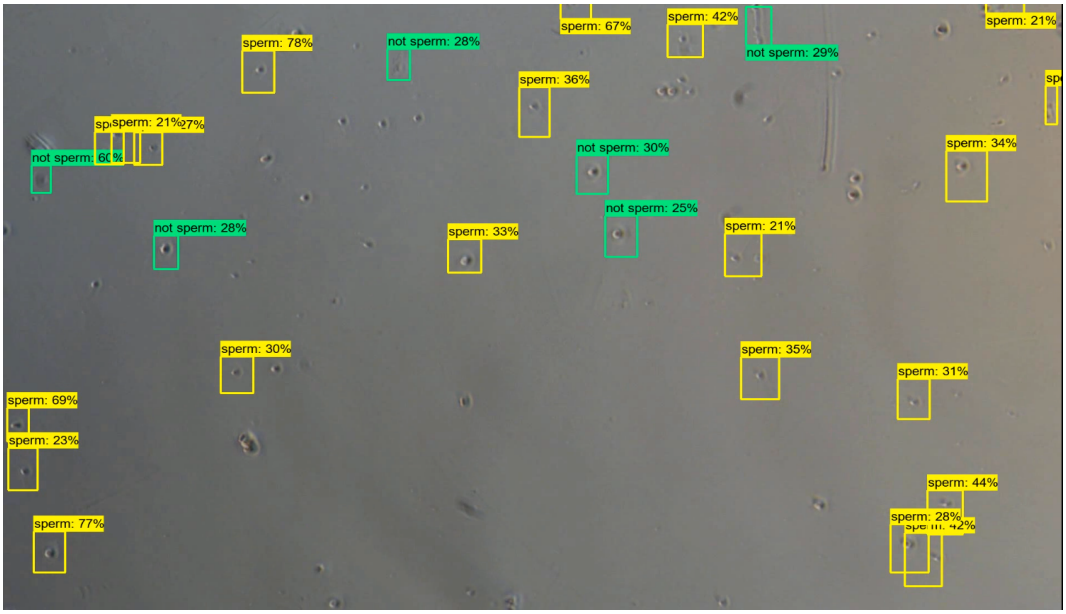


Figure 6.9 The best result in the EfficientDet model

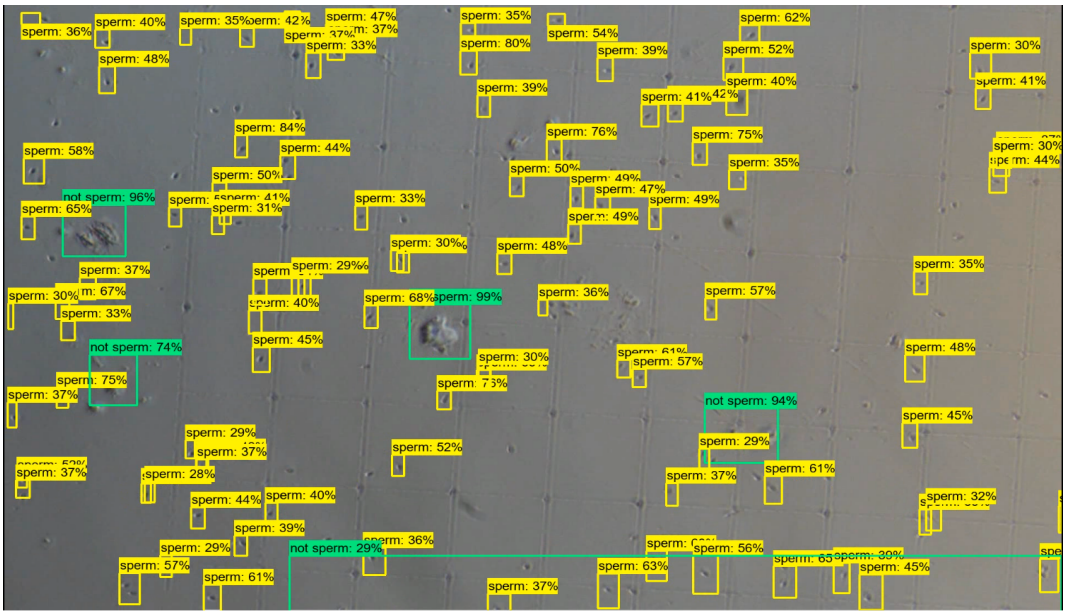


Figure 6.10 The worst result in the EfficientDet model

7
Experimental Results

In this chapter, our criteria for evaluating the models and the results we have obtained are shown.

7.1 Performance Metric

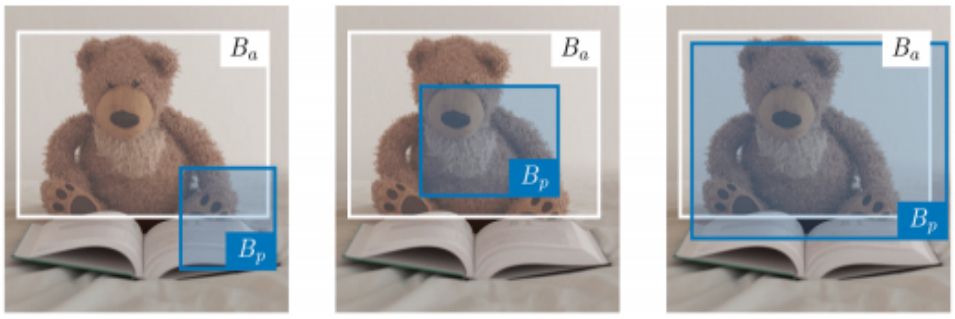
In our project, 2 different criteria have been chosen to measure the performance and evaluate the success between models. These are mAP @ IoU = 0.5 and the training times of each model. Explanations about these parameters are given below.

7.1.1 Intersection Over Union

Intersection over Union, also known as IoU, is a function that quantifies how correctly positioned a predicted bounding box B_p is over the actual bounding box B_a. It is defined as:

$$IOU(B_p, B_a) = (B_p \cap B_a)/(B_p \cup B_a)$$

We always have IOU $\in$ [0,1]. By convention, a predicted bounding box B_p is considered as being reasonably good if IOU(B_p,B_a) $\geq$ 0.5. Figure 7.1 gives an example for IOU.



$\text{IoU}(B_p, B_a) = 0.1$ $\text{IoU}(B_p, B_a) = 0.5$ $\text{IoU}(B_p, B_a) = 0.9$

Figure 7.1 IOU Example

7.1.2 mAP @ IoU = 0.5

In computer vision, mAP is a popular evaluation metric used for object detection. Localization determines the location of an instance and classification tells you what it is.

The mean Average Precision or mAP score is calculated by taking the mean AP over all classes or overall IoU thresholds, depending on different detection challenges that exist.

In PASCAL VOC2007 challenge, AP for one object class is calculated for an IoU threshold of 0.5. So the mAP is averaged over all object classes. For the COCO 2017 challenge, the mAP is averaged over all object categories and 10 IoU thresholds.

Pascal VOC mAP metric is used in this project. Threshold value was accepted as 0.5 and mAP was calculated accordingly.

7.1.3 Training Times

Training time is an important metric in object detection algorithms. While some algorithms are fast in training time, they may give low mAP results. In order to get proper results from some algorithms, it is necessary to train the model for a period of 1 day. Our aim is to collect data from the patient as soon as possible and to train the model and to perform sperm determinations. For these reasons, the training times for each model were recorded and compared. Training times are calculated in hours and written in hours in the tables.

7.2 Achieved Results

In this section, the achieved results with this project are shown.

Let's first look at the results in scenario 1. In this scenario, we have divided the dataset in different rates and completed the training of the models. When we looked at the results, we saw that we got the best mAP values when we split the dataset into %80 training and %20 test. However, in this case, we observed that the training periods increased. In another case, we reserved %20 of the dataset for training and the remaining %80 for testing. In this case, while the training times decreased considerably, the mAP values gave poor results. Finally, when we divided the data into %50 and %50 for training and testing, no significant increase in mAP values was observed, but again the training times increased slightly.

7.2.1 mAP@IoU=0.5 Results

Table 7.1 shows all results for %20 training and %80 testing. Table 7.2 shows all results for %50 training and %50 testing. Table 7.3 shows all results for %80 training and %20 testing.

Table 7.1 %20 Train %80 Test Results

	YOLOv5	MASK R-CNN	DETECTRON2	SSD	EFFICIENTDET
Video 1	**0.576**	**0.438**	**0.398**	**0.300**	**0.390**
Video 2	**0.749**	**0.634**	**0.536**	**0.161**	**0.383**
Video 3	**0.701**	**0.428**	**0.572**	**0.268**	**0.376**
Video 4	**0.770**	**0.459**	**0.509**	**0.0**	**0.402**
Video 5	**0.642**	**0.446**	**0.457**	**0.0016**	**0.387**
Video 6	**0.982**	**0.991**	**0.958**	**0.970**	**0.958**
Video 7	**0.927**	**0.678**	**0.588**	**0.667**	**0.836**
Video 8	**0.984**	**0.765**	**0.791**	**0.906**	**0.853**
Video 9	**0.879**	**0.764**	**0.755**	**0,638**	**0.640**
Video 10	**0.948**	**0.900**	**0.897**	**0.846**	**0.896**
Video 11	**0.887**	**0.848**	**0.862**	**0.754**	**0.861**
Video 12	**0.968**	**0.965**	**0.967**	**0.918**	**0.951**

Table 7.2 %50 Train %50 Test Results

	YOLOv5	MASK R-CNN	DETECTRON2	SSD	EFFICIENTDET
Video 1	**0.683**	**0.472**	**0.512**	**0.286**	**0.396**
Video 2	**0.697**	**0.658**	**0.630**	**0.235**	**0.465**
Video 3	**0.643**	**0.461**	**0.607**	**0.279**	**0.354**
Video 4	**0.921**	**0.358**	**0.645**	**0.0**	**0,261**
Video 5	**0.668**	**0.465**	**0.597**	**0.0**	**0.415**
Video 6	**0.994**	**0.990**	**0.971**	**0.981**	**0.912**
Video 7	**0.980**	**0.710**	**0.593**	**0.775**	**0.826**
Video 8	**0.990**	**0.719**	**0.805**	**0.942**	**0.976**
Video 9	**0.948**	**0.817**	**0.771**	**0.647**	**0.688**
Video 10	**0.973**	**0.915**	**0.916**	**0.883**	**0.911**
Video 11	**0.883**	**0.864**	**0.872**	**0.760**	**0.856**
Video 12	**0.980**	**0.976**	**0.969**	**0.933**	**0.918**

Table 7.3 %80 Train %20 Test Results

	YOLOv5	MASK R-CNN	DETECTRON2	SSD	EFFICIENTDET
Video 1	**0.717**	**0.529**	**0.544**	**0.291**	**0.359**
Video 2	**0.812**	**0.687**	**0.473**	**0.114**	**0,358**
Video 3	**0.650**	**0.481**	**0.625**	**0.243**	**0.365**
Video 4	**0.849**	**0.415**	**0.670**	**0.0**	**0,149**
Video 5	**0.694**	**0.530**	**0.553**	**0.001**	**0.352**
Video 6	**0.995**	**0.999**	**0.988**	**0.977**	**0.993**
Video 7	**0.983**	**0.719**	**0.601**	**0.871**	**0.639**
Video 8	**0.995**	**0.745**	**0.834**	**0.932**	**0.975**
Video 9	**0.971**	**0.813**	**0.791**	**0.638**	**0.657**
Video 10	**0.979**	**0.938**	**0.928**	**0.901**	**0.788**
Video 11	**0.942**	**0.878**	**0.878**	**0.803**	**0.863**
Video 12	**0.984**	**0.977**	**0.973**	**0.942**	**0.958**

7.2.2 Training Time Results

Table 7.4 shows the training times in all models.

Table 7.4 Training Time Results

	YOLOV5	MASK R-CNN	DETECTRON2	SSD	EFFICIENTDET
Scenario 1A	**02:07**	**03:46**	**02:25**	**04:13**	**06:32**
Scenario 1B	**02:56**	**03:52**	**02:00**	**04:38**	**06:43**
Scenario 1C	**03:13**	**03:57**	**02:26**	**04:41**	**06:50**

Let's show the results for the second scenario. In YOLOv5, Mask R-CNN and Detectron2, video 3 and video 5 were chosen as the videos with the worst mAP value. In SSD and EfficientDet models, video 4 was selected as the video with the worst mAP value, and the method we described in scenario 2 was applied to them.

Table 7.5 gives Video 3, Table 7.6 Video 5 results for scenario 2. Table 7.7 shows the results obtained in scenario 2 for Video 4.

Table 7.5 Video3 Results

		YOLOv5		**MASK R-CNN**		**DETECTRON2**	
		mAP@0.5	**Training Time**	**mAP@0.5**	**Training Time**	**mAP@0.5**	**Training Time**
Scenario 1	1A	**0.701**	**1:43**	**0.428**	**3:40**	**0.572**	**02:29**
	1B	**0.643**	**1:41**	**0.461**	**4:25**	**0.607**	**02:25**
	1C	**0.650**	**1:44**	**0.481**	**4:28**	**0.625**	**02:23**
Scenario 2	2A	**0.712**	**2:40**	**0.178**	**3:52**	**0.205**	**02:21**
	2B	**0.729**	**2:43**	**0.278**	**4:05**	**0.427**	**02:25**
	2C	**0.846**	**2:51**	**0.300**	**4:18**	**0.518**	**02:16**

Table 7.6 Video 5 Results

		YOLOv5		MASK R-CNN		DETECTRON2	
		mAP@0.5	**Training Time**	**mAP@0.5**	**Training Time**	**mAP@0.5**	**Training Time**
Scenario 1	1A	**0.642**	**1:43**	**0.446**	**4:00**	**0.457**	**2:21**
	1B	**0.668**	**2:05**	**0.465**	**4:12**	**0.597**	**2:22**
	1C	**0.694**	**1:49**	**0.530**	**4:35**	**0.553**	**2:21**
Scenario 2	2A	**0.753**	**2:18**	**0.202**	**4:16**	**0.164**	**2:19**
	2B	**0.762**	**2:20**	**0.275**	**4:25**	**0.412**	**2:22**
	2C	**0.835**	**2:25**	**0.326**	**4:18**	**0.418**	**2:21**

Table 7.7 Video 4 Results

		SSD		EFFICIENTDET	
		mAP@0.5	**Training Time**	**mAP@0.5**	**Training Time**
Scenario 1	1A	**0.099**	**3:42**	**0.402**	**4:25**
	1B	**0.111**	**3:45**	**0,261**	**4:30**
	1C	**0.114**	**4:05**	**0.149**	**4:15**
Scenario 2	2A	**0.022**	**3:58**	**0.323**	**8:15**
	2B	**0.167**	**3:51**	**0.352**	**8:20**
	2C	**0.787**	**4:12**	**0.381**	**8:23**

8
Performance Analysis

In this section, evaluations are made about the success of the models and their training periods are presented.

Based on the achieved test results, YOLOv5 is the very best detector by far. Its results are exceptional. The training time of YOLOv5 is brief and the its mAP values are great. The second place goes to Detectron2. It has a faster training time than YOLOv5 but its mAP average is lower. Mask RCNN, which is a deriving version of Faster RCNN, scored as well as Detectron2 but its training time average is almost double the size of Detectron2. Contrary to the good results, SSD, scored the lowest. SSD's bad score can give us an intuition that it results bad detecting small objects. This is a well known problem for this object detection method.

Table 8.1 shows the average training results and the detectors are sorted based on their performance.

Table 8.1 Average training results and the detectors sorted based on performance

	YOLO	DETECTRON2	MASK-RCNN	EFFICIENTDET	SSD
Video 1	**0.658**	**0.484**	**0.479**	**0.3816**	**0.2925**
Video 2	**0.752**	**0.546**	**0.659**	**0.402**	**0.1703**
Video 3	**0.664**	**0.601**	**0.456**	**0.365**	**0.2634**
Video 4	**0.846**	**0.608**	**0.41**	**0.27**	**0.111**
Video 5	**0.668**	**0.535**	**0.48**	**0.3846**	**0.127**
Video 6	**0.99**	**0.972**	**0.993**	**0.954**	**0.9762**
Video 7	**0.963**	**0.594**	**0.702**	**0.767**	**0.771**
Video 8	**0.989**	**0.81**	**0.743**	**0.934**	**0.9269**
Video 9	**0.932**	**0.772**	**0.798**	**0.661**	**0.641**
Video 10	**0.966**	**0.913**	**0.917**	**0.865**	**0.8764**
Video 11	**0.904**	**0.87**	**0.863**	**0.86**	**0.7726**
Video 12	**0.977**	**0.969**	**0.972**	**0.942**	**0.931**
mAP Avg.	**0.859**	**0.723**	**0.706**	**0.649**	**0.572**
Avg. Time	**2:45**	**2:17**	**3:51**	**6:41**	**4:30**

Additionally, the second scenario resulted worse except for YOLOv5.

9
Conclusion

In this project, sperm cells, which are difficult to detect with the human eye, have been studied. The advantages of performing this determination by computers in terms of accuracy and speed have been investigated. Some algorithms have been very successful in detecting small objects like sperm and have real-life usability. Some methods, on the other hand, gave good results in large objects, but were unsuccessful in small objects such as sperm. The project was completed with a dataset consisting of 12 videos in total, using 5 different object detection algorithms and performing 2 different scenarios.

In order to evaluate the results fairly, the trainings were carried out in 4000 iterations in every model and the image sizes were fixed as 640x640. The dataset plays a huge role for object detection. Good results can be obtained with a more properly labeled dataset. With the completion of all training, it has been observed that the best model to detect small objects is YOLOv5. With both fast training and high mAP value, the YOLOv5 model can be applied to patients in real life.

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] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 580–587.

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

[11] 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.

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

Curriculum Vitae

FIRST MEMBER

Name-Surname: Özgür KAN
Birthdate and Place of Birth: 04.08.1994, İstanbul
E-mail: ozgur.kan@std.yildiz.edu.tr
Phone: 0534 396 64 45
Practical Training: I haven't done an internship yet

SECOND MEMBER

Name-Surname: Usame İSLAMOĞLU
Birthdate and Place of Birth: 11.11.1998, İstanbul
E-mail: l1116905@std.yildiz.edu.tr
Phone: 0553 095 32 11
Practical Training: POLONOM TEKNOLOJİ SAN. VE TİC. A.Ş.
DELIVERS.AI ROBOTİK OTONOM B.T.A.Ş.

Project System Informations

System and Software: Windows İşletim Sistemi, Python
Required RAM: 8GB
Required Disk: 8GB