Using Deep Learning to Streamline Intracytoplasmic Sperm Injection in Cancer Patients

Manav Aggarwal
Stanford University
manav@stanford.edu

Varin Satish Nair
Stanford University
varin@stanford.edu

Tiger Sun
Stanford University
tgsun@stanford.edu

Abstract

In the midst of the sperm infertility problem across the United States, identifying and classifying viable sperm, especially among those with a lower sperm count, is an emerging research field. There has been relatively little research in this field that has taken advantage of deep learning. Here, we hope to use deep learning image detection and segmentation techniques to not only identify sperm from an image, but also determine whether it is viable for in-vitro fertilization (IVF) in an effort to streamline the IVF process. The dataset of around 1100 images was graciously provided by Stanford embryologists. After performing augmentation on the data, we had around 4400 images to work with.

We explored state-of-the-art algorithms for object detection and classification, such as the YOLOv3 and Faster-RCNN architectures. We achieved decent results, but noted some modification in the dataset that could make it easier to train in the future.

1. Introduction

Currently, there is a sperm infertility problem sweeping across the nation, where sperm are becoming less viable for IVF and fertilization in general. We are working with Stanford Medicine to tackle this problem using deep learning and computer vision. Our data consists of a few thousand images of testicular tissue from patients suffering from cancer and other clinical situations that may involve a low sperm count. Our task is to develop a model capable of first extracting sperm from an image, then labeling that specific sperm as "normal" or "abnormal." This is normally a very time-consuming task for a human expert to manually inspect these images and locate the sperm and label it as normal or abnormal. We think that automating this process can save time and at the same time, provide accurate results. Afterwards, viable sperm could be used for intracytoplasmic sperm injection (ICSI) methods, and could end up helping couples conceive through in-vitro fertilization methods.

This idea is fairly novel, and there has not been much research in the area. Recently, in April 2019, researchers at Stanford Hospital were able to classify rare spermatozoa from other types of cells, such as "red blood cells, white blood cells, Sertoli cells, spermatogenic precursors (i.e. spermatgonia, spermatocytes, spermatids) and debris" with decently high accuracy using deep convolutional neural networks [7]. We would like to build on this research by introducing detection and more specific classification into categories of "normal" and "abnormal." We are looking into object detection and segmentation frameworks for a two-tiered approach to solve this problem.

The given input for our model will be an image containing either normal sperm or abnormal sperm (or both) along with its bounding boxes and labels, stored in text format. We will run it through some sort of convolutional network created for object detection and classification, in our case the state-of-the-art YOLOv3 and Faster-RCNN models. The model will give us bounding boxes coordinates and any labels of sperm that it identifies.

2. Related Work

The current methods for sperm detection largely rest on computer-assisted sperm analysis (CASA), which greatly relies on the motion of the sperm to detect the morphology. Though the systems are promising, they are relatively expensive and still need to go through a validation and standardization process [1][8]. Additionally, current research shows that current CASA processes are less accurate than expert prediction, and thus still requires work in the future [10].

It has recently been suggested that deep learning be used to automate the process. Earlier, we mentioned that researchers at Stanford were able to classify sperm, amongst a host of different other types of cells, achieving an accuracy rate of about 70%, which was less than the expert rate of 93% [7]. This suggests that deep learning has potential but still needs to be refined. Additionally, the same lab at

Stanford was able to use convolutional neural networks to distinguish between types of zygotes with accuracy similar to that of an expert[14]. Though the difference between normal and abnormal sperm may be more subtle, this shows that convolutional neural networks can certainly be on-par with experts when classifying some types of data.

Current non-AI methods for detection and segmentation largely rely on edge-finding, curve fitting, and advanced mathematical equations and have been shown to do a good job of being able to identify sperm [3]. Additionally, considering detection, researchers have also been able to detect sperm within a picture with high accuracy of over 90% using 2-conv and 3-conv network [6]. However, it is unclear how the method proposed bounding box areas, and the paper lacked results images showing identified sperm. These papers also do not go into how to classify the sperm as normal or abnormal.

As for our classification step, we were able to find a paper on the morphological classification of sperm heads [15]. Using a multi-layer perceptron, the researchers were able to separate sperm into groups based on the sperm-head shape. This is promising; however it required images of just sperm heads, which we did not use.

As current state-of-the-art methods have not yet been applied to sperm images, we researched methods for detection and classification that have been used in other medical imaging projects. The YOLO (You-Only-Look-Once) model was applied to lung cancer samples with great precision and sensitivity, both around 90% [13], which could be a promising place to start. Similarly, the Faster-RCNN network has been used in medical imaging to distinguish different types of blood cells in malaria at a nearly expert level [5]. These two methods are proven to work in a medical setting, and would be valuable to consider.

3. Methods

3.1. YOLOv3

For this task, we looked into general methods that could label classes from an image. We looked into traditional architectures and algorithms used to train ImageNet and CoCo models.

One of the deep learning algorithms we used for our project was YOLOv3.

Classifiers like VGGNet or Inception can be converted into object detectors by using sliding a window across the image. At each step, you run the classifier to get a prediction about the object that is inside the current window. This gives hundreds or thousands of predictions for the image, but we keep only the ones that the classifier is most certain about.

While this approach works, it is very slow, since you need to run the classifier many times. YOLO takes a completely different approach. YOLO only looks at the image once (hence its name: You Only Look Once) but in a clever way [4].

It divides up the image into a grid of 13 by 13 cells like Figure 1.

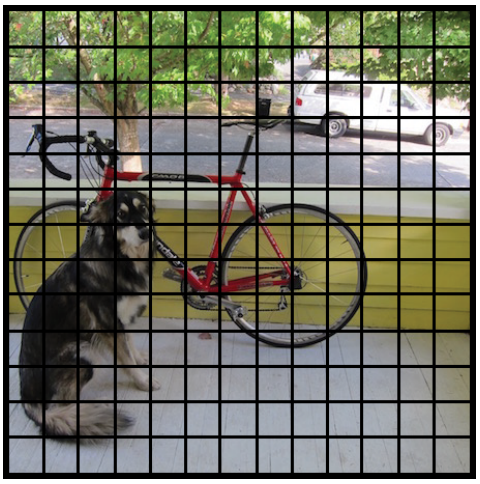


Figure 1. YOLO divides image into 13 x 13 cells

Each of these cells is responsible for predicting 5 bounding boxes. It also outputs a confidence score that tells us how certain it is that the predicted bounding box actually encloses an object. The confidence score is not indicative of what kind of object is enclosed by the bounding box, just if the bounding box actually contains an object.

The predicted bounding boxes may look something like figure 2 (the higher the confidence score, the fatter the box is drawn):



Figure 2. Each cell predicts 5 bounding boxes

Each cell also predicts a class for each bounding box. This works exactly like a classifier: it gives a probability distribution over all the possible classes. The confidence

scores of the bounding box and class prediction are combined into a final score which tells us the probability that the bounding box contains a specific type of object, like figure 3:

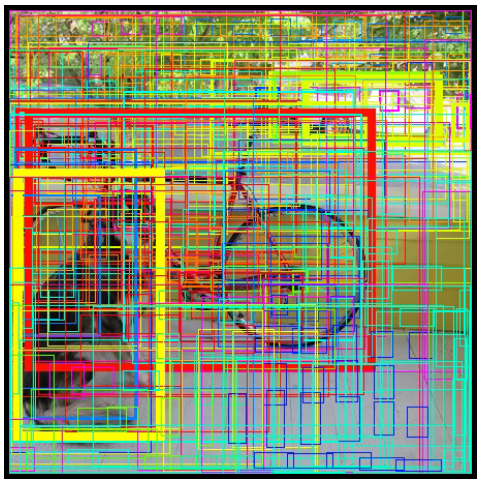


Figure 3. The fat yellow bounding box is 85% sure it contains the object "dog".

There are 13 x 13 = 169 grid cells, each cell predicting 5 bounding boxes. Hence, we end up with 169 x 5 = 845 bounding boxes in total. It turns out that most of these boxes will have very low so we only keep the boxes whose final score is 30% or more (the 30% threshold is the default and can be changed as well, depending on how accurate we want the detector to be).

The final prediction is then something like figure 4:

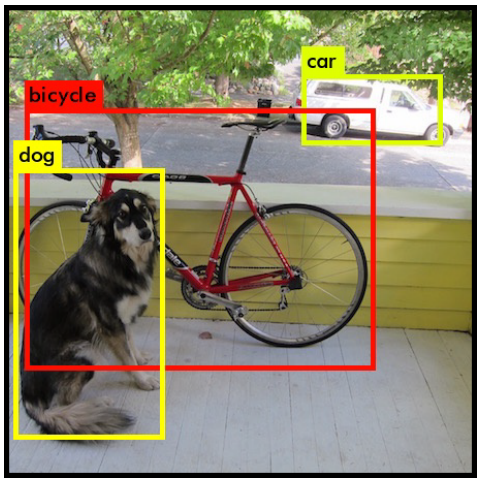


Figure 4. Each cell predicts 5 bounding boxes

From the 845 total bounding boxes we only kept the three in 4 because they gave the best results. But note that even though there were 845 separate predictions, they were all made at the same time the neural network just ran once.

While using YOLO in our project, we used the Mean Squared Error (MSE) loss to optimize for x-y and w-h, cross entropy (CE) loss to optimize for class confidence, and binary cross entropy (BCE) loss to optimize for the bounding box offset prediction. Below, we write the formulas for each of these loss functions:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$$

$$\text{CE} = \sum_{c=1}^{N} y_{o,c}\log(p_{o,c})$$

$$\text{BCE} = -\frac{1}{N}\sum_{i=1}^{N}(y_i \log(p(y_i)) + (1-y_i)\log(1-p(y_i))$$

While the x-y and w-h deal with the dimensions of the bounding box, the bounding box offset refers to exact location of the bounding box in the image.

3.2. Faster-RCNN

One of the other frameworks we tried was Faster-RCNN. Faster-RCNN is one of the most well known object detection neural networks. It is the modified version of Fast RCNN. The major difference between them is that Fast RCNN uses selective search for generating Regions of Interest, while Faster RCNN uses Region Proposal Network, aka RPN. RPN takes image feature maps as an input and generates a set of object proposals, each with an objectness score as output. This method was covered in the Object Detection lecture of class.

4. Dataset and Features

4.1. Raw Dataset

The raw dataset that we were given consisted of 1110 images which consisted of one or more sperm. Along with each image, we were also given rectangular bounding box coordinates or polygon coordinates that give the location of the sperm in the image. In our dataset, only 25% of the sperm are normal sperm - the rest of the 75% are abnormal sperm. This could lead to a class imbalance in our dataset.

Along with images, we were also given a CSV file which contained annotations for the bounding boxes of sperm, along with a classification of whether each of the sperm was normal or abnormal. One of the rows in our CSV was formatted such as the following:

{image1.jpeg, 91, 213, 425, 157, A}

Here, image1.jpeg refers to the image filename, 91 and 213 refer to top left x and top left y of bounding box respectively, 425 and 157 refer to the width and height of bounding box respectively, and 'A' refers to the label. Here, there are

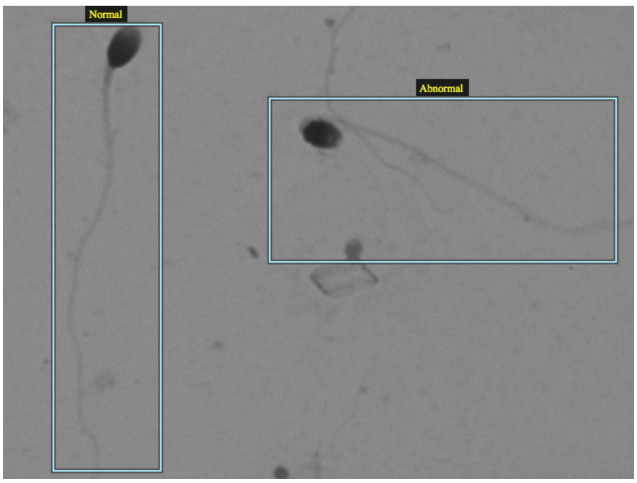


Figure 5. Sample labeled image

only two possible labels that can be given - 'A' which means an abnormal sperm, and 'N' which means normal sperm.

To visualize our dataset better to see the similarity between abnormal and normal sperms, we ran a t-SNE dimensionality reduction algorithm.

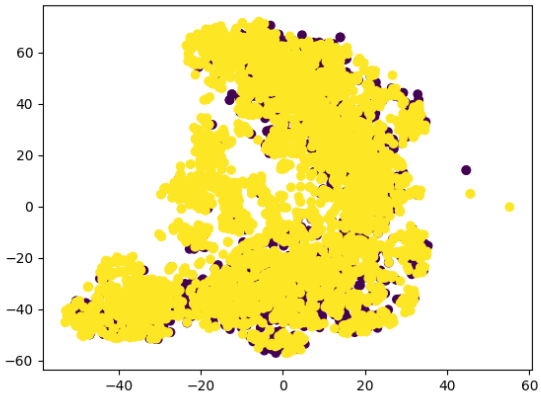


Figure 6. t-SNE visualization of Dataset

The purple markers in the image represent normal sperm and the yellow markers represent abnormal sperm. Since images in both the classes have a lot of overlap and look similar, it is hard to distinguish between abnormal and normal sperm.

4.2. Preprocessing and Augmentation Techniques

We decided that we only wanted rectangular bounding boxes, and therefore needed to convert the polygon coordinates of the sperm locations to rectangular coordinates. To get the smallest possible rectangular bounding box from the polygon coordinates, we used OpenCV (specifically cv2.boundingRect() function).

Initially, we decided to use white-padding to convert all the images to a size of 800 x 600. This involved padding the width by 24 pixels, and padding the height by 18 pixels. The padding of 24 pixels can be distributed between the left and right side of the image in many different combinations - e.g. we could have 1 pixel pad on the left and 23 pixel pad on the right, or 5 pixel pad on the left and 19 pixel pad on the right. We elected to use strides of 4 pixels to increase the data set size. Thus, considering the width for example, we could have 0 pixels on the left and 24 on the right, then 4 on the left and 20 on the right, and so on. For every single combination of pixel padding between the width and height, the bounding box coordinates would also be updated. However, we realized that even with strides of 4, the images are extremely similar to each other. Also, white padding in the images is not something that is seen in normal images. For these two reasons, white padding would not appropriate as a data augmentation technique.

We then decided to do flips to augment our data instead. So, we performed a horizontal flip, vertical flip, and a combination of horizontal and vertical flips. Hence, for each image, we ended up getting 3 more augmented images, and ended up increasing our data set by a factor of 4.

For example, figure 7 refers to one of the images in our original dataset before the data augmentation. After performing, we got images like figure 8.

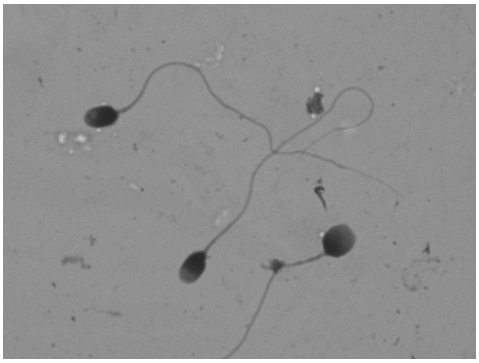


Figure 7. Original image in our data

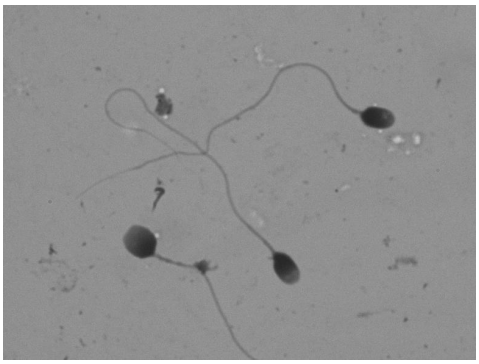


Figure 8. Horizontally flipped image of the original image.

4.3. Final Dataset

After data augmentation, our final data set contained 4,440 images. Just recently, our collaborator sent us more raw images that would have helped correct the class imbalance of normal-abnormal, but we have not yet had enough to act on that data.

5. Experiments/Results/Discussion

5.1. Primary Metrics

The primary metric we decided to look at was the mean average precision (mAP) [11]. However, to have a better grasp of mAP, we must first consider the classic metrics of precision and recall in an object detection setting. Let TN denote true negative, TP denote true positive, FN denote false negative, and FP denote. Then, we have

$$\text{Recall} = \frac{TP}{TP + FN}, \text{ Precision} = \frac{TP}{TP + FP}$$

To calculate the TN, FN, TP, FP, we must consider intersection over union (IOU) which is intuitively the intersection of two bounding box areas divided by the union. A prediction is classified as TP if the $IOU > 0.5$ and the classification is correct. If the classification is wrong, we call it a FN by definition. Finally, if $IOU < 0.5$ or we have extra bounding boxes, this is our FP since the bounding box we predicted is not either accurate enough either way. The mAP is the average precision (AP) calculated for all classes, which in our case is just 2. The AP can be calculated in a variety of ways. We will explore one example here where

$$AP = \frac{1}{11} \sum_{r \in \{0, 0.1, \cdots, 1\}} p_{interp}(r)$$

where p_{interp} refers to the maximum precision for each value of r, which is the recall. Other AP metrics also use this idea of creating thresholds in the relationship between recall and precision, and averaging over them.

5.2. YOLOv3 Experiment

5.2.1 YOLOv3 Setup

We used a PyTorch-based YOLOv3 model given by Ultralytics on Github.[12] The majority of the code we wrote was boilerplate for transferring the original data into the COCO format required by YOLOv3. In our given CSV, each row contained a sperm, its bounding box (in (x, y, w, h) format where x, y are the coordinates of the top left corner and w, h are the width and height), and the file it was contained it.

However, COCO format is a different format where instead, each file has an associated text file containing data for every sperm in that file. The format for each line in the text file is as follows. Suppose W and H are the width and height of the entire image. Let L be the label for that sperm (where 0 denotes 'normal' and 1 denotes 'abnormal'). For a particular sperm, the associated text for that sperm would

$$\begin{bmatrix} L & \frac{1}{W}(x + \frac{w}{2}) & \frac{1}{H}(y + \frac{w}{2}) & \frac{w}{W} & \frac{h}{H} \end{bmatrix}$$

More intuitively, this format is (label, regularized horizontal center, regularized vertical center, regularized width, regularized height).

As our previous baseline, we had run YOLOv3 with the default parameters with a stochastic gradient optimizer on the padded data. However, we realized that this was causing overfitting problems as the extraneous data made it difficult for the algorithm to generalize. Thus, the new models are all trained the newly augmented data consisting of flips and rotations. We trained the model with Darknet as the backbone for 120 epochs and a batch size of 16. On the NVIDIA Tesla V100, each epoch took an average of 90 seconds.

5.2.2 YOLOv3 Hyperparameter Evolution

As part of our experiment, to help tune our hyperparameters we ran an evolution based system. At each generation, we vary our initial hyperparameters by

```
variation = (random(0,1) * hyp + 1)^1.1
```

where hyp is the parameter we want to vary. Our fitness parameter is the mAP calculated after $n = 24$ epochs. As we evolve, we keep the hyperparameter set that outperforms the best one, which we store. If the current hyperparameter set does not have a better mAP after 24 epochs, we keep the previous best set. This way, we can hope to generate better hyperparameters as generations go on. We chose the three best hyperparameter sets after 24 epochs to run for our normal 120 epoch series. Additionally, instead of stochastic gradient descent, we also ran an experiment with the Adam optimizer with (beta1, beta2) as (0.9, 0.999), initial learning rate of 0.000198, and weight decay of 0.0007838. The results of the hyperparameter tuning are shown in Figure 9 of the next subsection.

5.2.3 YOLOv3 Results

As we can see in Figure 9, the mAP grows steadily with the number of epochs, which is exactly what we want. However, for all the models run with stochastic gradient descent, we saw that test loss also increases. This signals as some kind of overfitting in the SGD models, but since mAP is also steadily increasing, this should be fine. Our best hyperparameter set is given in Figure 10. We see the training loss continuing to decrease and level off as it approaches 120 epochs on all the parameter sets, which is what we want to see.

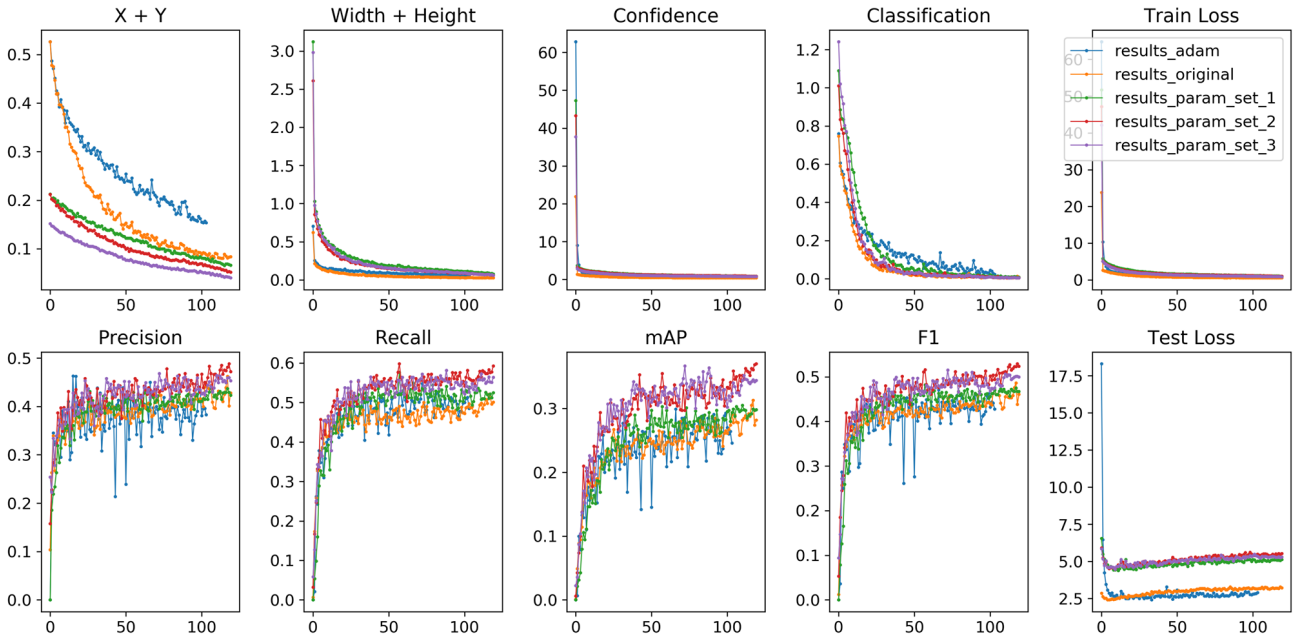


Figure 9. Comparative loss and mAP graphs for various hyperparameter sets and optimizers

Hyperparameter Metric	Value
X-Y Loss Gain	0.05135
W-H Loss Gain	0.3035
Classification Loss Gain	0.03947
Confidence Loss Gain	8.901
IOU Target-anchor Training Threshold	0.09312
Initial Learning Rate	0.0002466
Final Learning Rate Exponent	-4.44
Momentum	0.8651
Weight Decay	0.0001844

Figure 10. Parameter Set 2

We also give the best mAP of each set of parameters in Figure 11. Though we see that the best mAP are very similar for sets 2 and 3, from our graph in Figure 9, we note that set 2 is much more consistent in its high level performance. Through parameter evolution, we achieved a 18% increase in the mAP.

Parameters	Best mAP
Original	0.313
Adam Optimizer	0.293
Parameter Set 1	0.305
Parameter Set 2	0.37
Parameter Set 3	0.367

Figure 11. Parameter set performances

Our final test statistics for parameter set 2 are given below as well in Figure 12.

Precision	Recall	mAP	F1
0.472	0.593	0.37	0.524

Figure 12. Statistics on the Test set

Finally, we include some images where the YOLOv3 network correctly detected the sperm and compare them to some images where the network failed to correctly identify sperm. We will analyze these together with pictures from our next model in a subsequent subsection.

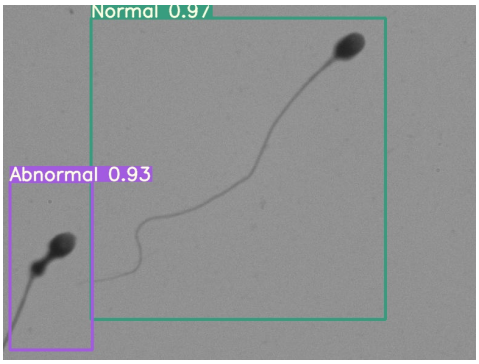


Figure 13. YOLOv3: Correctly detected sperm

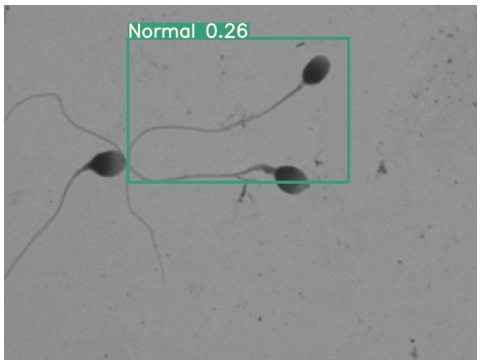


Figure 14. YOLOv3: Incorrectly Detected Sperms

5.3. Faster-RCNN Experiment

5.3.1 Faster-RCNN Results

We followed a Faster-RCNN tutorial [9] which guided us through converting our csv data into a (filename,xmin, ymin, xmax, ymax, label) format and used an existing implementation [2] of Faster-RCNN in Keras. We trained the model with ResNet50 as a backbone on 50 epochs with an epoch length of 1000. On a NVIDIA Tesla V100, each epoch took an average of 720 seconds. The figures below seem to show that the data has been trained well, since test loss is very close to the training loss. Additionally, as epochs increases, we see that accuracy does as well, which is exactly what we would like. Moreover, the mean number of overlapping bounding boxes from the RPN with the ground truth increases with epochs which is promising since the overlap between predicted and ground truth value increases leading to an increase in Total IoU. We have included graphs of individual losses and the mean number of overlapping bounding boxes from RPN with the ground truth, all of which that go in the appendix for reference.

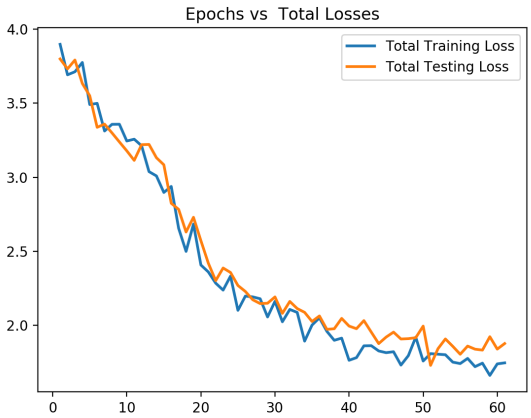


Figure 15. Epochs vs Total Losses

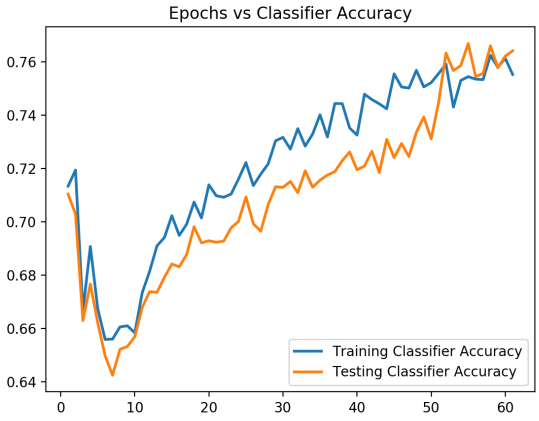


Figure 16. Epochs vs Classifier Accuracy

Below, we give more examples of correctly and incorrectly identified data from the network. In a following section, we will qualitatively discuss both the Faster-RCNN and YOLOv3 classifications.

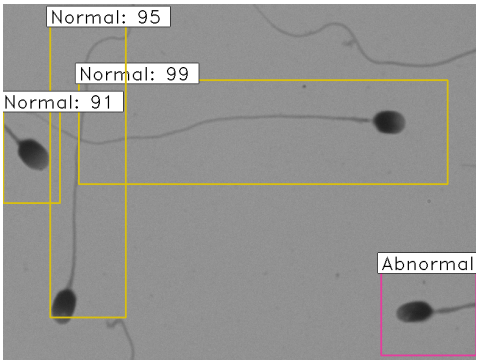


Figure 17. Faster-CNN: Correctly detected sperm

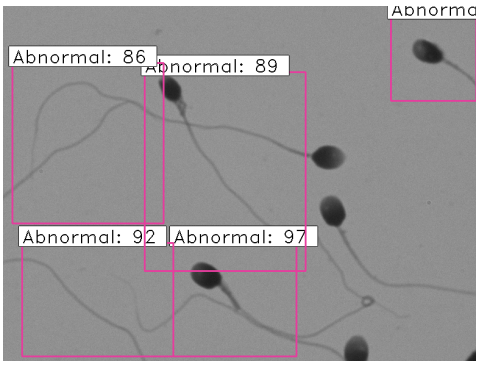


Figure 18. Faster-CNN: Incorrectly detected sperm

In addition, we created a confusion matrix of the outputs of the Faster-CNN.

	Predict Normal	Predict Abnormal
Actual Normal	179	207
Actual Abnormal	198	716

Figure 19. Confusion Matrix with IOU = 0.5

Metric	Value
Accuracy	0.6885
Recall	0.4637
Precision	0.4748
F1-Measure	0.4692
Mean IoU	0.7409
mAP	0.5682

Figure 20. Test Metrics for Faster-RCNN

An interesting thing to note here is the accuracy here might have been skewed by the class imbalance. Within the confusion matrix, we note that the classifier is much better at predicting abnormal data versus normal data, which might be because there is much more abnormal data than normal data. Also, we looked at the predicted results to find that the model predicts bounding boxes for most of the test data. Combining this result with our high Mean IoU (0.7409 with a 0.5 threshold) and mAP values, but lower accuracy, recall and precision, we conclude that our model either does really well at detection, but doesn't always classify the sperm correctly between normal and abnormal.

5.4. Qualitative Analysis

Here, we aim to investigate some of the reasons that YOLOv3 and Faster-RCNN correctly predicated Figures 13 and 17 but incorrectly predicted Figures 14 and 18. A closer look at the dataset can provide some key insights. All in all, we have 1110 images with around 1624 total sperm within those images, giving us an average of $\frac{1624}{1110} = 1.463$ sperm per image. However, when we look at Figure 18, we see that there are many more sperm than the average. Thus, from the data, it is entirely conceivable that the model has not been trained enough on data with multiple sperm intermingled with each other. Indeed, when we look at Figure 14, the model seems to be confused by the intersections of sperm. This is contrasted by Figure 13, where each sperm has its own distinct "box" that it is contained within. We see that the model works well with many images with just a single sperm. Also, we discovered that our input data sometimes contains just the head or the tail of a sperm or even just an empty image which is not sufficient to classify normal/abnormal sperm. This contributes to a bad classification performance and detecting empty regions as sperms [Figure 18] sometimes even when the normal and abnormal data is already really similar [Figure 6].

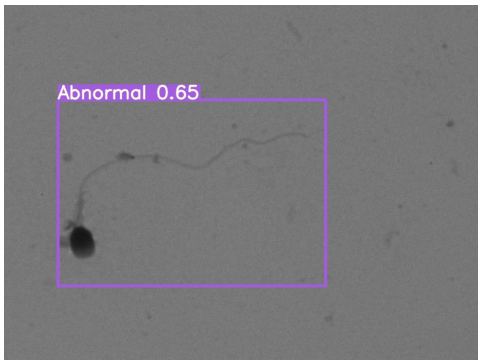


Figure 21. YOLOv3: Correctly identified singleton

6. Conclusion/Future Work

We conclude from our experiments that Faster-RCNN performs better than YOLO since Faster-RCNN has a mAP of 0.568 whereas YOLO has a mAP of 0.37, which is expected as Faster-RCNN works better as a two stage detector. However, this mAP is still relatively low, and could still be improved. As evidenced by our incorrect guesses, there are still some fundamental problems with the model.

With regards to future work, we believe more data could help solve this problem. Having more data with overlapped sperm could help boost the performance of the model, where it could learn to distinguish between each sperm better. Additionally, having more normal sperm data would help too with regards to the classification problem. That being said, there are such subtle differences between some abnormal and normal sperm that it may ultimately be difficult to distinguish between the two categories of sperm.

Ultimately, we conclude that deep learning has potential in the field of sperm detection and classification. We are excited to be one of the first to apply state-of-the-art deep learning models such as YOLOv3 and Faster-RCNN to this problem, and the results are promising for the future.

As for future work, we plan to rerun our models with the new, more balanced data set. This will hopefully give us better final results. Additionally, we were thinking of designing a custom architecture. We noticed that normal and abnormal sperm are just so similar that it could be possible to run a detection model to simply identify sperm in an image. Afterwards, we would run a model to classify that sperm. We figure that having two separate models that can get very good at two very different tasks could be an interesting way to tackle this problem.

7. Appendix

Below, we have attached many graphs given from the Faster-RCNN training.

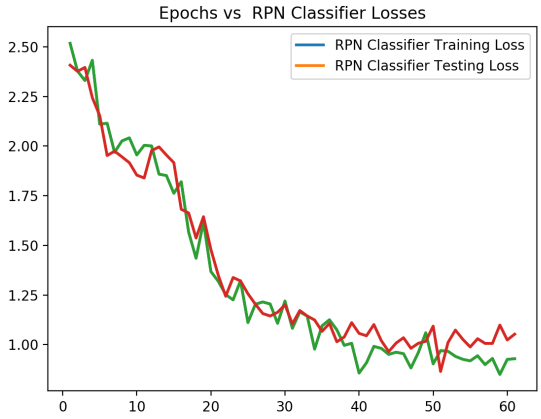


Figure 22. Epochs vs RPN Classifier Loss

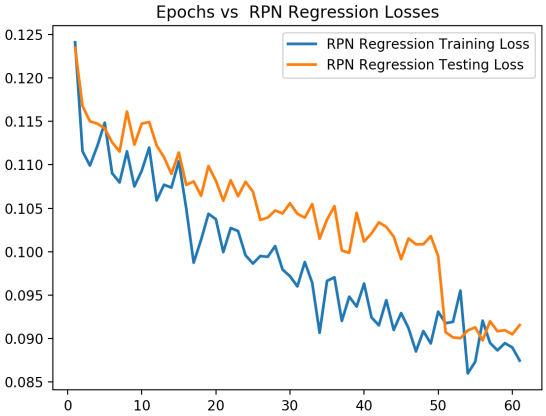


Figure 23. Epochs vs RPN Regression Loss

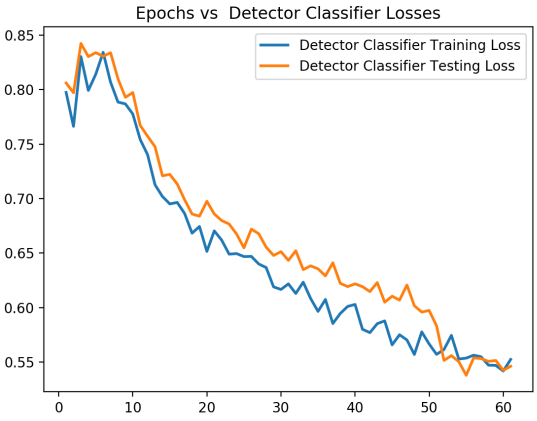


Figure 24. Epochs vs Detector Classifier Loss

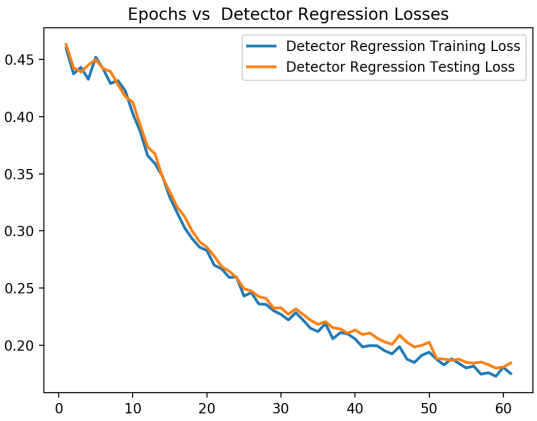


Figure 25. Epochs vs Detector Regression Loss

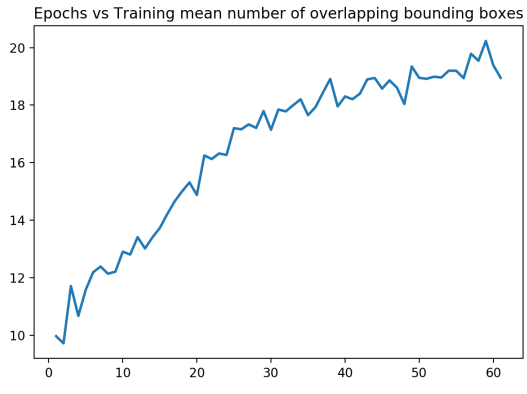


Figure 26. Epochs vs Training mean number of overlapping bounding boxes

8. Acknowledgements

Here we would like to list our individual member contributions.

Tiger wrote scripts to convert the augmented data format into the COCO format used by the YOLOv3 network, and other scripts to generate the necessary data files for the YOLOv3 network. He trained the network on his virtual machine, hosted by Google Compute Engine optimized for deep learning, and tweaked the hyperparameters using the evolution technique described above.

Manav wrote scripts to convert the augmented data into a standardized format, divided the data into training, validation, and a testing set. He trained the Faster-RCNN network on his virtual machine, hosted by Google Compute Engine optimized for deep learning, and modified the original scripts to add model evaluation metrics.

Varin wrote Python scripts that performed data augmentation on the original dataset. The augmentation techniques included white padding in the beginning, and subsequently flips. He also tried testing out a custom VGG architecture for sperm classification on Google Compute Engine.

We would like to thank Odgerel Badamjav and Vik Reddy from Stanford Medicine for helping collect and label the data for us. We would also like to thank Noor Siddiqiu for her mentorship and help throughout the process. Finally, we would like to link the Github resources that we used for the YOLOv3 network [12] and Faster-RCNN infrastructure [2].

References

[1] W. D. Amann RP. Computer-assisted sperm analysis (casa): capabilities and potential developments. URL: https://www.ncbi.nlm.nih.gov/pubmed/24274405.

[2] K. Bardool. Keras implementation of faster r-cnn, 2017. URL: https://github.com/kbardool/keras-frcnn.

[3] A. R. N.-N. Fariba Shaker, S. Amirhassan Monadjemi. Automatic detection and segmentation of sperm head, acrosome and nucleus in microscopic images of human semen smears. 2016. URL: https://dl.acm.org/citation.cfm?id=2955970.

[4] M. Hollemans. https://machinethink.net/blog/object-detection-with-yolo/. 2017. URL: https://towardsdatascience.com/.

[5] J. Hung. Applying faster r-cnn for object detection on malaria images. URL: https://arxiv.org/pdf/1804.09548.pdf.

[6] K. A. S. K. T. T. N. Malte S. Nissen, Oswin Krause and M. Nielsen. Convolutional neural networks for segmentation and object detection of human semen. 2018. URL: https://arxiv.org/abs/1704.00498v1.

[7] D. M. M. E. B. B. Odgerel Badamjav, Vikrant Reddy. Preliminary application of convolutional neural network to improve the efficieny of finding rare spermatozoa in microtese samples, 2019. URL: https://www.fertstert.org/article/S0015-0282(19)30191-8/pdf.

[8] M. D. N. W. Rijsselaere T, Van Soom A. Computer-assisted sperm analysis in dogs and cats: an update after 20 years. URL: https://www.ncbi.nlm.nih.gov/pubmed/23279500.

[9] P. Sharma. A practical implementation of the faster r-cnn algorithm for object detection. 2018. URL: https://www.analyticsvidhya.com/blog/2018/11/implementation-faster-r-cnn-python-object-detection/

[10] T.-H. G. H. J. P. L. J. P. Talarczyk-Desole J, Berger A. Manual vs. computer-assisted sperm analysis: can casa replace manual assessment of human semen in clinical practice? 2017. URL: https://www.ncbi.nlm.nih.gov/pubmed/28326513.

[11] R. J. Tan. Breaking down mean average precision (map). 2019. URL: https://towardsdatascience.com/breaking-down-mean-average-precision-map-ae462f623a5

[12] Ultralytics. Yolov3 in pytorch, 2015. URL: https://github.com/ultralytics/yolov3.

[13] S. R. S. J. G. S. S. V. V. V. Using yolo based deep learning network for real time detection and localization of lung nodules from low dose ct scans. 2018. URL: https://www.spiedigitallibrary.org/conference-proceedings-of-spie/10575/0000/Using-YOLO-based-deep-learning-network-for-real10.1117/12.2293699.full?SSO=1.

[14] D. M. B. B. Vikrant Reddy, Odgerel Badamjav. Pilot utilization of convolutional neural networks to improve the efficiency of fertilization checks. URL: https://www.fertstert.org/article/S0015-0282(19)30156-6/pdf.

[15] J. S. P. Won Jin Yi, Kwangsuk Park. Morphological classification of sperm heads using artificial neural networks. 2016.