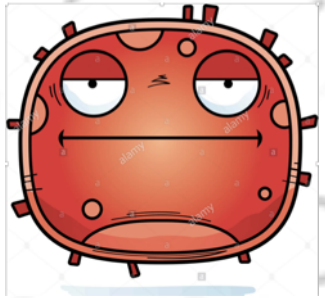
Blood Cells Detection

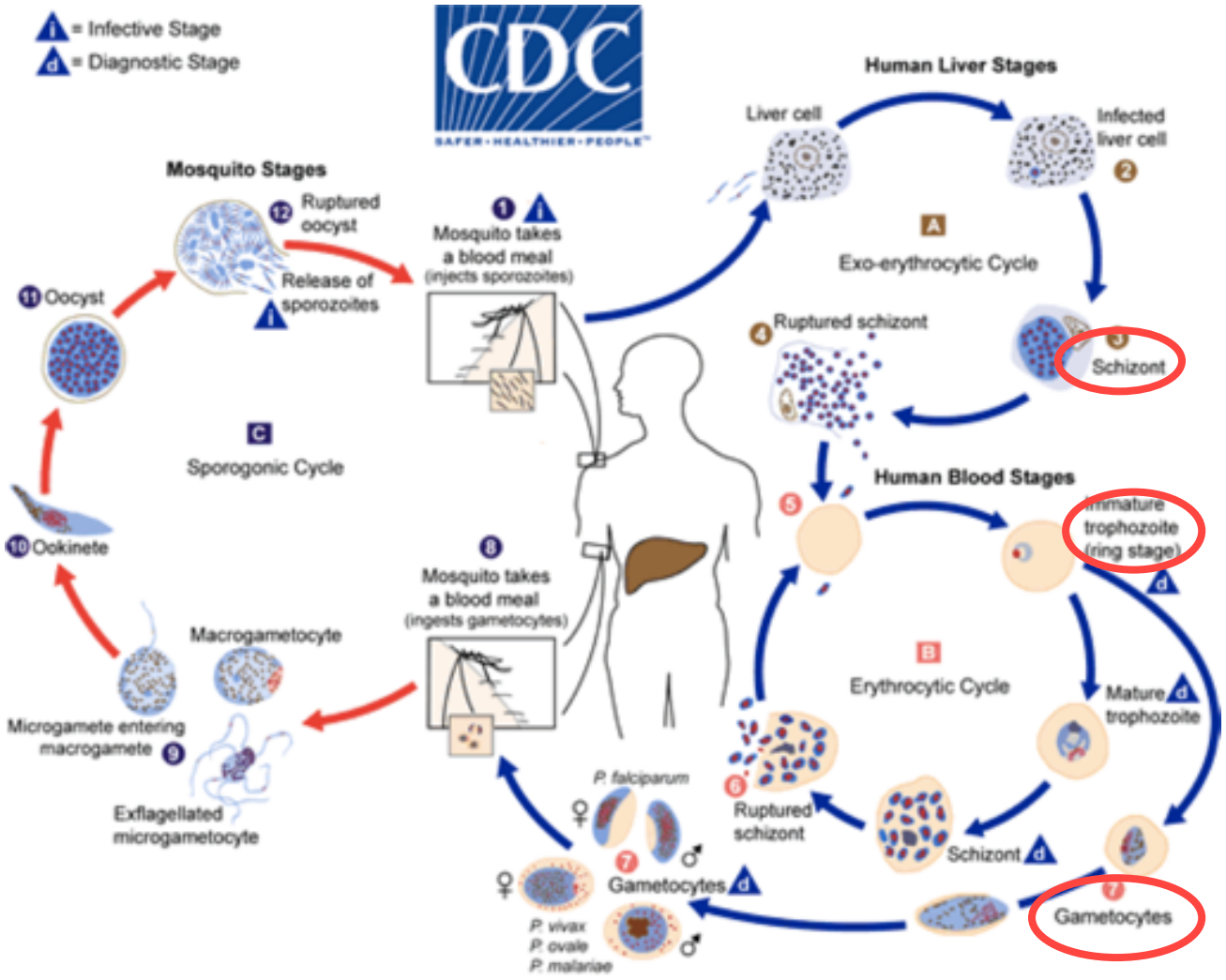
Group 3 : Chao Zhou, Danlei Qian, Ya Liu



CONTENTS

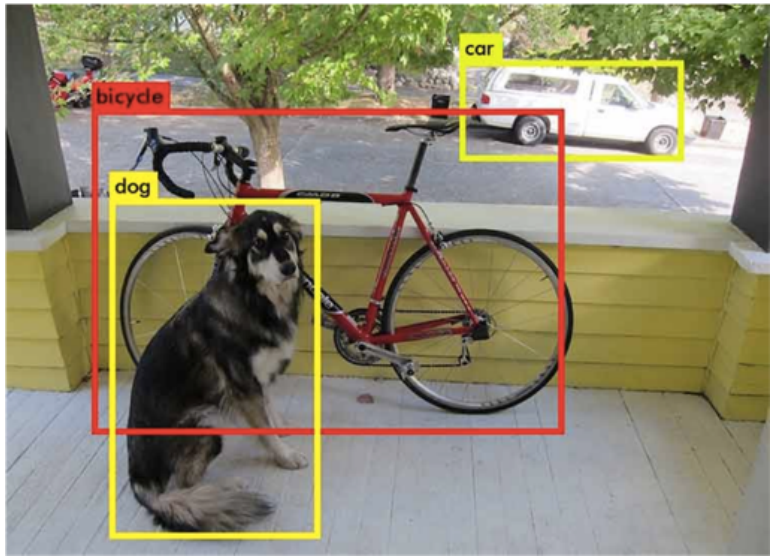
/01 Introduction

Malaria



- Different kinds of cells indicate some stage of malaria infection.
- If a model can successfully detect these types of cells from images taken with a microscope, this would allow to automate a very time-consuming testing process

Object Detection



Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks

Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun

Abstract—State-of-the-art object detection networks depend on region proposal algorithms to hypothesize object locations. Advances like SPPnet [1] and Fast R-CNN [2] have reduced the running time of these detection networks, exposing region proposal computation as a bottleneck. In this work, we introduce a *Region Proposal Network* (RPN) that shares full-image convolutional features with the detection network, thus enabling nearly cost-free region proposals. An RPN is a fully convolutional network that simultaneously predicts object bounds and objectness scores at each position. The RPN is trained end-to-end to generate high-quality region proposals, which are used by Fast R-CNN for detection. We further merge RPN and Fast R-CNN into a single network by sharing their convolutional features—using the recently popular terminology of neural networks with "attention" mechanisms, the RPN component tells the unified network where to look. For the very deep VGG-16 model [3], our detection system has a frame rate of 5fps (*including all steps*) on a GPU, while achieving state-of-the-art object detection

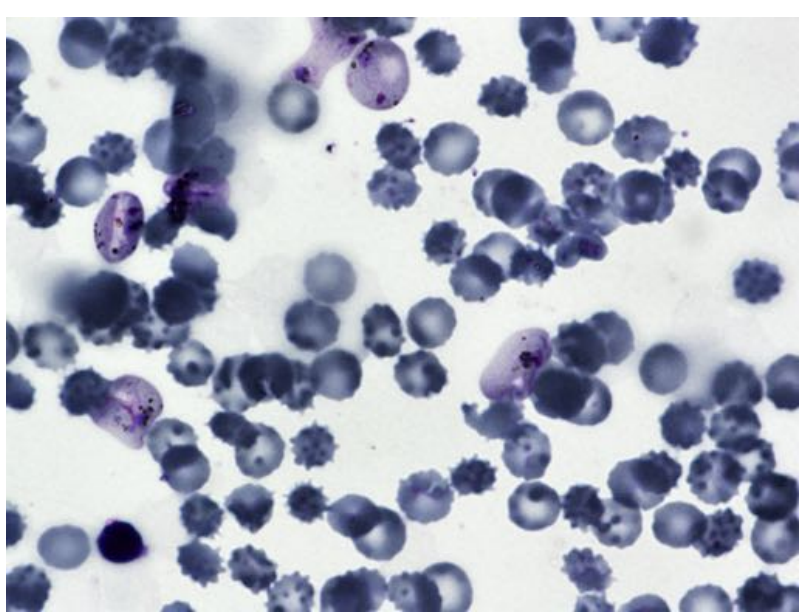
- **Object detection** is the problem of finding and classifying a variable number of objects on an image.
- **Algorithms Classification**: two-stage algorithm(R-CNN -> Fast R-CNN -> Faster R-CNN -> R-FCN), one-stage algorithm(YOLOv1 -> YOLOv2 -> YOLOv3 -> SSD).
- **Challenges**: Variable number of objects, Size of objects

- **Faster R-CNN** was originally published in 2015.
- A classical object detection algorithm, cost-efficient because it brings up with **RPN**.

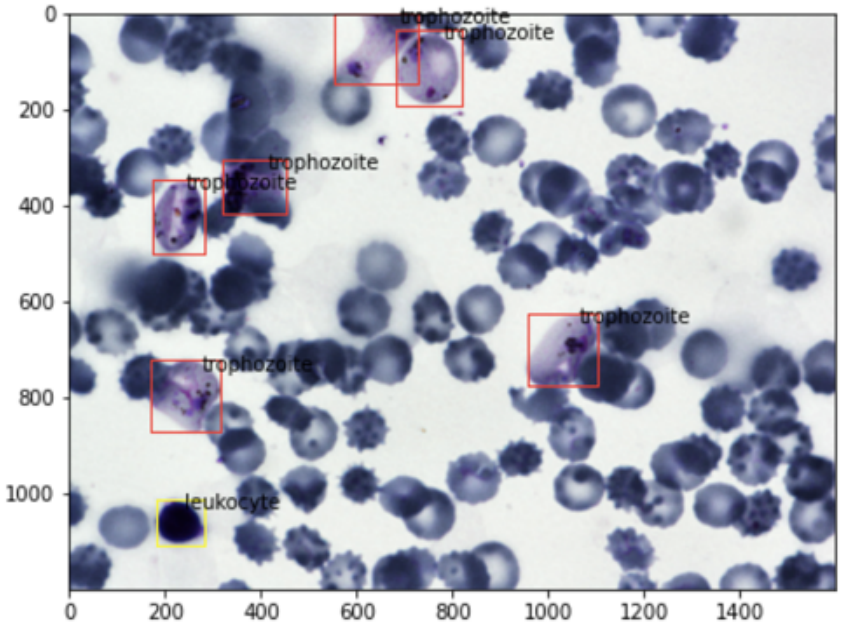
/02

Problem Statement

Problem Statement



Faster R-CNN

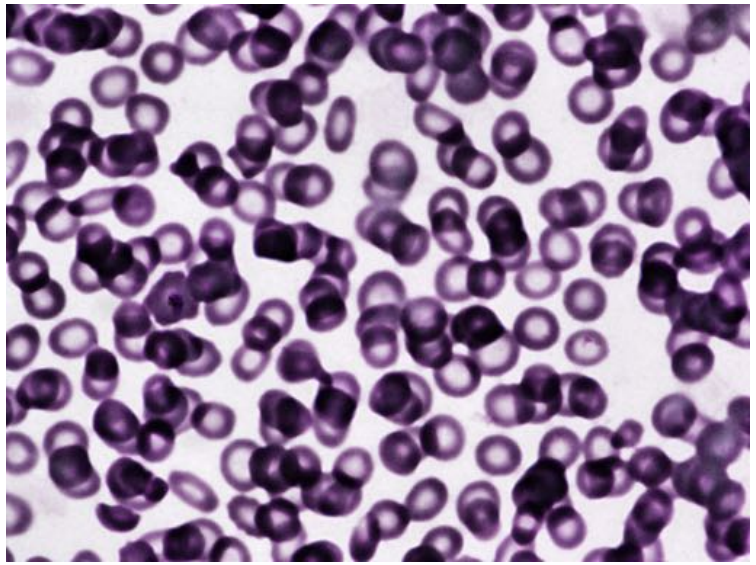


- **Inputs**: images, bounding box, labels
- **Outputs**: bounding box, probability

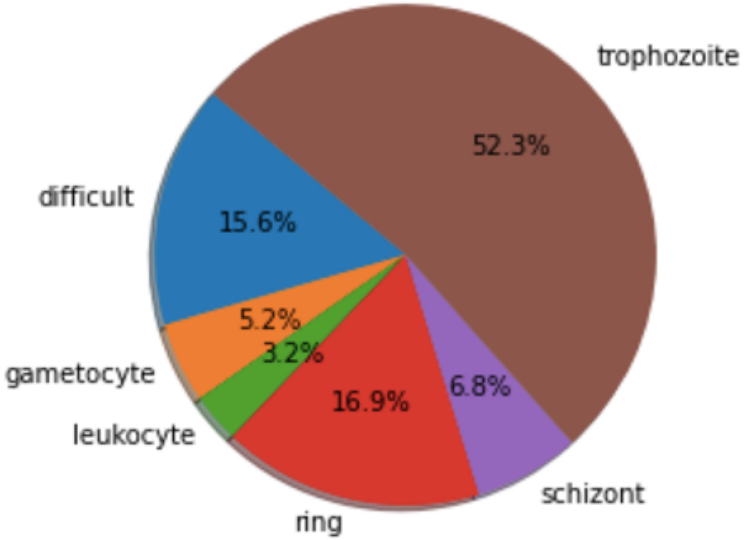
/03

Data Collection

Data Description



image



{'bounding_box': {'minimum': {'r': 574,
'c': 300}, 'maximum': {'r': 692, 'c': 418}},
'category': 'trophozoite'}

Json file

- 802 images
- 6 kinds of cells
- 2100 cells

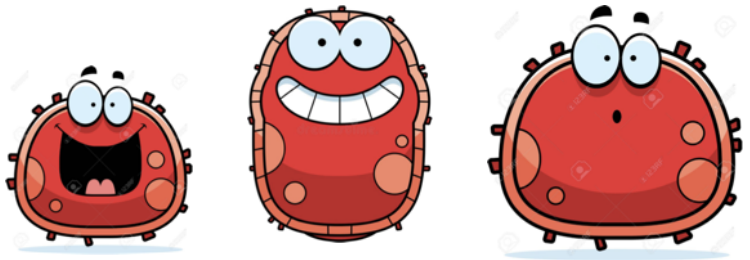
/04 Data Preprocessing

Data Preprocessing

- **Required format of inputs**:
  filepath,x1,y1,x2,y2,class_name
  filepath is the path of the image
  x1 is the xmin coordinate for bounding box
  y1 is the ymin coordinate for bounding box
  x2 is the xmax coordinate for bounding box
  y2 is the ymax coordinate for bounding box
  class_name is the name of the class in that bounding box
- E.g., ['/content/train/cells_1.png', 300, 574, 418, 692, 'trophozoite']

```
/cells_1017.png,711,1105,844,1225,trophozoite
/cells_1017.png,761,972,891,1101,trophozoite
/cells_1017.png,907,53,1004,179,trophozoite
/cells_1017.png,195,176,315,292,trophozoite
/cells_1003.png,384,988,547,1153,difficult
/cells_218.png,463,621,603,750,trophozoite
/cells_218.png,320,1236,475,1374,trophozoite
/cells_230.png,172,941,311,1101,difficult
/cells_224.png,882,873,965,958,trophozoite
/cells_224.png,218,282,321,373,trophozoite
/cells_224.png,229,761,326,857,trophozoite
/cells_595.png,293,501,380,635,trophozoite
```

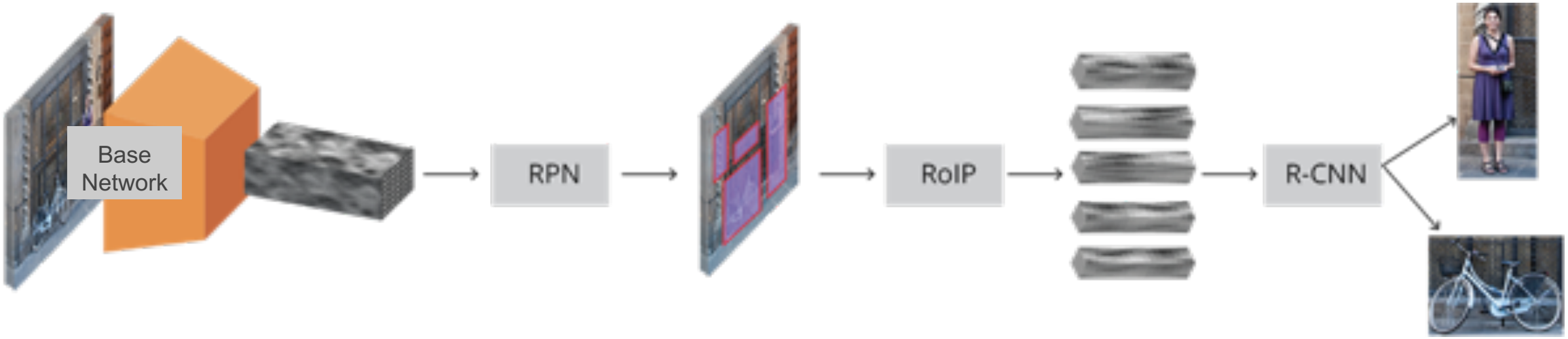
Annotate.txt



/05

Faster R-CNN Theory

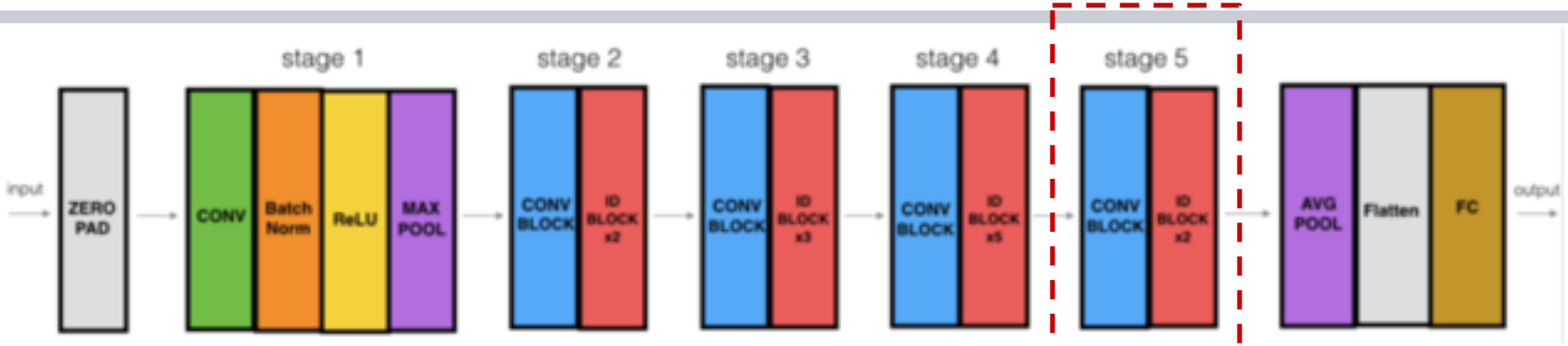
Faster R-CNN Architecture



We want...

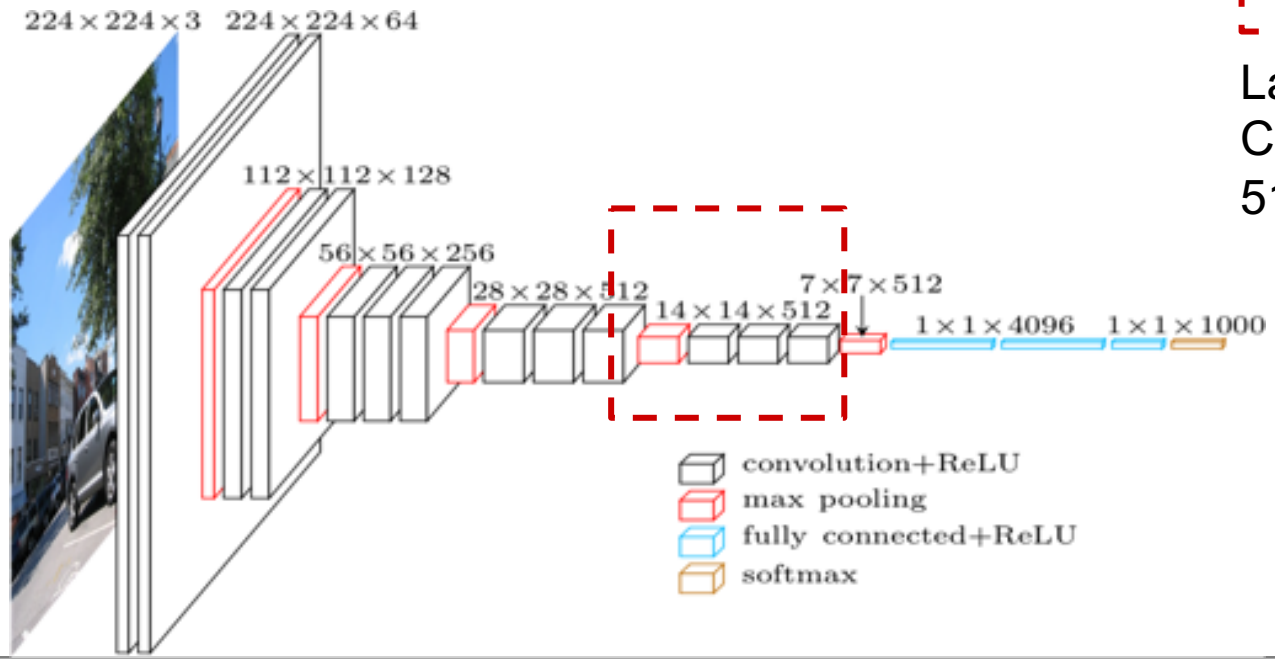
- **a list of bounding boxes.**
- **a label assigned to each bounding box.**
- **a probability for each label and bounding box.**

Base Network



Resnet50

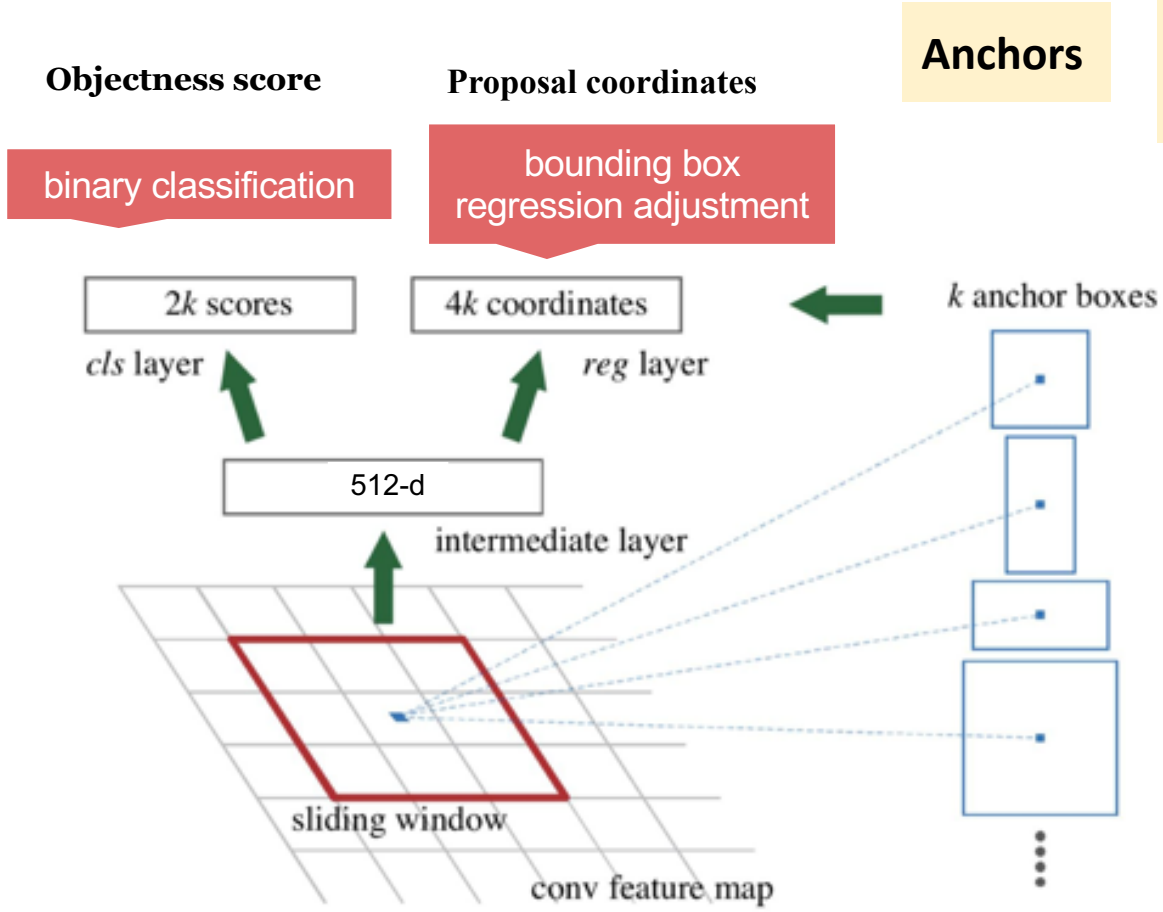
Last layer of Conv5 block 512-d



VGG16

RPN(Region Proposal Network)

How to generate a variable-length list of bounding boxes?



- **Does this anchor contain a relevant object?**
- **How would we adjust this anchor to better fit the relevant object?**

Objectness score: Intersection-over-Union(IoU)

overlap a ground-truth box

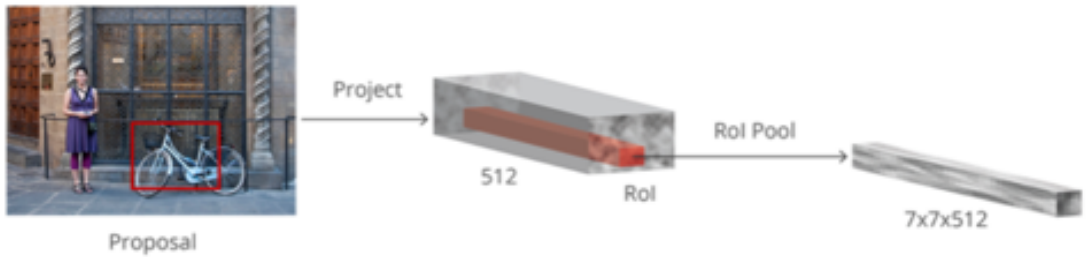
- Positive(foreground): IOU >0.7
- Negative(background): IOU <0.3
- Not included: IOU >0.3 or <0.7
- If zero positive(foreground) anchors: use anchors with the biggest IoU

Randomly sampling anchors to a mini batch of 256

- 128 positive
- 128 negative

ROI(Region of Interest) Pooling

How to take these bounding boxes and classify them into our desired categories.



Goals:

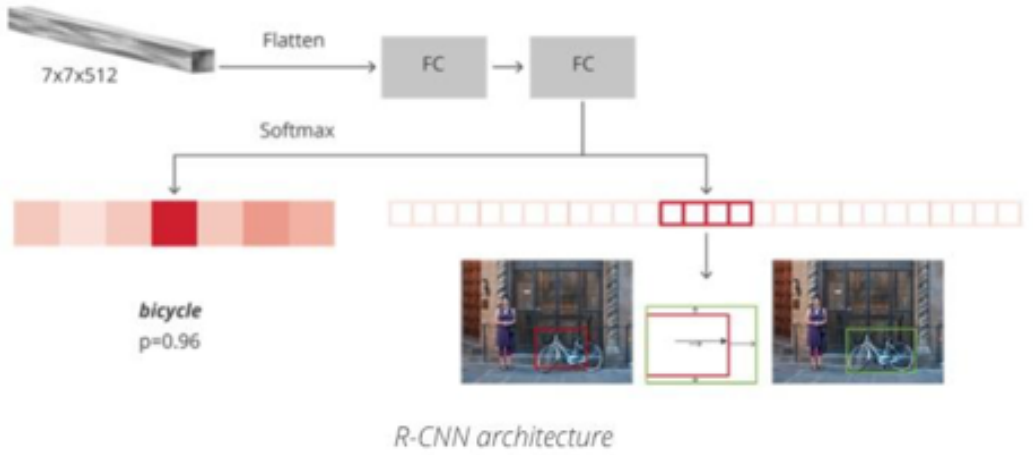
- Solves the problem of fixed image size requirement for object detection network.
- Improve efficiency

Simplest approach:
take each proposal, crop it, and then pass it through the pre-trained base network. Then, we can use the extracted features as input for a vanilla image classifier.

Inefficient and slow

Extracting fixed-sized feature maps for each proposal using region of interest pooling

R-CNN (Region-based Convolutional Neural Network)

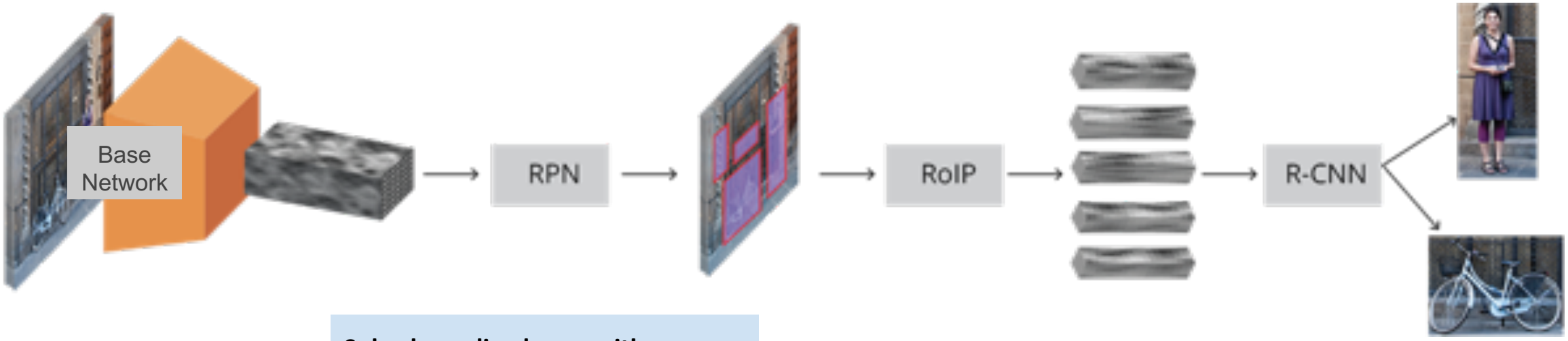


R-CNN architecture

Goals:

- Classify proposals into one of the classes
- Better adjust the bounding box for the proposal according to the predicted class.

Summary



- Extract features into feature map

Solve bounding boxes with variable-length

- Classification: if an anchor contain an object (“objectness” score)
- Regression: Final proposal coordinates

- Satisfy fixed image size requirements for R-CNN

- Classification: the content categories
- Regression: Adjust the bounding box coordinates

/06

Model Training

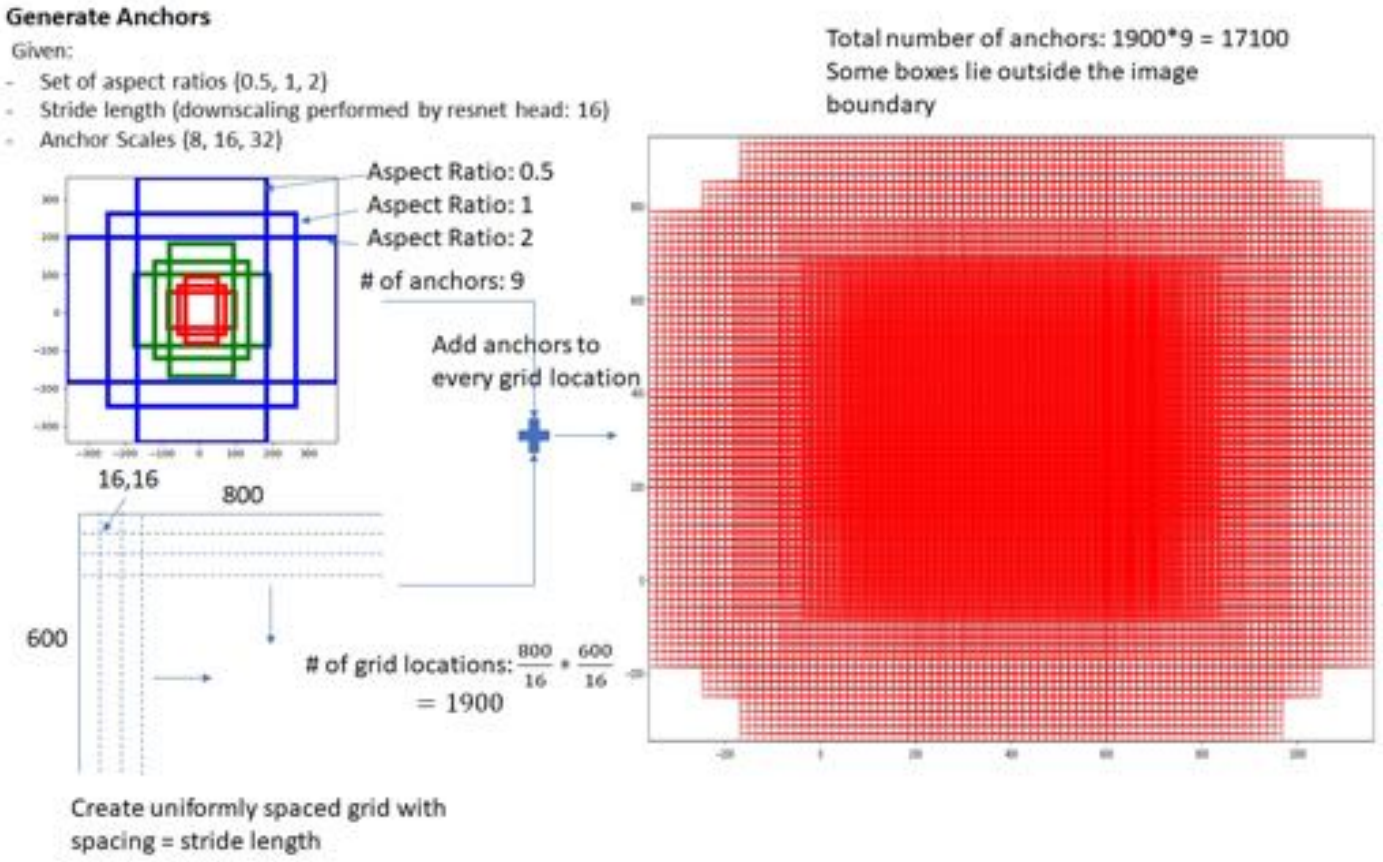
Hyperparameters

Common:

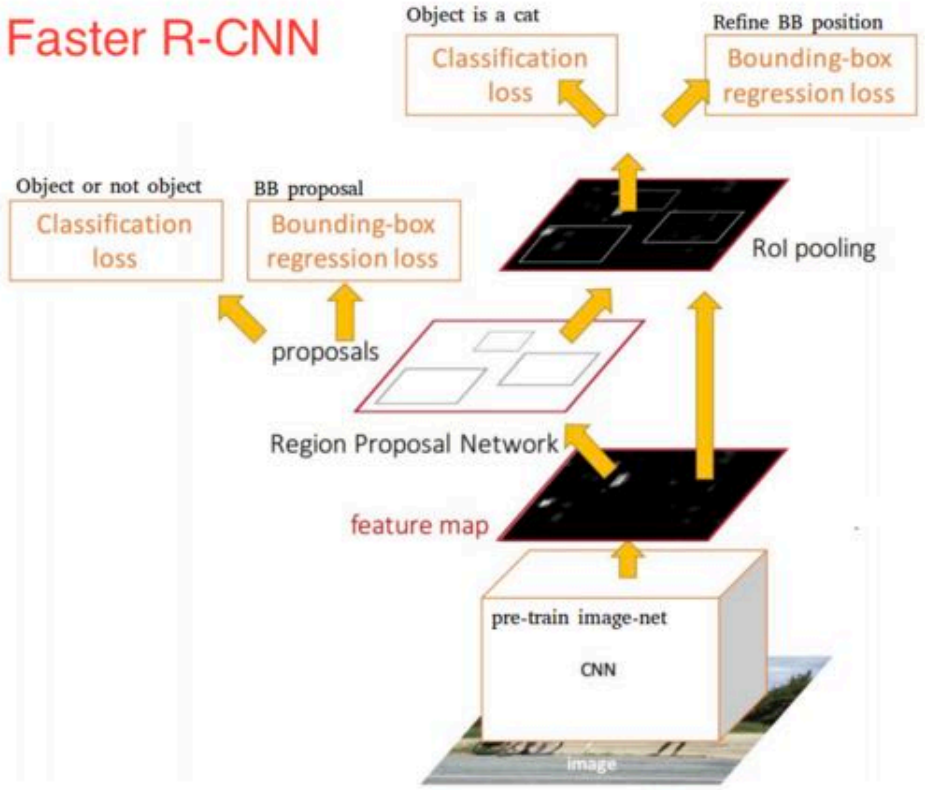
- Performance index (4 losses in this case)
- Optimization algorithm (Adam, SGD)
- Number of epoch

Uncommon:

- Size of image (shorter side get resized to 600 pixel)
- Proposal threshold value (0.7)
- Anchor aspect ratio (1:1, 1:2, 1:0.5)
- Anchor scale (64, 128, 256)
- Pre-trained model (resnet50, vgg16)



Performance Index



Multi-task loss formula:

$$L(\{p_i\},\{t_i\}) = \frac{1}{N_{cls}}\sum_i L_{cls}(p_i, p_i^*) + \lambda \frac{1}{N_{reg}}\sum_i p_i^* L_{reg}(t_i, t_i^*).$$

Parameterization of 4 coordinates of bounding box regression:

$$t_x = (x - x_a)/w_a, \quad t_y = (y - y_a)/h_a,$$
$$t_w = \log(w/w_a), \quad t_h = \log(h/h_a),$$
$$t_x^* = (x^* - x_a)/w_a, \quad t_y^* = (y^* - y_a)/h_a,$$
$$t_w^* = \log(w^*/w_a), \quad t_h^* = \log(h^*/h_a),$$

Training Screenshot

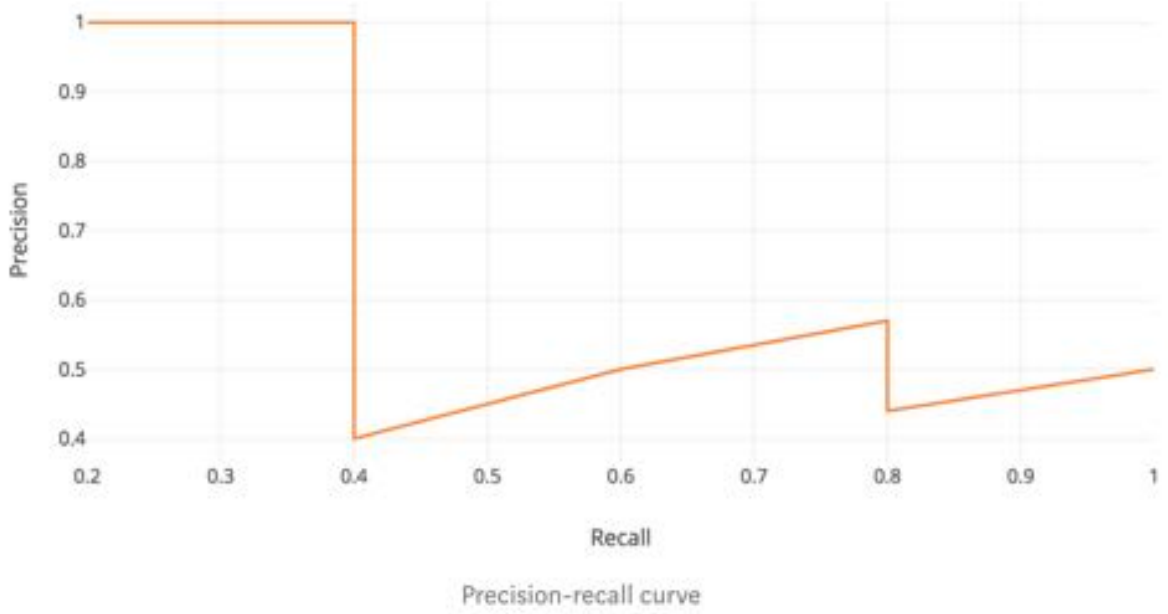
```
Epoch 50/2000
 997/1000 [==============================>.] - ETA: 2s - rpn_cls: 0.0846 - rpn_regr: 0
.0022 - detector_cls: 0.0327 - detector_regr: 0.0162Average number of overlapping bo
unding boxes from RPN = 9.454 for 1000 previous iterations
1000/1000 [==============================] - 672s 672ms/step - rpn_cls: 0.0847 - rpn
_regr: 0.0022 - detector_cls: 0.0328 - detector_regr: 0.0162
Mean number of bounding boxes from RPN overlapping ground truth boxes: 9.477
Classifier accuracy for bounding boxes from RPN: 0.9898125
Loss RPN classifier: 0.08468251236362775
Loss RPN regression: 0.0021800349318218 65
Loss Detector classifier: 0.032800445523636 58
Loss Detector regression: 0.016236067515565084
Elapsed time: 671.7986044883728
```

/07

Result

Result

cells	AP
ring	0.4595
difficult	0.9888
trophozoite	0.9640
leukocyte	0.8202
schizont	0.9734
gametocyte	1
mAP = 0.8677	

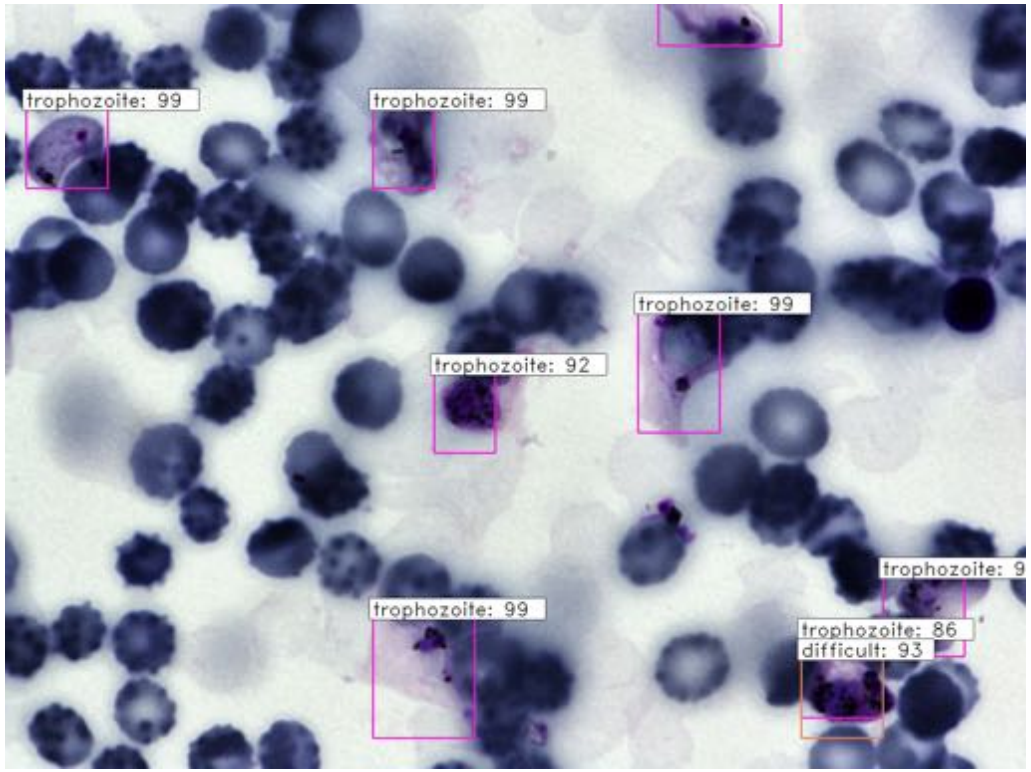


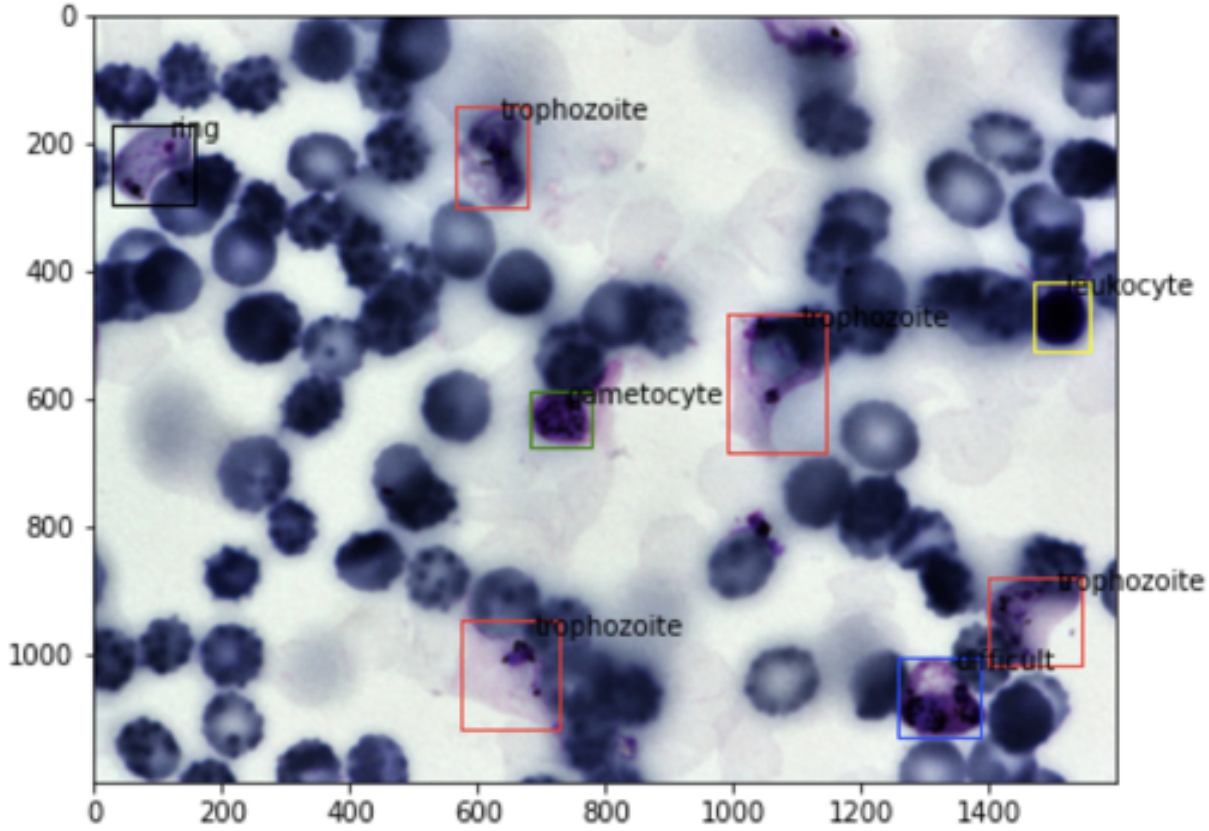
Precision-recall curve

The general definition for the Average Precision (AP) is finding the area under the precision-recall curve above.

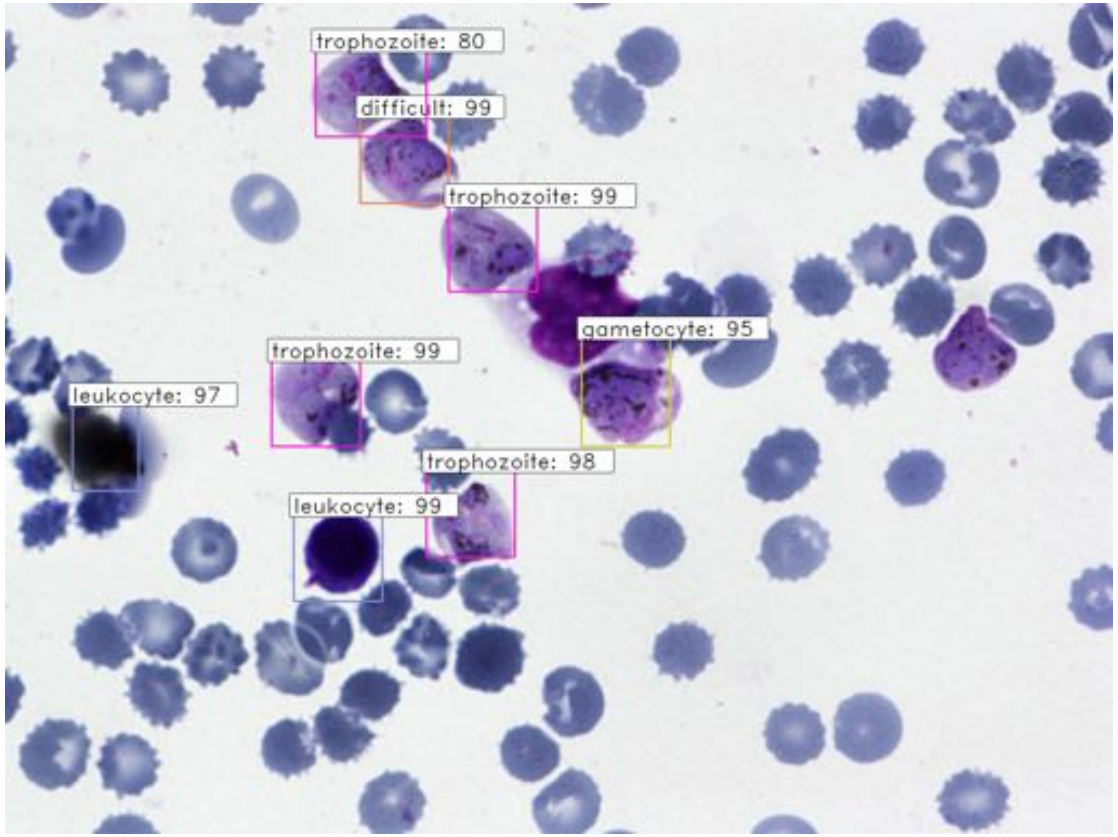
$$\mathbf{AP} = \int_0^1 p(r)dr$$

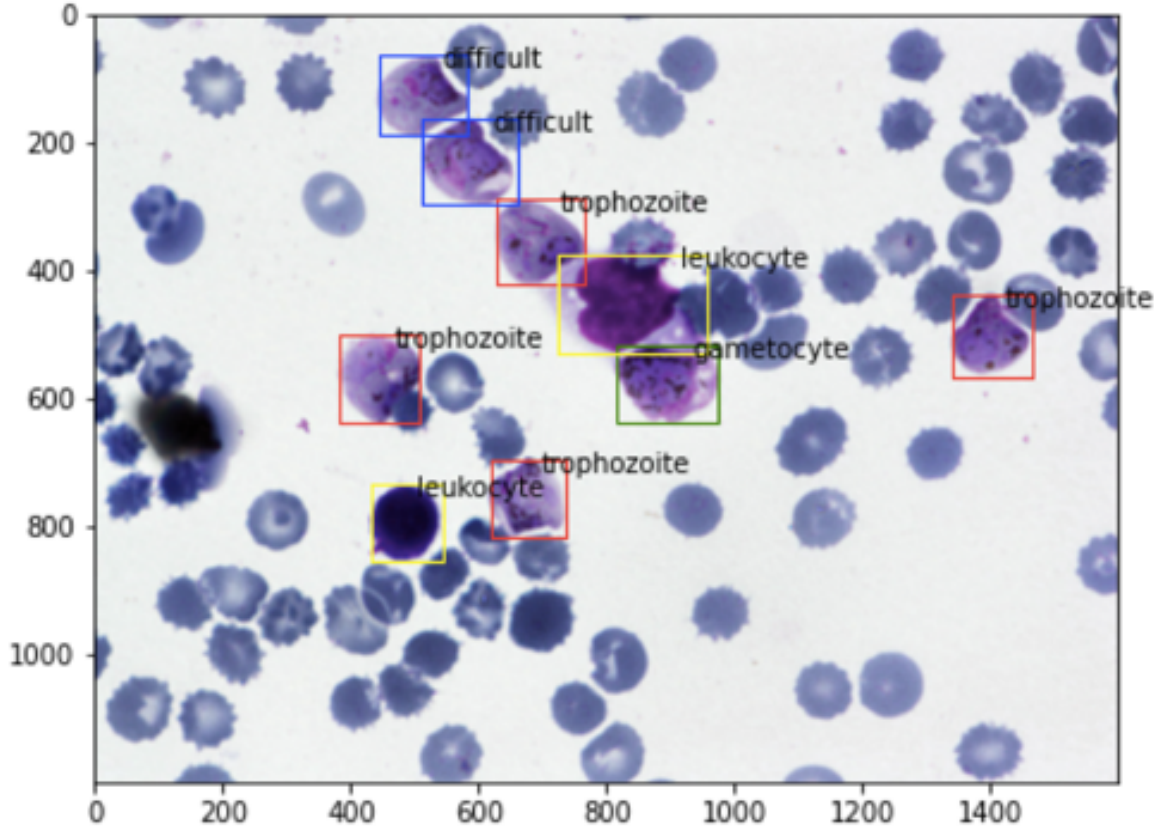
Result


trophozoite: 99
trophozoite: 99
trophozoite: 99
trophozoite: 92
trophozoite: 99
trophozoite: 86
difficult: 93


0
200
400
600
800
1000
0
200
400
600
800
1000
1200
1400
ring
trophozoite
gametocyte
trophozoite
trophozoite
difficult

Result


trophozoite: 80
difficult: 99
trophozoite: 99
gametocyte: 95
trophozoite: 99
leukocyte: 97
trophozoite: 98
leukocyte: 99


difficult
difficult
trophozoite
leukocyte
trophozoite
gametocyte
trophozoite
trophozoite
leukocyte
0
200
400
600
800
1000
0
200
400
600
800
1000
1200
1400

/08 Contribution & Conclusion

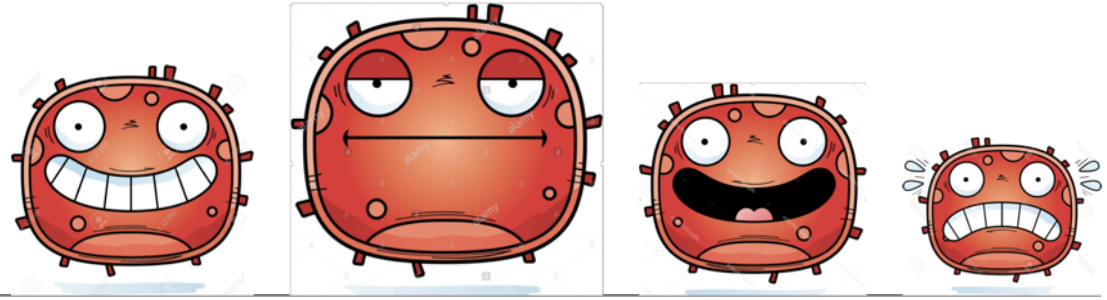
Contribution

Code wise!

- Added documentation for environment requirement for running this code
- Fixed error in train_frcnn.py where it does not load pre-trained network weights
- Fixed error in test_frcnn.py where the network's parameters should be frozen when being tested
- Fixed several bugs in the measure_map.py which it cannot run successfully

Conclusion and Future Improvement

- Understanding complex model before hyperparameter tuning is beneficial.
- This cell detection model could be improved if uncommon hyperparameters are being tuned.
- Develop systematic way to test the model.
- Use cross-validation to train this model.
- Use tensorboard to help visualize the loss during training.
- Use gridsearch, random search or bayesian search for finding better hyperparameters.
- Migrate from tensorflow 1.0 code to tensorflow 2.0 in order to achieve more efficient training and testing.



Bibliography

- https://arxiv.org/abs/1506.01497
- https://arxiv.org/abs/1512.03385
- https://tryolabs.com/blog/2018/01/18/faster-r-cnn-down-the-rabbit-hole-of-modern-object-detection/
- https://zhuanlan.zhihu.com/p/31426458
- https://www.analyticsvidhya.com/blog/2018/11/implementation-faster-r-cnn-python-object-detection/
- https://github.com/rbgirshick/py-faster-rcnn
- https://www.kdnuggets.com/2017/10/deep-learning-object-detection-comprehensive-review.htm
- https://medium.com/@jonathan_hui/map-mean-average-precision-for-object-detection-45c121a31173l

Thanks for your listening