A Practical Implementation of the Faster R-CNN Algorithm for Object Detection (Part 2 – with Python codes)

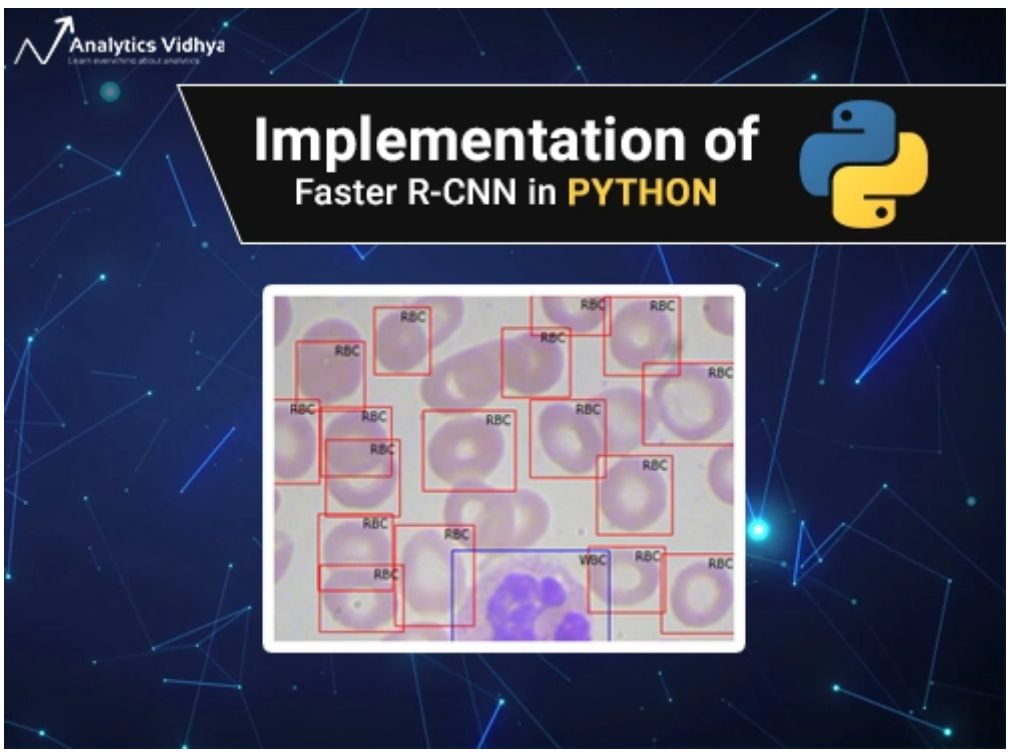
Pulkit Sharma — Published On November 4, 2018 and Last Modified On August 26th, 2021

Advanced Algorithm Computer Vision Deep Learning Image Object Detection Project Python Supervised Unstructured Data

Introduction

Which algorithm do you use for object detection tasks? I have tried out quite a few of them in my quest to build the most precise model in the least amount of time. And this journey, spanning multiple hackathons and real-world datasets, has usually always led me to the R-CNN family of algorithms.

It has been an incredible useful framework for me, and that's why I decided to pen down my learnings in the form of a series of articles. The aim behind this series is to showcase how useful the different types of R-CNN algorithms are. The first part received an overwhelmingly positive response from our community, and I'm thrilled to present part two!



In this article, we will first briefly summarize what we learned in part 1, and then deep dive into the implementation of the fastest member of the R-CNN family – Faster R-CNN. I highly recommend going through this article if you need to refresh your object detection concepts first: A Step-by-Step Introduction to the Basic Object Detection Algorithms (Part 1).

Part 3 of this series is published now and you can check it out here: A Practical Guide to Object Detection using the Popular YOLO Framework – Part III (with Python codes)

We will work on a very interesting dataset here, so let's dive right in!

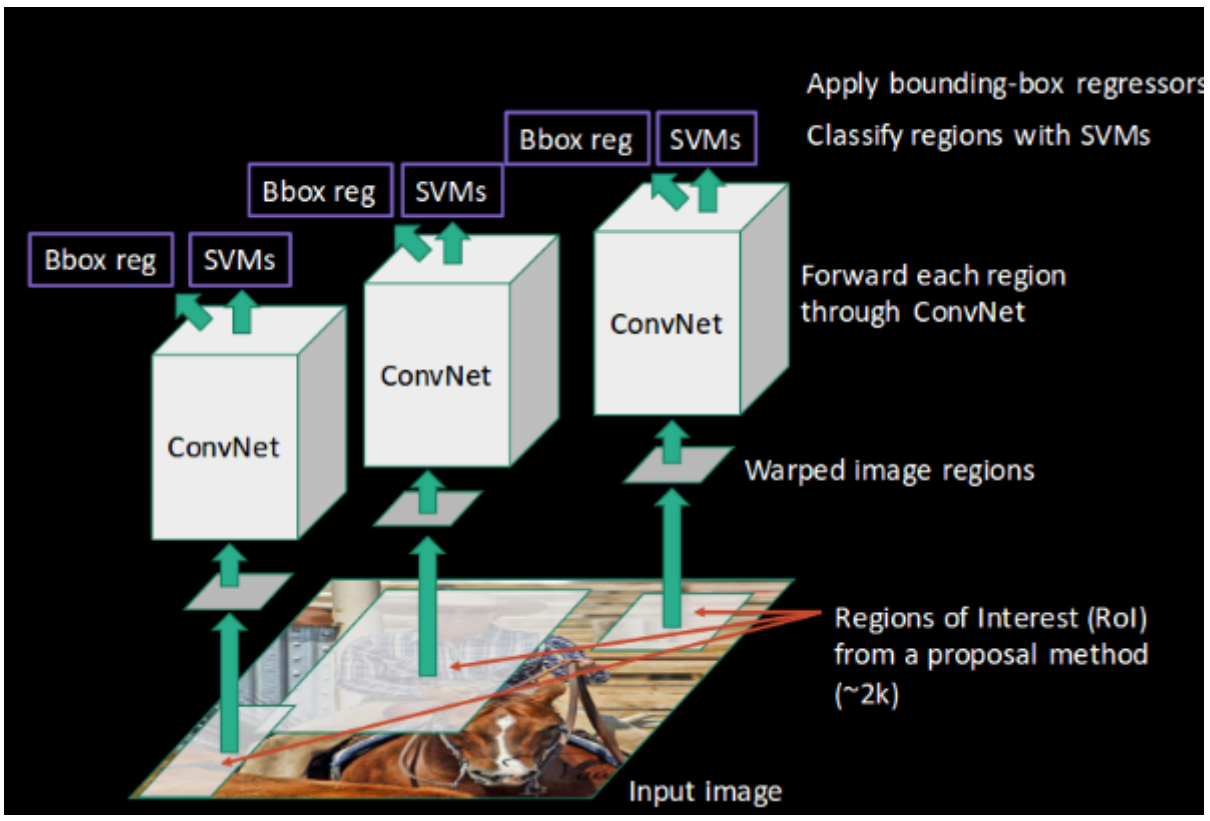
Table of Contents

A Practical Implementation of the Faster R-CNN Algorithm for Object Detection (Part 2 – with Python codes)

A Brief Overview of the Different R-CNN Algorithms for Object Detection

Let's quickly summarize the different algorithms in the R-CNN family (R-CNN, Fast R-CNN, and Faster R-CNN) that we saw in the first article. This will help lay the ground for our implementation part later when we will predict the bounding boxes present in previously unseen images (new data).

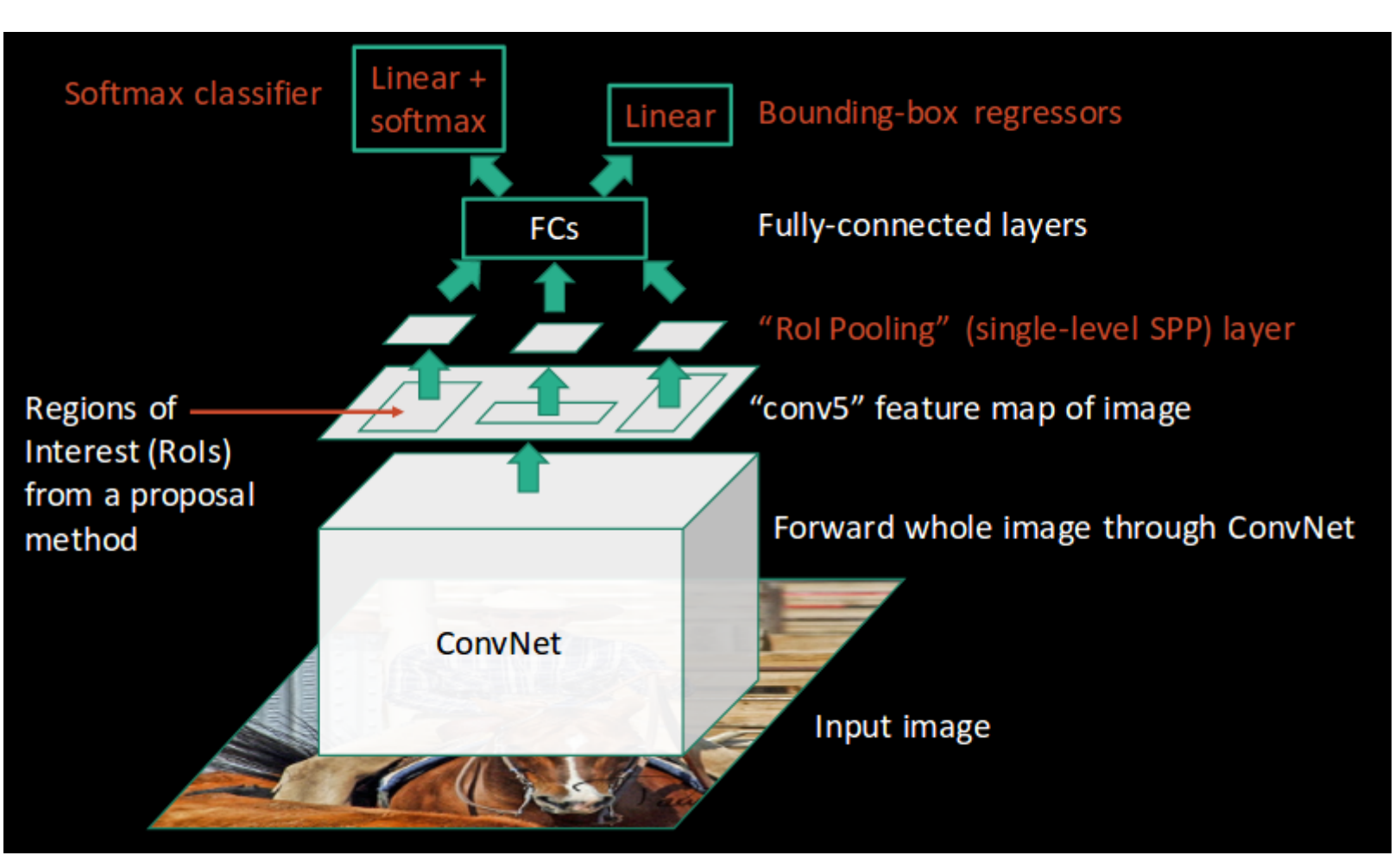
R-CNN extracts a bunch of regions from the given image using selective search, and then checks if any of these boxes contains an object. We first extract these regions, and for each region, CNN is used to extract specific features. Finally, these features are then used to detect objects. Unfortunately, R-CNN becomes rather slow due to these multiple steps involved in the process.



R-CNN

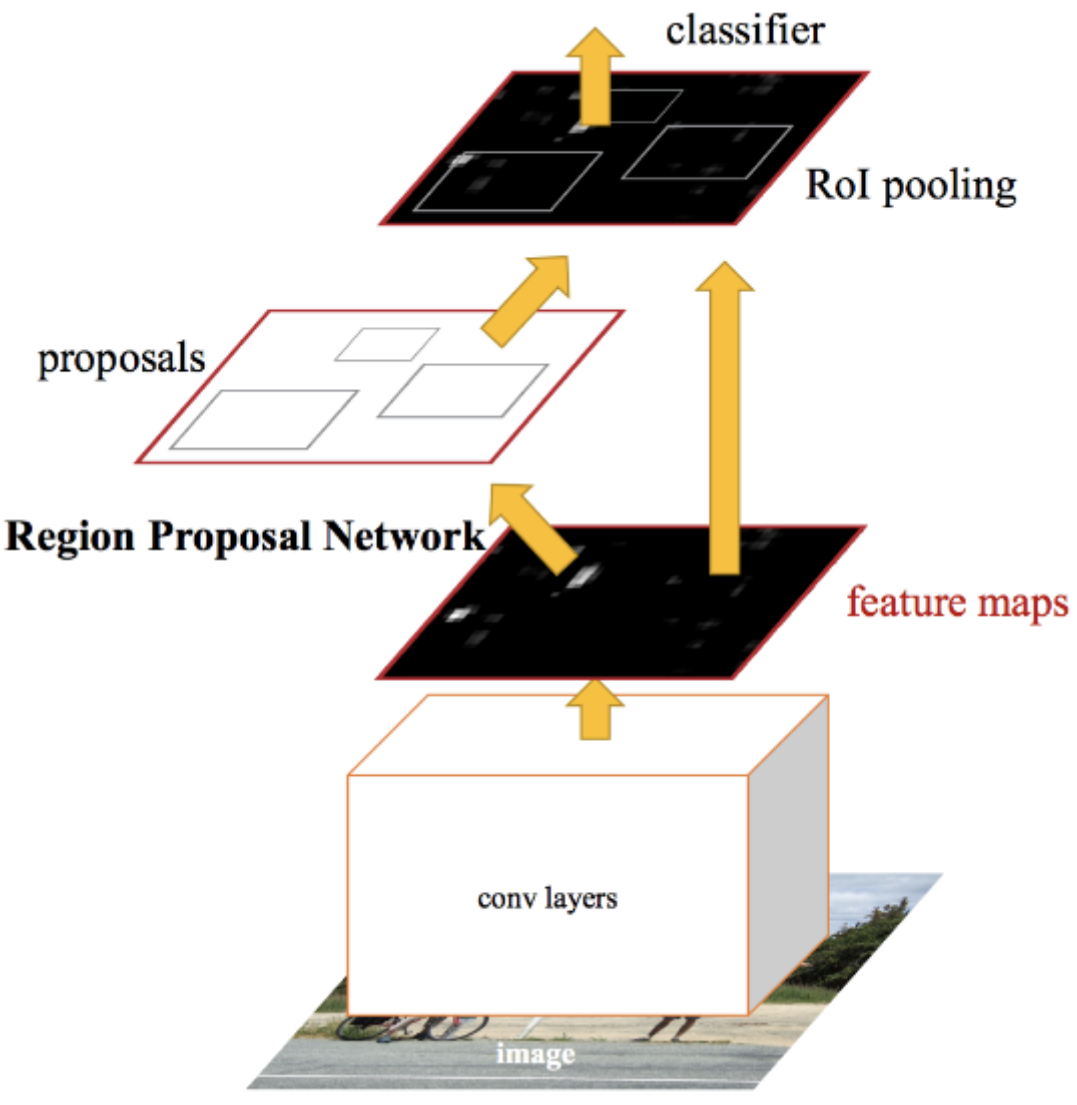
Fast R-CNN, on the other hand, passes the entire image to ConvNet which generates regions of interest (instead of passing the extracted regions from the image). Also, instead of using three different models (as we saw in R-CNN), it uses a single model which extracts features from the regions, classifies them into different classes, and returns the bounding boxes.

All these steps are done simultaneously, thus making it execute faster as compared to R-CNN. Fast R-CNN is, however, not fast enough when applied on a large dataset as it also uses selective search for extracting the regions.



A Practical Implementation of the Faster R-CNN Algorithm for Object Detection (Part 2 – with Python codes)

Faster R-CNN fixes the problem of selective search by replacing it with Region Proposal Network (RPN). We first extract feature maps from the input image using ConvNet and then pass those maps through a RPN which returns object proposals. Finally, these maps are classified and the bounding boxes are predicted.



Faster R-CNN

I have summarized below the steps followed by a Faster R-CNN algorithm to detect objects in an image:

1. Take an input image and pass it to the ConvNet which returns feature maps for the image
2. Apply Region Proposal Network (RPN) on these feature maps and get object proposals
3. Apply ROI pooling layer to bring down all the proposals to the same size
4. Finally, pass these proposals to a fully connected layer in order to classify any predict the bounding boxes for the image

What better way to compare these different algorithms than in a tabular format? So here you go!

Algorithm	Features	Prediction time / image	Limitations
CNN	Divides the image into multiple regions and then classifies each region into various classes.	–	Needs a lot of regions to predict accurately and hence high computation time.
R-CNN	Uses selective search to generate regions. Extracts around 2000 regions from each image.	40-50 seconds	High computation time as each region is passed to the CNN separately. Also, it uses three different models for making predictions.
Fast R-CNN	Each image is passed only once to the CNN and feature maps are extracted. Selective search is used on these maps to generate predictions. Combines all the three models used in R-CNN together.	2 seconds	Selective search is slow and hence computation time is still high.

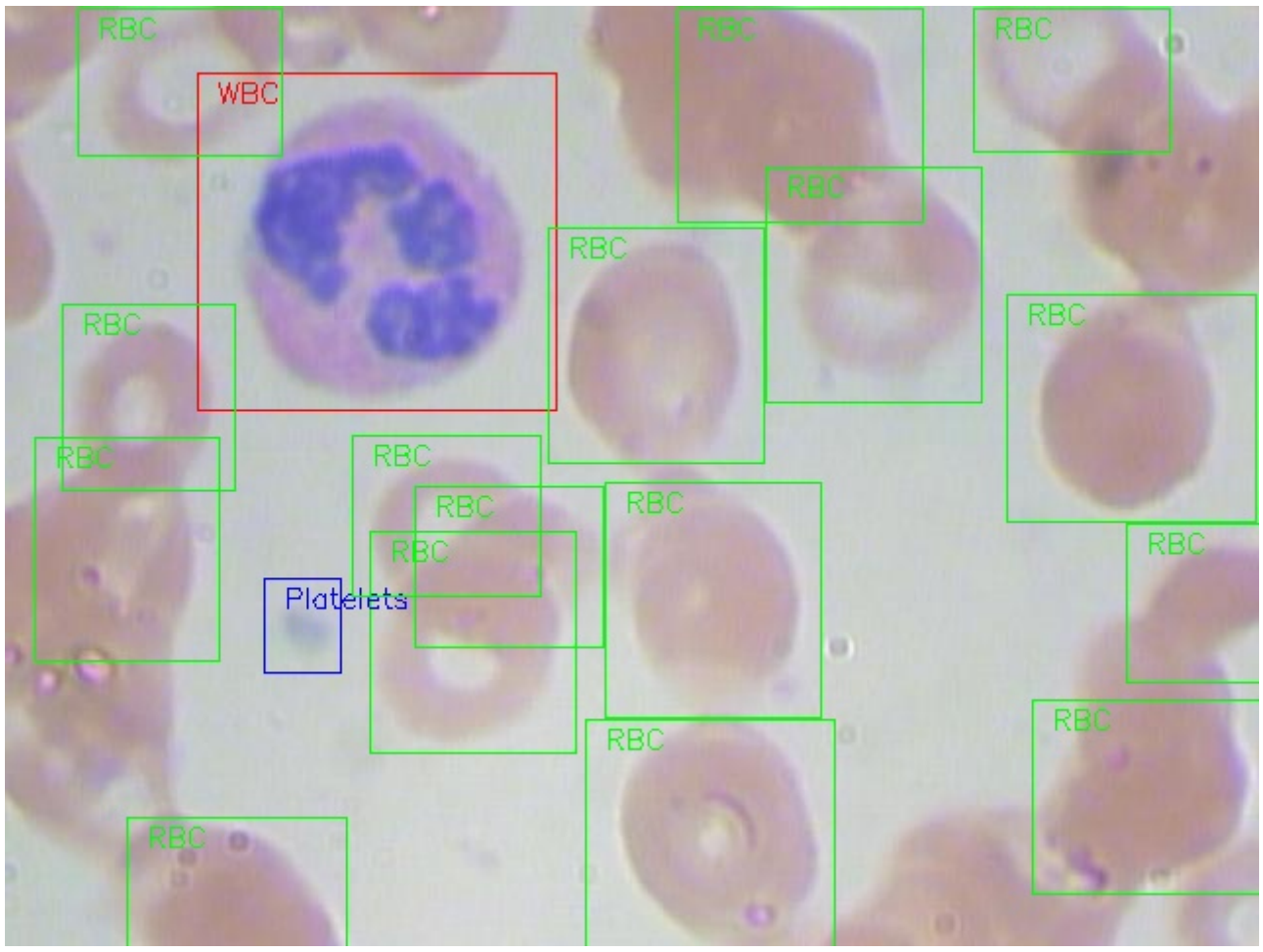
A Practical Implementation of the Faster R-CNN Algorithm for Object Detection (Part 2 – with Python codes)

Faster R-CNN	region proposal network (RPN) which makes the algorithm much faster.	0.2 seconds	one after the other, the performance of systems depends on how the previous system has performed.

Now that we have a grasp on this topic, it's time to jump from the theory into the practical part of our article. Let's implement Faster R-CNN using a really cool (and rather useful) dataset with potential real-life applications!

Understanding the Problem Statement

We will be working on a healthcare related dataset and the aim here is to solve a Blood Cell Detection problem. Our task is to detect all the Red Blood Cells (RBCs), White Blood Cells (WBCs), and Platelets in each image taken via microscopic image readings. Below is a sample of what our final predictions should look like:



The reason for choosing this dataset is that the density of RBCs, WBCs and Platelets in our blood stream provides a lot of information about the immune system and hemoglobin. This can help us potentially identify whether a person is healthy or not, and if any discrepancy is found in their blood, actions can be taken quickly to diagnose that.

Manually looking at the sample via a microscope is a tedious process. And this is where Deep Learning models play such a vital role. They can classify and detect the blood cells from microscopic images with impressive precision.

The full blood cell detection dataset for our challenge can be downloaded [from here](#). I have modified the data a tiny bit for the scope of this article:

- The bounding boxes have been converted from the given .xml format to a .csv format
- I have also created the training and test set split on the entire dataset by randomly picking images for the split

Note that we will be using the popular Keras framework with a TensorFlow backend in Python to train and build our model.

Setting up the System

A Practical Implementation of the Faster R-CNN Algorithm for Object Detection (Part 2 – with Python codes)

- pandas
- matplotlib
- tensorflow
- keras – 2.0.3
- numpy
- opencv-python
- sklearn
- h5py

Most of the above mentioned libraries will already be present on your machine if you have Anaconda and Jupyter Notebooks installed. Additionally, I recommend **downloading the requirement.txt file from** this link and use that to install the remaining libraries. Type the following command in the terminal to do this:

```
pip install -r requirement.txt
```

Alright, our system is now set and we can move on to working with the data!

Data Exploration

It's always a good idea (and frankly, a mandatory step) to first explore the data we have. This helps us not only unearth hidden patterns, but gain a valuable overall insight into what we are working with. The three files I have created out of the entire dataset are:

1. **train_images:** Images that we will be using to train the model. We have the classes and the actual bounding boxes for each class in this folder.
2. **test_images:** Images in this folder will be used to make predictions using the trained model. This set is missing the classes and the bounding boxes for these classes.
3. **train.csv:** Contains the name, class and bounding box coordinates for each image. There can be multiple rows for one image as a single image can have more than one object.

Let's read the .csv file (you can create your own .csv file from the original dataset if you feel like experimenting) and print out the first few rows. We'll need to first import the below libraries for this:

```
# importing required libraries
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
from matplotlib import patches
```

```
# read the csv file using read_csv function of pandas
train = pd.read_csv('train.csv')
train.head()
```

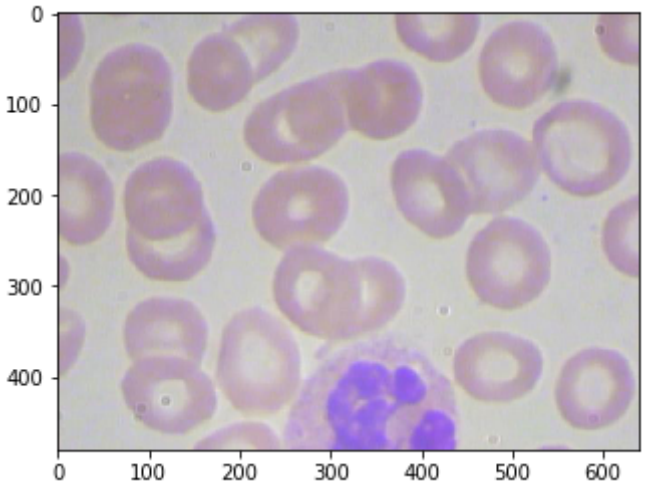
	image_names	cell_type	xmin	xmax	ymin	ymax
0	1.jpg	RBC	68	165	154	249
1	1.jpg	RBC	1	66	145	260
2	1.jpg	RBC	207	334	160	270
3	1.jpg	RBC	435	540	347	437
4	1.jpg	RBC	535	639	356	464

A Practical Implementation of the Faster R-CNN Algorithm for Object Detection (Part 2 – with Python codes)

2. **cell_type:** denotes the type of the cell
3. **xmin:** x-coordinate of the bottom left part of the image
4. **xmax:** x-coordinate of the top right part of the image
5. **ymin:** y-coordinate of the bottom left part of the image
6. **ymax:** y-coordinate of the top right part of the image

Let's now print an image to visualize what we're working with:

```
# reading single image using imread function of matplotlib
image = plt.imread('images/1.jpg')
plt.imshow(image)
```



This is what a blood cell image looks like. Here, the blue part represents the WBCs, and the slightly red parts represent the RBCs. Let's look at how many images, and the different type of classes, there are in our training set.

```
# Number of unique training images
train['image_names'].nunique()
```

So, we have 254 training images.

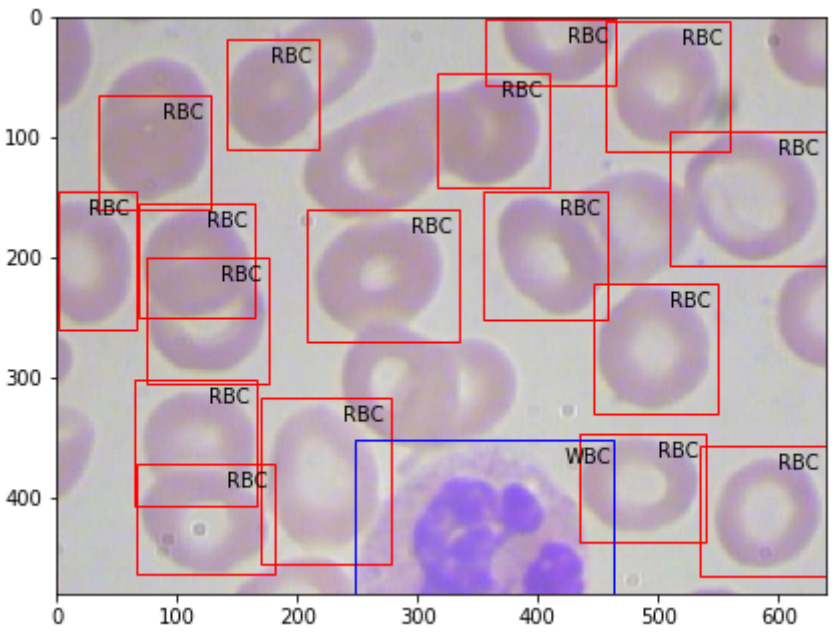
```
# Number of classes
train['cell_type'].value_counts()
```

```
RBC          2909
WBC           262
Platelets     252
Name: cell type, dtype: int64
```

We have three different classes of cells, i.e., RBC, WBC and Platelets. Finally, let's look at how an image with detected objects will look like:

A Practical Implementation of the Faster R-CNN Algorithm for Object Detection (Part 2 – with Python codes)

```
#add axes to the image
ax = fig.add_axes([0,0,1,1])

# read and plot the image
image = plt.imread('images/1.jpg')
plt.imshow(image)

# iterating over the image for different objects
for _,row in train[train.image_names == "1.jpg"].iterrows():
    xmin = row.xmin
    xmax = row.xmax
    ymin = row.ymin
    ymax = row.ymax

    width = xmax - xmin
    height = ymax - ymin

    # assign different color to different classes of objects
    if row.cell_type == 'RBC':
        edgecolor = 'r'
        ax.annotate('RBC', xy=(xmax-40,ymin+20))
    elif row.cell_type == 'WBC':
        edgecolor = 'b'
        ax.annotate('WBC', xy=(xmax-40,ymin+20))
    elif row.cell_type == 'Platelets':
        edgecolor = 'g'
        ax.annotate('Platelets', xy=(xmax-40,ymin+20))

    # add bounding boxes to the image
    rect = patches.Rectangle((xmin,ymin), width, height, edgecolor = edgecolor, facecolor = 'none')

    ax.add_patch(rect)
```



This is what a training example looks like. We have the different classes and their corresponding bounding boxes. Let's now train our model on these images. We will be using the *keras_frcnn* library to train our model as well as to get predictions on the test images.

Implementing Faster R-CNN

For implementing the Faster R-CNN algorithm, we will be following the steps mentioned in this Github repository. So as the first step, make sure you clone this repository. Open a new terminal window and type the following to do this:

A Practical Implementation of the Faster R-CNN Algorithm for Object Detection (Part 2 – with Python codes)

Move the *train_images* and *test_images* folder, as well as the *train.csv* file, to the cloned repository. In order to train the model on a new dataset, the format of the input should be:

```
filepath,x1,y1,x2,y2,class_name
```

where,

- *filepath* is the path of the training 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

We need to convert the .csv format into a .txt file which will have the same format as described above. Make a new dataframe, fill all the values as per the format into that dataframe, and then save it as a .txt file.

```
data = pd.DataFrame()
data['format'] = train['image_names']

# as the images are in train_images folder, add train_images before the image name
for i in range(data.shape[0]):
    data['format'][i] = 'train_images/' + data['format'][i]

# add xmin, ymin, xmax, ymax and class as per the format required
for i in range(data.shape[0]):
    data['format'][i] = data['format'][i] + ',' + str(train['xmin'][i]) + ',' + str(train['ymin'][i]) + ',' +
str(train['xmax'][i]) + ',' + str(train['ymax'][i]) + ',' + train['cell_type'][i]

data.to_csv('annotate.txt', header=None, index=None, sep=' ')
```

What's next?

Train our model! We will be using the *train_frcnn.py* file to train the model.

```
cd keras-frcnn
python train_frcnn.py -o simple -p annotate.txt
```

It will take a while to train the model due to the size of the data. If possible, you can use a GPU to make the training phase faster. You can also try to reduce the number of epochs as an alternate option. To change the number of epochs, go to the *train_frcnn.py* file in the cloned repository and change the *num_epochs* parameter accordingly.

Every time the model sees an improvement, the weights of that particular epoch will be saved in the same directory as "*model_frcnn.hdf5*". These weights will be used when we make predictions on the test set.

It might take a lot of time to train the model and get the weights, depending on the configuration of your machine. I suggest using the weights I've got after training the model for around 500 epochs. **You can download these weights from here**. Ensure you save these weights in the cloned repository.

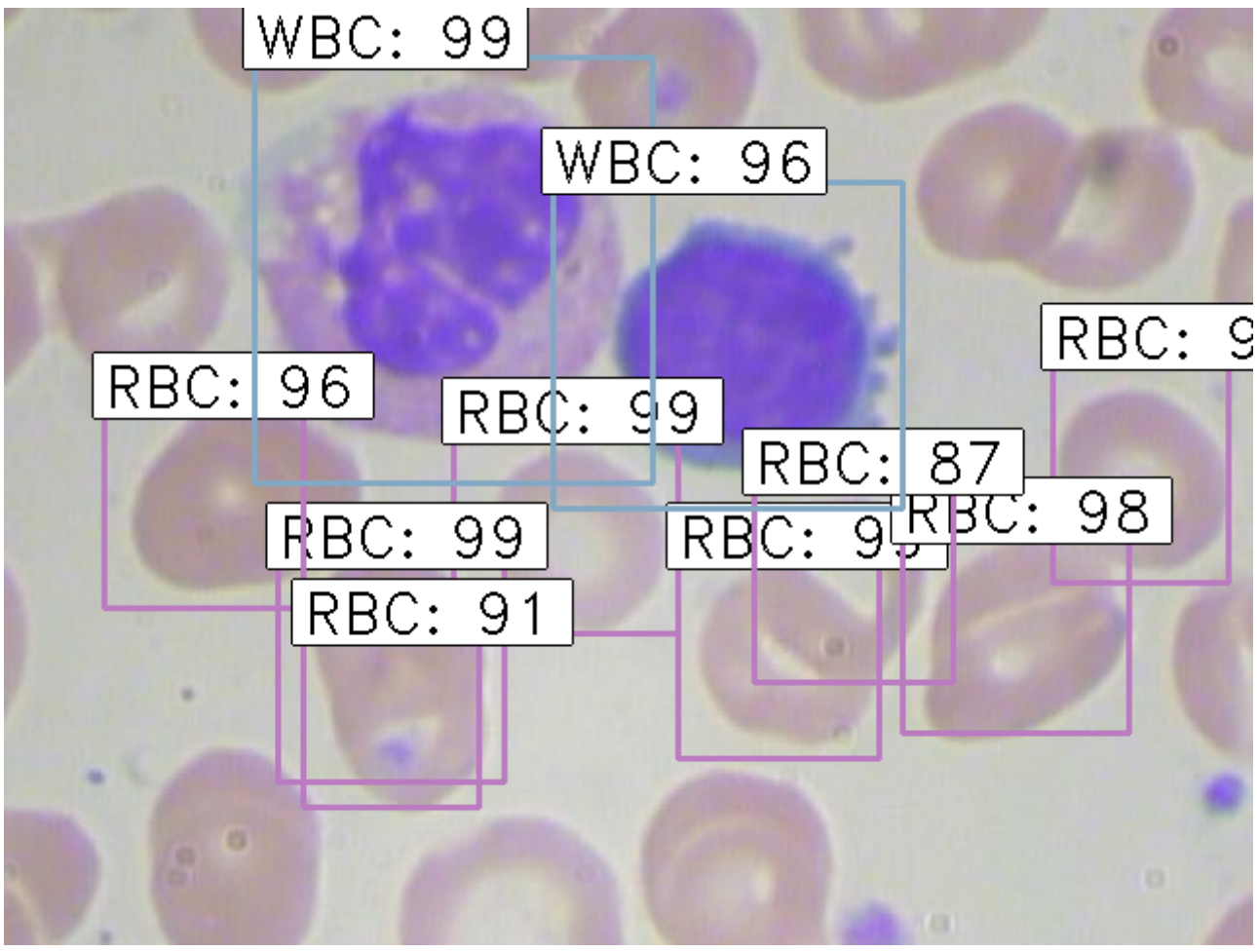
So our model has been trained and the weights are set. It's prediction time! *Keras_frcnn* makes the predictions for the new images and saves them in a new folder. We just have to make two changes in the *test_frcnn.py* file to save the images:

1. Remove the comment from the last line of this file:
   cv2.imwrite('./results_imgs/{}.png'.format(idx),img)
2. Add comments on the second last and third last line of this file:
   # cv2.imshow('img', img)

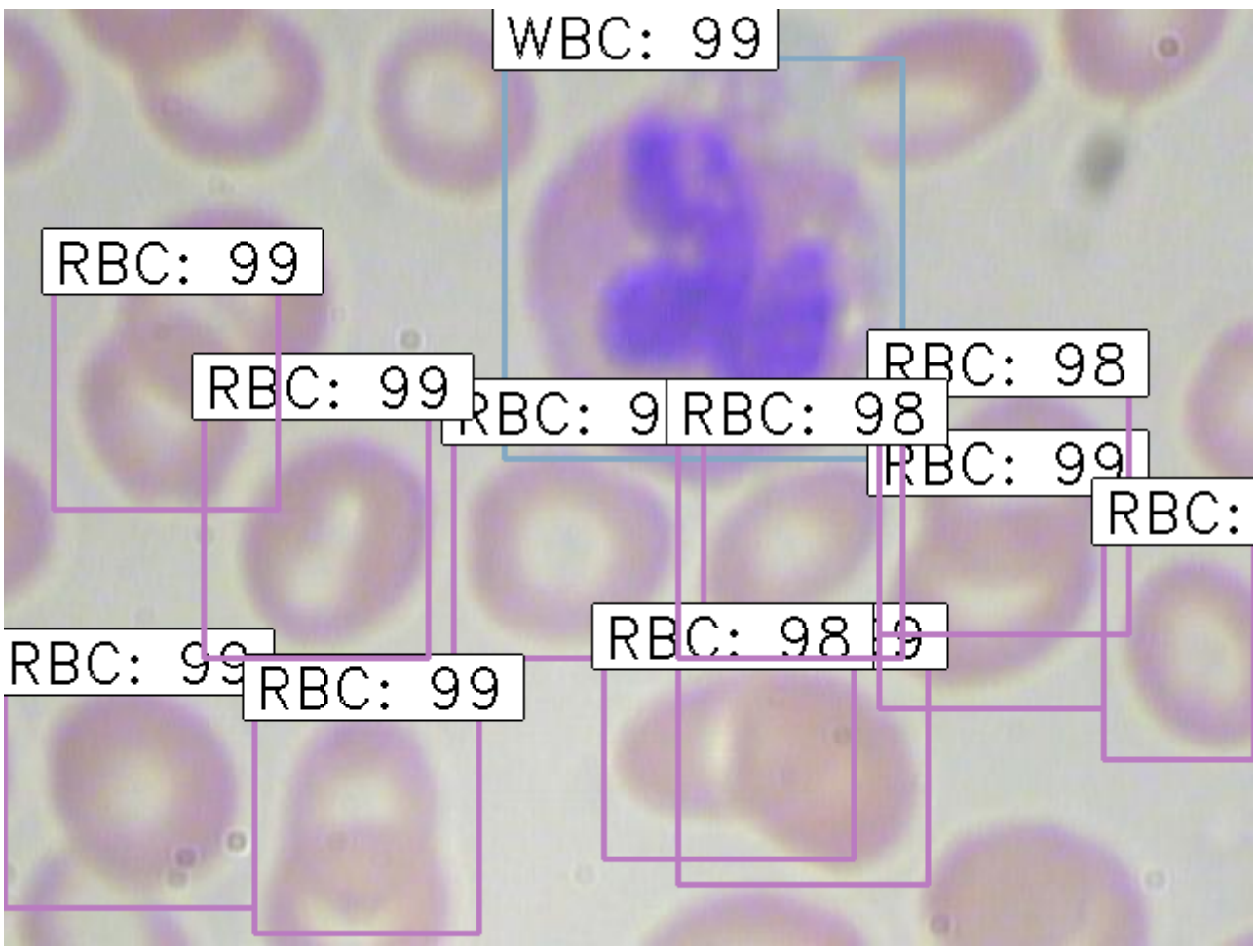
A Practical Implementation of the Faster R-CNN Algorithm for Object Detection (Part 2 – with Python codes)

```
python test_frcnn.py -p test_images
```

Finally, the images with the detected objects will be saved in the "results_imgs" folder. Below are a few examples of the predictions I got after implementing Faster R-CNN:

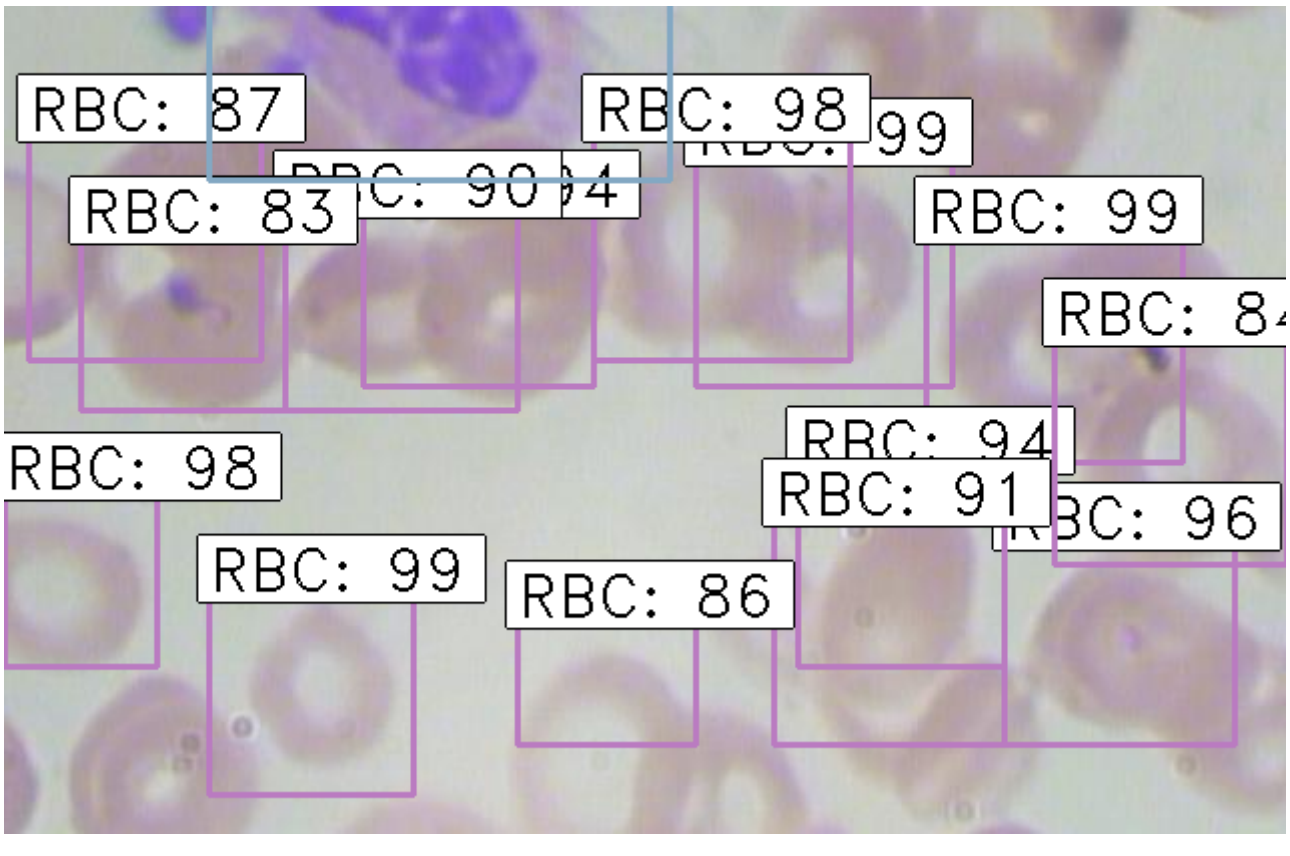


Result 1

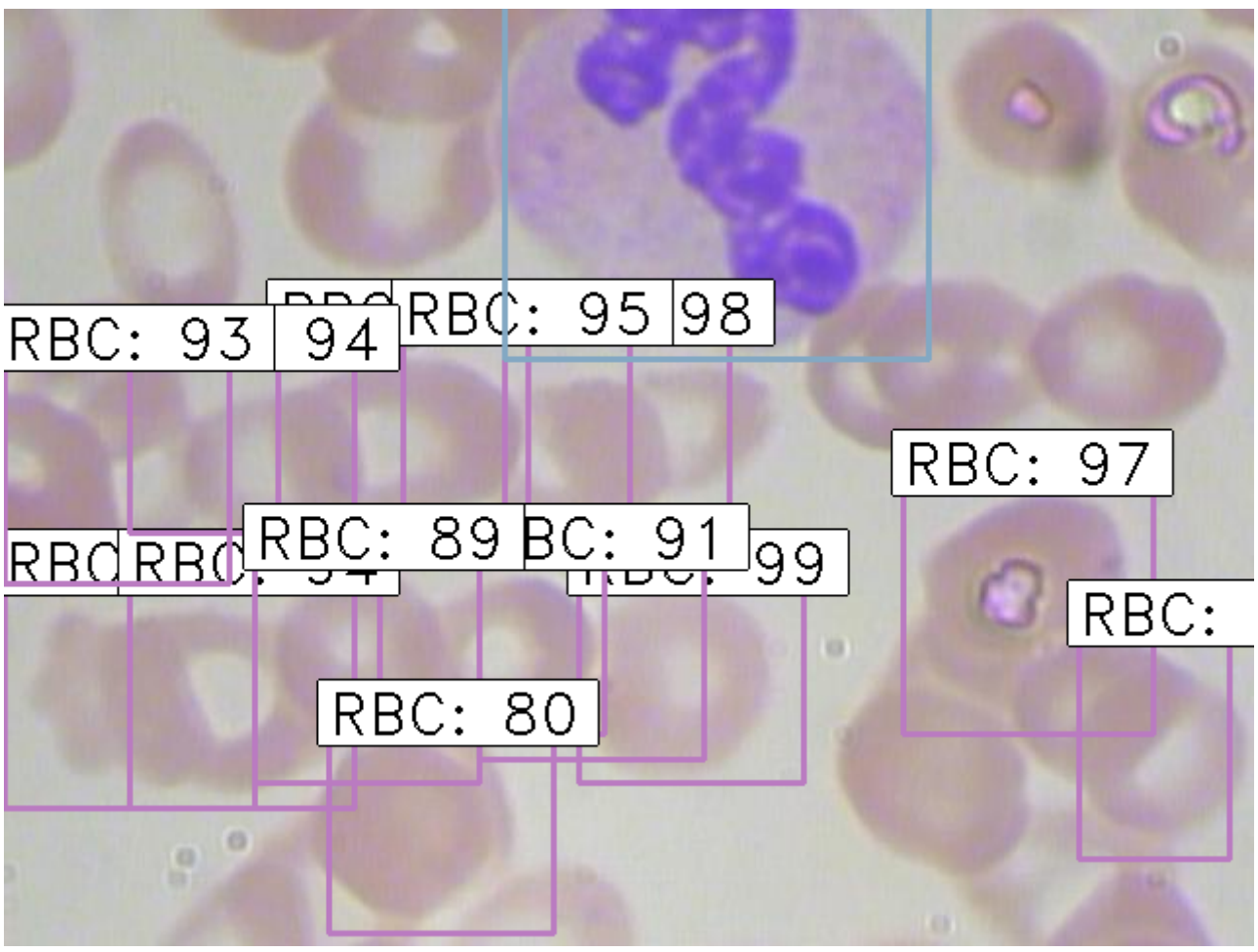


Result 2

A Practical Implementation of the Faster R-CNN Algorithm for Object Detection (Part 2 – with Python codes)



Result 3



Result 4

End Notes

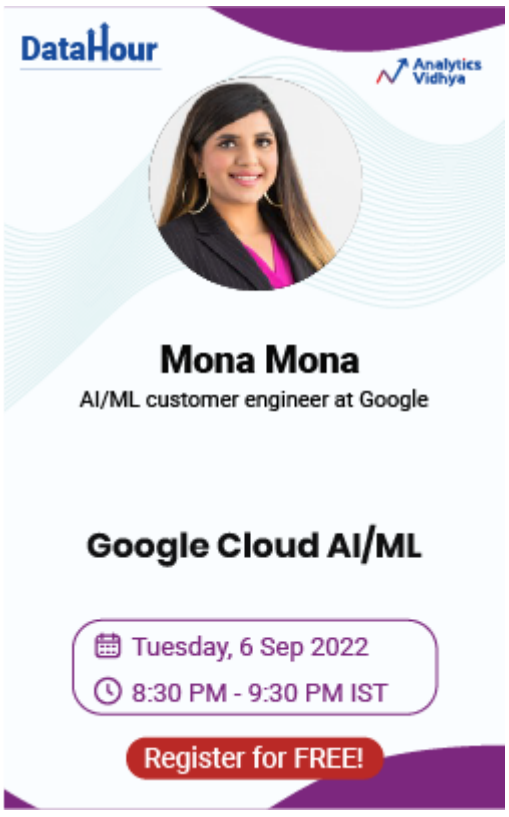
R-CNN algorithms have truly been a game-changer for object detection tasks. There has suddenly been a spike in recent years in the amount of computer vision applications being created, and R-CNN is at the heart of most of them.

Keras_frcnn proved to be an excellent library for object detection, and in the next article of this series, we will focus on more advanced techniques like YOLO, SSD, etc.

If you have any query or suggestions regarding what we covered here, feel free to post them in the comments section below and I will be happy to connect with you!

A Practical Implementation of the Faster R-CNN Algorithm for Object Detection (Part 2 – with Python codes)

[object detection algorithms] [python]



About the Author

Pulkit Sharma

My research interests lies in the field of Machine Learning and Deep Learning. Possess an enthusiasm for learning new skills and technologies.

Our Top Authors

Download

Analytics Vidhya App for the Latest blog/Article

Previous Post

Top 5 Machine Learning GitHub Repositories & Reddit Discussions (October 2018)

Next Post

Want to Become a Data Engineer? Here's a Comprehensive List of Resources to get Started

139 thoughts on "A Practical Implementation of the Faster R-CNN Algorithm for Object Detection (Part 2 – with Python codes)"

sset says:
November 05, 2018 at 8:40 pm

A Practical Implementation of the Faster R-CNN Algorithm for Object Detection (Part 2 – with Python codes)

Vishnu says:
November 06, 2018 at 1:54 am

Hey Pulkit, I am a Freshman at UIUC studying CS and one of my projects is in the same domain. Would it be possible to connect with you and talk more about this?

Reply

Pulkit Sharma says:
November 06, 2018 at 12:11 pm

Hi, I believe Faster RCNN works good enough for small objects as well. Currently, I am working on YOLO and SSD and will share my learnings on how they deal with small objects.

Reply

Pulkit Sharma says:
November 06, 2018 at 12:24 pm

Hi Vishnu, You can ask any query related to this project here. I will try my best to respond to them.

Reply

Toni says:
November 06, 2018 at 3:15 pm

The dataset that you have provided, it doesn't correspond with your example. Can you provide the right source?

Reply

Pulkit Sharma says:
November 06, 2018 at 3:26 pm

Hi Toni, The original dataset is available on GitHub, link for the same is provided in the article. I have made some changes in that dataset and have mentioned the changes as well. You just have to convert the dataset in csv format.

Reply

Toni says:
November 06, 2018 at 3:43 pm

Can you please provide the code to do that? Thank you

Reply

Pulkit Sharma says:
November 06, 2018 at 3:58 pm

Hi, I have shared the code in the previous comments. Please check.

Reply

saurabh says:
November 06, 2018 at 5:18 pm

I am having issues reading the data which is in .rec format

Reply

laurent cesaro says:
November 08, 2018 at 12:00 am

Hi, Thanks for this tutorial ! Do you have installed the last package of tensorflow here ? Thanks Laurent cESARO

Reply

shravankumar P says:
November 08, 2018 at 11:26 am

Very well documented all your learnings, thanks for sharing. Keep it going, all the best.

Reply

Michael Thomas says:
November 12, 2018 at 8:18 am

Hi, I'm wondering if it's possible to load existing weights from a pre-trained model such as Imagenet? Also how does the faster

A Practical Implementation of the Faster R-CNN Algorithm for Object Detection (Part 2 – with Python codes)

Shilpa says:
November 12, 2018 at 8:51 am

Hi Pulkit, Can you please also share the config file generated after training the model? Test_frcnn is looking for config file as well.

Reply

Pulkit Sharma says:
November 12, 2018 at 11:05 am

Hi Saurabh, You can convert the coordinates to csv format and then read it in python.

Reply

Pulkit Sharma says:
November 12, 2018 at 11:13 am

Hi, Yes, I have the latest version of Tensorflow. You can install the latest version too.

Reply

Pulkit Sharma says:
November 12, 2018 at 11:15 am

Thank You!

Reply

Pulkit Sharma says:
November 12, 2018 at 11:28 am

Hi Michael, We can use weights from any of the model which is trained for object detection. Faster RCNN predicts the bounding box coordinates whereas, Mask RCNN is used for pixel-wise predictions. So, it totally depends on the type of problem that you want to solve. In my opinion, both of these algorithms are good and can be used depending on the type of problem in hand.

Reply

Pulkit Sharma says:
November 12, 2018 at 11:35 am

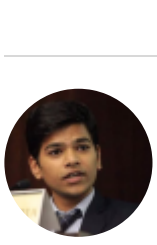
Hi Shilpa, You can download the config file from this link.

Reply

Sanjoy Datta says:
November 13, 2018 at 1:27 pm

Been able to follow your blog so far. Now stuck at preparing input data. Need more help to convert data as in github to familiar csv formats. Without it not able to proceed with the notebook

Reply



Pulkit Sharma says:
November 13, 2018 at 2:52 pm

Hi Sanjoy, I have used the below given code to convert the bounding boxes into csv file:

```
%pylab inline import os, sys, random import xml.etree.ElementTree as ET from glob import glob import pandas as pd from shutil import copyfile annotations = glob('BCCD/Annotations/*.xml') df = [] cnt = 0 for file in annotations: prev_filename = file.split('/') [-1].split('.')[0] + '.jpg' filename = str(cnt) + '.jpg' row = [] parsedXML = ET.parse(file) for node in parsedXML.getroot().iter('object'): blood_cells = node.find('name').text xmin = int(node.find('bndbox/xmin').text) xmax = int(node.find('bndbox/xmax').text) ymin = int(node.find('bndbox/ymin').text) ymax = int(node.find('bndbox/ymax').text) row = [prev_filename, filename, blood_cells, xmin, xmax, ymin, ymax] df.append(row) cnt += 1 data = pd.DataFrame(df, columns=['prev_filename', 'filename', 'cell_type', 'xmin', 'xmax', 'ymin', 'ymax']) data[['filename', 'cell_type', 'xmin', 'xmax', 'ymin', 'ymax']].to_csv('blood_cell_detection.csv', index=False)
```

Reply

Sanjoy Datta says:
November 14, 2018 at 1:35 am

Dear Pulkit, Thanks a ton for the code. Ran it. But it is not populating the variable based on 'annotations = glob(.....)'. Leading to: len(annotations) = 0. Hence the blood_cell_detection.csv it is creating is with only the header. No data is coming in. I must be doing some silly mistake. But not able to catch it. Got stuck again. What am I doing wrong? Sorry for bothering you again. Regards Sanjoy.

A Practical Implementation of the Faster R-CNN Algorithm for Object Detection (Part 2 – with Python codes)

November 14, 2018 at 11:34 am

Hi Sanjoy, Make sure to give the correct location of .xml files inside glob(). Replace 'BCCD/Annotations/*.xml' with the location where your .xml files are saved.

Reply

Sanjoy Datta says:
November 14, 2018 at 2:11 pm

Thank you Pulkit for prompt response again. Problem is elsewhere now. In short, do not have the xml and jpeg files yet at local folder. Have downloaded train.rec and val.rec. Not been able to extract .xml and .jpeg files out of them. So problem is stated as: 1. How to extract xml and jpeg files from .rec files? 2. Or, is there a way to download files directly from respective github folders? Been searching stackoverflow. But not found a way It appears from other posts, solution will help a few others like me too. Thank you in advance Sanjoy

Reply

Pulkit Sharma says:
November 14, 2018 at 3:05 pm

Hi Sanjoy, You can clone this GitHub repository which contains the JPEG Images and annotations inside the BCCD folder.

Reply

Sonal says:
November 14, 2018 at 9:49 pm

Thanks for the article. I am a beginner in ML / DS field. Can I use this code to detect the Region of Interest (ROI) from Glaucoma images?

Reply

---

ping wan says:
November 15, 2018 at 9:29 am

Hi, Pulkit, Thanks for this wonderful tutorial ! I have downloaded the BCCD dataset, but there is no train_images and test_images folder, as well as the train.csv file in it, shall I generate them by myself? Thank you!

Reply

Pulkit Sharma says:
November 15, 2018 at 11:04 am

Hi Sonal, Yes, you can use this for your own dataset as well. You just have to make a .txt file in the same format as I have used.

Reply

Pulkit Sharma says:
November 15, 2018 at 11:25 am

Hi, Yes, you have to create the training set, testing set and csv file by yourself. You can divide the images into training and testing set in any ratio.

Reply

Sanjoy Datta says:
November 15, 2018 at 2:15 pm

Thank you Pulkit. It is working finally. Thanks for all your help.

Reply

---

ping wan says:
November 15, 2018 at 2:38 pm

Thank you for your prompt reply! Best regards.

Reply

Hareesh Kumar says:
November 16, 2018 at 7:01 pm

Hi , Thanks for the article. I am not able to train model on tensorflow-gpu. I get error "ValueError: Shape must be rank 1 but is rank 0 for 'bn_conv1_7/Reshape_4' (op: 'Reshape') with input shapes: [1,1,1,64], []." Keras version on server : 2.2.4 Tensorflow version on server: 1.11.0 The training works fine without any issues on cpu machine. Please help.

Reply

A Practical Implementation of the Faster R-CNN Algorithm for Object Detection (Part 2 – with Python codes)

November 16, 2018 at 7:45 pm

Hi Hareesh, I have used the following version of Keras and Tensorflow: Keras - 2.2.0 Tensorflow - 1.8.0 And the codes run smoothly without any error.

Reply

---

sset says:
November 17, 2018 at 12:31 am

How do we know when to stop training? For very small object detections any parameters to tune? Any anchor sizes, RPN to change? Thanks

Reply

Pulkit Sharma says:
November 17, 2018 at 11:13 am

Hi, You can make a validation set from the training data and if the validation accuracy stops to increase or if it starts to decrease, you can say that the model is overfitting and you can stop the training at that point. Regarding the hyperparameter tuning, you can always try different set of hyperparameters to check whether they improve the model performance. There is no clear-cut answer to what hyperparameters should be used. You always have to try a range of hyperparameters and select the one which gives highest performance. Same goes with anchor sizes and RPN.

Reply

Shilpa says:
November 18, 2018 at 5:10 am

Thanks !!

Reply

---

ping wan says:
November 18, 2018 at 2:06 pm

I met the same problem as yours. After I use Keras – 2.2.0 and Tensorflow – 1.8.0, the problem is fixed.

Reply

Hareesh Kumar says:
November 19, 2018 at 12:06 pm

Hi , Can you tell us how did you arrive at the img_channel_mean, classifier_regr_std values and whats is the need of it? # image channel-wise mean to subtract self.img_channel_mean = [103.939, 116.779, 123.68] # scaling the stdev Similarly for self.classifier_regr_std = [8.0, 8.0, 4.0, 4.0]

Reply

Pulkit Sharma says:
November 19, 2018 at 12:28 pm

Hi Hareesh, These are the pre-processing steps which make the training faster. These are used to normalize the data. The values are taken by the people who have created the keras_frcnn library. They must have done experiments using different values and found this set as the best. You can try to change these values as well.

Reply

Asad Rauf Khan says:
November 19, 2018 at 3:42 pm

>>>train_images: Images that we will be using to train the model. We have the classes and the actual bounding boxes for each class in this folder. test_images: Images in this folder will be used to make predictions using the trained model. This set is missing the classes and the bounding boxes for these classes. train.csv: Contains the name, class and bounding box coordinates for each image. There can be multiple rows for one image as a single image can have more than one object. >>> Even tough you have provided the code to convert xml to csv, I'm not sure if I have reliably run it and I have a valid data. Can you please provide the exact data set with the exact train/test ratio so I can get results results identical to yours?

Reply

Pulkit Sharma says:
November 19, 2018 at 3:58 pm

Hi Asad, Once you have the complete csv file, you can divide it randomly in a ratio of say 70:30 (train:test) or 80:20 or any other ratio you want. Then you will have two different csv files, say train.csv and test.csv. Finally, for the images in train.csv, put them in

A Practical Implementation of the Faster R-CNN Algorithm for Object Detection (Part 2 – with Python codes)

Pankaj says:
November 19, 2018 at 6:55 pm

Hi Pulkit, Can you please upload all the data files which you have exactly used for this and share? I am facing issues with splitting the datasets to train/test dataset. Thanks !

Reply

Imran says:
November 20, 2018 at 2:50 am

Bug in Keras 2.2.4, this works until Keras 2.2.2 it seems https://github.com/keras-team/keras/issues/10382

Reply

Pankaj says:
November 24, 2018 at 6:28 pm

Hi Pulkit, The command ran : python test_frcnn.py -p test_images successfully , but did not detect any bounding boxes in the image. Can you please suggest what is wrong ? Thanks, Pankaj

Reply

Otavio Souza says:
November 25, 2018 at 11:54 pm

Hi! Great article! About weights that you make available, you know what the loss, the accuracy, time of the treining and you hardware that you used ? Thank you !

Reply

Otavio Souza says:
November 26, 2018 at 9:22 pm

Hi! Great Article! How I load your weights for training more epochs? Thanks

Reply

Tanuj Misra says:
November 28, 2018 at 2:13 pm

When I am running the code: python3 train_frcnn.py -o simple -p annotate3.txt; following error is coming: Using TensorFlow backend. Traceback (most recent call last): File "train_frcnn.py", line 15, in from keras_frcnn import config, data_generators ModuleNotFoundError: No module named 'keras_frcnn' Also showing 'keras_frcnn' is not available when trying to install/import it explicitly.

Reply

AMM says:
November 29, 2018 at 10:51 am

Hi sir, thank you for this article, pleas can i apply this code on facial component detection?

Reply

Pulkit Sharma says:
November 29, 2018 at 11:42 am

Hi Otavio, The overall loss when I trained the model was around 2.3 The model was trained for more than 2 hours for 1000 epochs and I was using a GPU with 12 cores and 16 GB RAM

Reply

Pulkit Sharma says:
November 29, 2018 at 11:45 am

Hi Otavio, I think you have to retrain the model from the beginning. I am not aware of how to use the model weights and then start training on top of them. I will search for it and will let you know if any solution is found.

Reply

Pulkit Sharma says:
November 29, 2018 at 11:49 am

Hi Tanuj, You are getting this error because it might be possible that you have not cloned the keras_frcnn GitHub repository. You have to clone this repository before running the script.

A Practical Implementation of the Faster R-CNN Algorithm for Object Detection (Part 2 – with Python codes)

Pulkit Sharma says:
November 29, 2018 at 11:55 am

Hi, Yes! You can apply this algorithm for facial component detection. For that you first have to train your model on facial components. Weights used in this model will not help in detecting facial components as this model is trained on different dataset. So, first train your model on facial components and then you can use those weights to detect facial components in new set of images.

Reply

Pulkit Sharma says:
November 29, 2018 at 11:58 am

Hi Pankaj, Make sure you have used the weights provided in the article. Or else you can train your model again and then make predictions for test_images.

Reply

Pulkit Sharma says:
November 29, 2018 at 12:09 pm

Hi Pankaj, Can you please tell me what is the issue that you facing in splitting the data? Once you have the csv file (which you will get after running the code provided in the previous comments) you can randomly pick few images for the training and rest for the test. Please let me know what is the issue that you are facing so that I can help you to overcome it.

Reply

Otavio Souza says:
November 30, 2018 at 4:41 am

I don't understand, I am using a 2 GPU's (Titan X 12G and Geforce GTX 1080 ti) and I took around 4 hours for 20 epochs. Did you change anything in files like config.py or train_frccn,py? I don't know where I am going wrong I would like to ask, if you can post the complete codes that you used for train this model in github or in other plataform, because I'm not knowing how train the model very well Thank you!

Reply

manoj singh says:
December 11, 2018 at 6:34 pm

hi pulkit , i have trained my data but while predicting i am getting images with no bounding boxex, can you tell me the reason??

Reply

Pulkit Sharma says:
December 12, 2018 at 11:54 am

Hi Manoj, Which dataset are you using to train the model? Are the annotations and classes correctly defined? And what is the number of epochs that you have used? Also please share the dataset so that I can look into it and help you in a better way.

Reply

RomDos says:
December 13, 2018 at 8:07 pm

Hi, Thanks for this article it is great ! As I use the model on big images (4000x3000), the training is very long. I am using one Tesla K80 GPU, but I could find a way to train on multiple GPUs. Is there a way to do that ? How should the code be changed ? I have an other question about validation set. A validation set is created in train_frcnn but never I can not see where it is used after in the code. Am I wrong or is there a problem with that part ? Regards

Reply

Pulkit Sharma says:
December 14, 2018 at 11:23 am

Hi, I am not yet aware of how to use multiple GPUs for training the model. I will look for it and will let you know how to do that. Regarding the validation set, they have passed the test images into the validation set as they have true labels and bounding boxes for test images. Whereas, in our case we do not have true labels for test images so we have not passed it in validation set.

Reply

RomDos says:
December 14, 2018 at 3:12 pm

Thanks ! I tried with keras multi_gpu library but didi not get good results.

A Practical Implementation of the Faster R-CNN Algorithm for Object Detection (Part 2 – with Python codes)

December 27, 2018 at 4:05 pm

Hello PULKIT SHARMA, Thank you for great post. Just making a note here you have typo in above code. xmin = int(node.find('bndbox/ymin').text) should be ymin = int(node.find('bndbox/ymin').text). and same for all four. I don't know is there any purpose of this or just a typo. But yeah thank you so much for a great post.

Reply

Jyoti says:

December 27, 2018 at 4:12 pm

Hi Pulkit - thanks for this - great article. I wanted to understand if the training set can be extended. I wanted to train some of the classes from the "Luminoth" library - would that be possible - where can I add the extra class labels? Thanks.

Reply

Pulkit Sharma says:

December 28, 2018 at 11:24 am

Hi Jyoti, You can make changes in the csv file. You can add classes in them and then pass those labels while training the model.

Reply

aaditya says:

January 08, 2019 at 9:57 pm

Hey pulkit, I ran this for epoch length=100 and epoch =5 still i am not getting any output at least some wrong output should be there. dont know the reason why?

Reply

Pulkit Sharma says:

January 09, 2019 at 2:23 pm

Hi Aaditya, Make sure you are giving the correct bounding box coordinates for their corresponding images while training the model. If you give incorrect coordinates, you will not get results. So, I would suggest before passing the coordinates and images for training, first plot some images with their corresponding bounding boxes and check whether the data is in correct form.

Reply

Sajjad says:

January 10, 2019 at 4:58 am

Hi and thanks for the tutorial. I have a problem with bounding box detection in python. I use this tutorial on Stanford cars dataset. http://ai.stanford.edu/~jkrause/cars/car_dataset.html I provide train_annotate.txt and it runs successfully on a few epochs with loss value of 2. it's natural because of the small number of epochs. but when I run test_frcnn.py on the test_images folder it saves pictures with no bounding box. I expected at least a wrong bounding box appear on pictures. would you please help me? thank you. and my other question is, how can i evaluate the accuracy of the trained model on test set? here is the test log for 5 test picture: physical GPU (device: 0, name: GeForce GTX 1060, pci bus id: 0000:01:00.0, compute capability: 6.1) 000001.jpg Elapsed time = 6.0193772315979 [] 000146.jpg Elapsed time = 1.3050158023834229 [] 000150.jpg Elapsed time = 1.2876217365264893 [] 000160.jpg Elapsed time = 0.6831655502319336 [] 000162.jpg Elapsed time = 0.76279616355896 []

Reply

Shivangi says:

January 10, 2019 at 7:43 am

Hi Pulkit, I have two questions to ask 1. During training, we are only training on one image at a time if I understood the code correctly right. So if I have to say train on 2500 training images then to complete one epoch in one go, then I have to set `epoch_length=2500`. 2. Could you please explain the outputs losses `rpn_cls` and `rpn_reg`? I know they are related to RPN output predictions. What I got from the code that we are generating ground truth for RPN using numpy operations and comparing RPN network output and training. But why is it actually required? 3. What exactly is metric bounding box accuracy? Could you explain in more detail. 4. Also for me all the 4 losses are not going down simultaneously, all of them are like really fluctuating. I am using VGG-16, Any suggestions on that

Reply

Md Ashraful Haque says:

January 10, 2019 at 1:08 pm

Hey, Pulkit, thanks for your article. Can you guide me on how to annotate the images with bounding box?? I am confused with this part?? plzz help me.. Thanks

Reply

A Practical Implementation of the Faster R-CNN Algorithm for Object Detection (Part 2 – with Python codes)

January 10, 2019 at 1:16 pm

Thanks.I want to know how to set GPU config.

Reply

Pulkit Sharma says:
January 10, 2019 at 2:55 pm

Hi, Refer to this link.

Reply

Pulkit Sharma says:
January 10, 2019 at 2:58 pm

Hi, You have to manually annotate the images if you are using a new dataset. Or else, you can look for open source datasets which have their corresponding bounding boxes.

Reply

Pulkit Sharma says:
January 10, 2019 at 3:13 pm

Hi Shivangi, 1. 1 epoch is when we have gone through all the images once. So, epoch_length has nothing to do with the training images. 2. The overall loss is calculated based on these losses. Here, one loss is for classification and other for regression. rpn_cls will tell us how accurately we have classified the correct class for the identified object and rpn_reg will tell how accurate our bounding boxes are. 3. It is similar to rmse, not exactly same but conveys the similar message. It tells us how accurate our predicted bounding boxes are. 4. The losses can fluctuate. Ideally they all should be synced but practically when we implement things, there can be fluctuations. Hope this helps!!

Reply

Pulkit Sharma says:
January 10, 2019 at 3:19 pm

Hi Sajjad, Make sure that the training and testing images have similar shape, also you have done similar pre-processing on these images. If the distribution of train and test images is different, you will not get fair results. Also, try to run the model for more number of epochs so that it can learn the patterns and can become better. Regarding your other question, you can evaluate the performance of model on test images only if you have the true labels for the test images, i.e. the class of the objects and their actual bounding boxes. If you do not have these values, you will not be able to evaluate the model on test images.

Reply

Shivangi says:
January 10, 2019 at 5:18 pm

Hi Pulkit, 1. I know what epoch is, but if you look at the code you have used variable `epoch_length`. So my question is different. I want to know if I have to train on 2500 training images then to complete one epoch in one go, then I have to set `epoch_length=2500` because we are only training on one image at a time. 2. If `rpn_clas` and `rpn_reg` are related to classification (say WBC, RBC etc) the what is classifier outputs `detector_cls` and `detector_regr` related to? I think they rpns are not related to classifier output, they are something else. Correct me if I am wrong 3. Still, I can't really make out what it is . Could you explain it a bit more?

Reply

Shivangi says:
January 10, 2019 at 5:57 pm

Could you please un-delete the comment which you deleted about the follow ups on your reply.

Reply

Pulkit Sharma says:
January 10, 2019 at 6:42 pm

Hi Shivangi, 1. I checked the codes and all the images are used in every epoch. Can you please tell in which part of the code you are having doubts? 2. For detail on various losses, please refer to the losses.py which is inside the keras_frcnn folder of the github repository. Link for the repository is given in the article. 3. I checked the codes and found out that this is the accuracy of classes and not the bounding boxes. It can be seen in this code: print('Classifier accuracy for bounding boxes from RPN: {}'.format(class_acc))

Reply

Shivangi says:

A Practical Implementation of the Faster R-CNN Algorithm for Object Detection (Part 2 – with Python codes)

1. Let me frame it differently, what is variable "epoch_length" doing in the code? 2. Yes, "rpn_clas" and "rpn_reg" are NOT related to classification (WBC, RBC). They are related to object presence or not. 3. Got it

Reply

Martin says:
January 11, 2019 at 10:06 am

Hello ! First of all, thanks you for this amazing tuto. I am having just a little problem : Using the weights you provided (my PC is way too slow for getting good ones myself) as well as the option file, when i use test_frcnn.py the labels on WBC and RBC are inverted. The cells themselfs are very-well recognised, it is just that every single WBC is labelled as RBC and every single RBC is labelled as WBC ... Any idea of what may have caused than ?

Reply

Pulkit Sharma says:
January 11, 2019 at 11:43 am

Hi Shivangi, 1. If you look at this line of code: `if iter_num == epoch_length: loss_rpn_cls = np.mean(losses[:, 0]) loss_rpn_regr = np.mean(losses[:, 1]) loss_class_cls = np.mean(losses[:, 2]) loss_class_regr = np.mean(losses[:, 3]) class_acc = np.mean(losses[:, 4])` Suppose you define epoch_number as 1000, so after every 1000 iterations, all the losses will be calculated and the weights will be updated accordingly. In single epoch, we can have multiple iterations. This is also known as Mini Batch Gradient Descent. So, instead of taking single image for training and updating the weights, we take batches of images and train the model on them and update the weights accordingly. Once the model has seen all the images once, one epoch is complete.

Reply

Pulkit Sharma says:
January 11, 2019 at 11:52 am

Hi Martin, Can you please share the screenshot of some of the results that you are getting? That would help me to clarify your doubt in a better way.

Reply

saadiq says:
January 21, 2019 at 3:06 am

Please can you teach me how to use use VOC pascal dataset? i have downloaded the dataset but its not in path/x1,y1,x2,y2,class_name format.

Reply

Karl Magdales says:
January 21, 2019 at 5:56 am

train_frcnn.py -o simple -p annotate.txt File "", line 1 train_frcnn.py -o simple -p annotate.txt ^ SyntaxError: invalid syntax Im running it through IPython console, I dont understand why it isnt working

Reply

Pulkit Sharma says:
January 21, 2019 at 11:46 am

Hi Karl, You have to use the following code: `python train_frcnn.py -o simple -p annotate.txt` Go to the directory where the train_frcnn.py file is and run the above line of code.

Reply

Pulkit Sharma says:
January 21, 2019 at 11:56 am

Hi saadiq, You have to convert the format of the data to use this library. You can do some coding or search online how to convert the VOC pascal dataset. Or else share the format of your data and I will look for solution.

Reply

Sayak Chakraborty says:
January 23, 2019 at 12:01 am

Hello, I am landing up to this error, i have my train_images folder and train.csv file inside keras-frcnn folder and i am trying to use %run from Jupyter, could you please help on this. Parsing annotation files ---------- AttributeError Traceback (most recent call last) ~\Image_Processing\Object_Detection\keras-frcnn\train_frcnn.py in () 77

A Practical Implementation of the Faster R-CNN Algorithm for Object Detection (Part 2 – with Python codes)

filename 39 all_imgs[filename]['width'] = cols AttributeError: 'NoneType' object has no attribute 'shape'

Reply

Pulkit Sharma says:
January 23, 2019 at 11:42 am

Hi Sayak, Open the train_frcnn.py and check the filename that you are giving. Currently it is not able to go to the images folder. Try to edit the filename, i.e. the path of the images.

Reply

Pulkit Sharma says:
January 23, 2019 at 12:51 pm

Hi Mihir, Thanks for pointing it out. That's a good catch and I have updated the code.

Reply

Sayak Chakraborty says:
January 23, 2019 at 9:25 pm

Yes it is solved now by adding full path inside annotation.txt. Thanks. But I am getting this error, please suggest ---------- ---------- InvalidArgumentError Traceback (most recent call last) C:\ProgramData\Anaconda3\lib\site-packages\tensorflow\python\framework\ops.py in _create_c_op(graph, node_def, inputs, control_inputs) 1627 try: -> 1628 c_op = c_api.TF_FinishOperation(op_desc) 1629 except errors.InvalidArgumentError as e: InvalidArgumentError: Shape must be rank 1 but is rank 0 for 'bn_conv1/Reshape_4' (op: 'Reshape') with input shapes: [1,1,1,64], [].

Reply

Leonardo says:
January 23, 2019 at 11:07 pm

Do you have the config.pickle of your model trained? It is needed for testing.

Reply

Pulkit Sharma says:
January 24, 2019 at 12:14 pm

Hi Leonardo, You can download the config.pickle file from this link.

Reply

Pulkit Sharma says:
January 24, 2019 at 12:31 pm

Hi Sayak, You can refer this link and see if you find something helpful.

Reply

saadiq says:
January 24, 2019 at 6:27 pm

Is it possible to generate the test.csv without labelling the images, which will enable you to have the X-coordinates, Y-coordinates and the class of the object?. i thought all the images will be labelled(i.e putting the bboxes and then exporting them in the desired format).

Reply

saadiq says:
January 24, 2019 at 6:49 pm

Is it possible to generate the test.csv without labelling the images, which will enable you to have the X-coordinates, Y-coordinates and the class of the object?. i thought all the images will be labelled(i.e putting the bboxes and then exporting them in the desired format). then, we divide them into train and test.

Reply

Pulkit Sharma says:
January 25, 2019 at 12:16 pm

Hi saadiq, In real life scenarios, we do not have the true labels for the images. Test set is similar to that. We train our model and then predict on test images. You can split the train set given to you into training and validation set. First train your model on training set, get predictions for the validation set, and if you are satisfied with the results, use that model for predicting labels for

A Practical Implementation of the Faster R-CNN Algorithm for Object Detection (Part 2 – with Python codes)

Ravi says:
January 26, 2019 at 9:22 pm

Can we use the same algorithm for detecting text in images? Do you have any resources on end-to-end text recognition from images?

Reply

Chris H says:
January 27, 2019 at 12:51 am

Thank you, Pulkit, for your most excellent and informative post. I've been having troubles getting your XML to CSV code to ouput correctly and I was hoping you might edit the code to include the correct indentations as well as an addition line that writes the parsed data to file.

Reply

Pulkit Sharma says:
January 28, 2019 at 11:51 am

Hi Ravi, I have not yet worked on text recognition project. If I find any relevant resource, I will share it with you.

Reply

Pulkit Sharma says:
January 28, 2019 at 11:55 am

Hi Chris, Can you please tell me what is the error that you are getting?

Reply

Abhijit says:
January 28, 2019 at 12:35 pm

Hi pulkit, First of all thanks for you blog post on object detection, i trained 40 images (my own dataset) on 100 epochs , but when i passed test images it doesn't recognize any of given images means it didn't recognize bounding boxes around images at least wrong prediction is expected but no bounding boxes are detected, i have resized test images in same manner as i did for train images. I have no clue what is happening, so can you please check what is the problem?

Reply

Pulkit Sharma says:
January 28, 2019 at 1:29 pm

Hi Abhijit, It might be possible that the training dataset is small that's why the model is not able to understand the signals. Also, make sure you are providing the correct bounding boxes while training. Try to plot few images with their corresponding bounding boxes to make sure that the notations are correct.

Reply

Abhijit says:
January 28, 2019 at 3:18 pm

Hi Pulkit, i agree that dataset is small but i am expecting wrong prediction at least and i tried by plotting train image where bounding boxes are present with correct notion. Do you want me to share image dataset.

Reply

Abhijit says:
January 28, 2019 at 3:20 pm

Hi Pulkit, i agree that dataset is small but i am expecting wrong prediction at least and i tried by plotting train image where bounding boxes are present with correct notion. Do you want me to share image dataset. test_images prediction
111:mpn:0a0f6d7b3a9f0093d6ce022b6d67f02a.jpeg Elapsed time = 12.53669023513794 []
111:mpn:0a11ac3820dc5591d2f7353eb0e5a966.jpeg Elapsed time = 9.79170274734497 []
111:mpn:0a30eb770fcd3fcb8c590b078825ea50.jpeg Elapsed time = 8.541940212249756 []
111:mpn:0a4a85bd72695d975893fd9f072c127d.jpeg Elapsed time = 7.845632553100586 []
111:mpn:0a4b198321cf36d751b61db66caaba5e.jpeg Elapsed time = 6.625025272369385 []
111:mpn:0a5b89b59272586ff617d915b490ba18.jpeg Elapsed time = 8.454639434814453 []
111:mpn:0a64fc71a5b9de382c964eb929ee0b13.jpeg Elapsed time = 7.680825710296631 []
111:mpn:0a8690b6afc68e32fa59aca5403f0ebd.jpeg Elapsed time = 8.591627359390259 [] why [] is present in prediction?

Reply

A Practical Implementation of the Faster R-CNN Algorithm for Object Detection (Part 2 – with Python codes)

Hi Pulkit, The output does not appear to be correct (I get 8 fields rather than 6 among other apparent output problems): prev_filename filename cell_type xmin xmax ymin ymax 0 BloodImage_00000.jpg 0.jpg WBC 260 491 177 376 1 BloodImage_00000.jpg 0.jpg RBC 78 184 336 435 2 BloodImage_00000.jpg 0.jpg RBC 63 169 237 336 3 BloodImage_00000.jpg 0.jpg RBC 214 320 362 461 4 BloodImage_00000.jpg 0.jpg RBC 414 506 352 445 5 BloodImage_00000.jpg 0.jpg RBC 555 640 356 455 6 BloodImage_00000.jpg 0.jpg RBC 469 567 412 480 7 BloodImage_00000.jpg 0.jpg RBC 1 87 333 437 8 BloodImage_00000.jpg 0.jpg RBC 4 95 406 480 9 BloodImage_00000.jpg 0.jpg RBC 155 247 74 174 10 BloodImage_00000.jpg 0.jpg RBC 11 104 84 162 11 BloodImage_00000.jpg 0.jpg RBC 534 639 39 139 12 BloodImage_00000.jpg 0.jpg RBC 547 640 195 295 13 BloodImage_00000.jpg 0.jpg RBC 388 481 11 111 14 BloodImage_00000.jpg 0.jpg RBC 171 264 175 275 15 BloodImage_00000.jpg 0.jpg RBC 260 374 1 83 16 BloodImage_00000.jpg 0.jpg RBC 229 343 91 174 17 BloodImage_00000.jpg 0.jpg RBC 69 184 144 235 18 BloodImage_00000.jpg 0.jpg RBC 482 594 131 230 19 BloodImage_00000.jpg 0.jpg RBC 368 464 89 176 20 BloodImage_00001.jpg 20.jpg WBC 68 286 315 480 21 BloodImage_00001.jpg 20.jpg RBC 346 446 361 454 22 BloodImage_00001.jpg 20.jpg RBC 53 146 179 299 23 BloodImage_00001.jpg 20.jpg RBC 449 536 400 480 24 BloodImage_00001.jpg 20.jpg RBC 461 548 132 212 25 BloodImage_00001.jpg 20.jpg RBC 454 541 295 375 26 BloodImage_00001.jpg 20.jpg RBC 417 508 283 383 27 BloodImage_00001.jpg 20.jpg RBC 278 369 342 451 28 BloodImage_00001.jpg 20.jpg RBC 545 636 62 159 29 BloodImage_00001.jpg 20.jpg RBC 485 576 91 188 ... ... 4858 BloodImage_00409.jpg 4851.jpg RBC 169 261 79 170 4859 BloodImage_00409.jpg 4851.jpg RBC 2 93 1 83

Reply

Pulkit Sharma says:
January 29, 2019 at 12:32 pm

Hi Abhijit, Please share the files that you are using to train the model, as well as few test images.

Reply

Pulkit Sharma says:
January 29, 2019 at 12:38 pm

Hi Chris, You can remove the column named 'prev_filename'.

Reply

Abhijit says:
January 30, 2019 at 2:35 pm

Hi pulkit, please provide your email id so i can share files.

Reply

Josh says:
January 31, 2019 at 1:27 am

When I got this error it ended up being an issue with the file path being incorrect relative to where you were running the script from. Go into your txt file that has all the file paths and x/y values and make sure the relative path is correct.

Reply

Omkar Halikar says:
January 31, 2019 at 10:11 pm

the approach used here is it yolo v1,v2 ?

Reply

Pulkit Sharma says:
February 01, 2019 at 11:14 am

Hi Omkar, I have implemented Faster R-CNN in this article.

Reply

Sidharth P says:
February 23, 2019 at 6:10 pm

Can we train different object with the same code

Reply

Pulkit Sharma says:
February 25, 2019 at 11:44 am

Hi Sidharth, Yes! you can train this model for different objects as well. You just have to change the annotation file

A Practical Implementation of the Faster R-CNN Algorithm for Object Detection (Part 2 – with Python codes)

Sidharth P says:
February 25, 2019 at 6:32 pm

How can we check the accuracy of the model at last ?

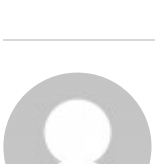
Reply

---

rabiya says:
March 04, 2019 at 11:26 am

kindly guide me the where is train.xml files....?? i cant find them i have train.idx file train.rec file but cant find train.xml file kindly help me

Reply



Pulkit Sharma says:
March 05, 2019 at 1:20 pm

Hi Rabiya, The xml files are in the BCCD - Annotations folder of this repository.

Reply

RISHABH SINHA says:
March 20, 2019 at 2:00 am

Hey pulkit, I trained my own model using the codes for thermal images.I am getting the output but false predictions are also coming.How to resolve it? can you please help me with it

Reply

Pulkit Sharma says:
March 20, 2019 at 1:18 pm

Hi Rishabh, You can try to increase the number of epochs and let the model learn more features. This might help to improve the model performance.

Reply

RISHABH SINHA says:
March 25, 2019 at 11:04 am

Thanks Pulkit, And also i want to ask about the performance metrics,how to calculate mAP?can you share the code here please.

Reply

RISHABH SINHA says:
March 25, 2019 at 11:06 am

Also i trained the model for only 30 epochs as i don't have GPU.so i ill try training it on a better hardware specifications.

Reply

---

rabiya says:
March 28, 2019 at 2:13 am

kindly guide me in the demo code i cant show any image in the last its giving this error Image data cannot be converted to float what show i do?? kindly reply on my 4email rabiyaa.04@gmail.com

Reply

RISHABH SINHA says:
March 28, 2019 at 12:59 pm

hey pulkit, I wanted to ask about what is getting saved in config.pickle file and where are the parameters such a learning rate in the code?

Reply

Urvish Trivedi says:
March 31, 2019 at 10:04 am

import os, sys, random import xml.etree.ElementTree as ET from glob import glob import pandas as pd from shutil import copyfile import pandas as pd annotations = glob('*.xml') df = pd.DataFrame(columns=['prev_filename', 'filename', 'cell_type','xmin', 'xmax', 'ymin', 'ymax']) cnt = 0 for file in annotations: prev_filename = file.split('/')[-1].split('.')[0] + '.jpg' filename = str(cnt) + '.jpg' row = [] parsedXML = ET.parse(file) for node in parsedXML.getroot().iter('object'): blood_cells = node.find('name').text xmin = int(node.find('bndbox/xmin').text) xmax = int(node.find('bndbox/xmax').text) ymin = int(node.find('bndbox/ymin').text) ymax =

A Practical Implementation of the Faster R-CNN Algorithm for Object Detection (Part 2 – with Python codes)

'ymin', 'ymax']) #data[['filename', 'cell_type', 'xmin', 'xmax', 'ymin', 'ymax']].to_csv('blood_cell_detection.csv', index=False) df.to_csv('blood_cell_detection.csv', index=False)

Reply

Arvind says:
March 31, 2019 at 7:45 pm

Hi, Pulkit.. i have 4 images for training, each one consisting of many objects of same class. Then i have 3 images for testing, containing some number of objects of all 4 classes. I want to build this classifier and thought to train Faster RCNN, but facing trouble in preparing Training.csv file and training model further. can you help me with it.

Reply

Pulkit Sharma says:
April 01, 2019 at 11:23 am

Hi Arvind, Since you have only 4 images for training, the model might not be able to learn the features and it will not perform well. Also, can you specify what is the trouble that you are facing in creating the training.csv file?

Reply

Rahul says:
April 10, 2019 at 6:44 pm

Hi Hareesh, Did you manage to get it to run on Keras 2.2.4 ? I am facing the same issue. I have Keras 2.2.4 and Tensorflow 1.13. I can downgrade Keras to 2.2.0, but I am unable to compile Tensorflow 1.8.0 from the source code. So backtracking is not possible for me. How did you solve the issue ?

Reply

Z Knight says:
April 16, 2019 at 5:12 am

I had this same error. I seem to have gotten the program to start learning by editing the keras source. In file keras/backend/tensorflow_backend.py I found 4 reshape functions near each other, one of which was involved in the error. I changed the second argument of each of these from (-1) to [(-1)]. This allowed the program to run. Unfortunately, this is a dangerous change since I don't actually know everything that will be affected.

Reply

RISHABH SINHA says:
April 19, 2019 at 3:08 am

Hey Pulkit, When i am running the code of measure_map.py -o simple -p measure.txt. measure.txt is my annotation file for testing images but it is showing error Traceback (most recent call last): File "measure_map.py", line 271, in t, p = get_map(all_dets, img_data['bboxes'], (fx, fy)) File "measure_map.py", line 66, in get_map if not gt_box['bbox_matched'] and not gt_box['difficult']: KeyError: 'difficult' Can you please help me resolve it?

Reply

RISHABH SINHA says:
April 19, 2019 at 3:11 am

Hey pulkit, I am getting eerror when i am running measure_map.py Traceback (most recent call last): File "measure_map.py", line 271, in t, p = get_map(all_dets, img_data['bboxes'], (fx, fy)) File "measure_map.py", line 66, in get_map if not gt_box['bbox_matched'] and not gt_box['difficult']: KeyError: 'difficult' can you please help me with it

Reply

JQ says:
April 25, 2019 at 7:44 pm

Hi Pulkit ! Thanks for your sharing ! I copy your code and run. But I have a problem that when the "training" starts , it tells me the ETA is 28 HOURS ! .It too long ,when the epoch is 2000 , the training time is a VERY HUGE number.(I use the Tesla T4 GPU).I dont know how long is your training time? Is my setting is wrong ? Or I need to use more faster GPU ? Please tell me what should I do to short the training time. Or could you share the training result to me? Thank you so much!🙏

Reply

Pulkit Sharma says:
April 26, 2019 at 10:14 am

Hi, You can try to reduce the number of epochs to reduce the training time or you can try a faster version of GPU to train your

A Practical Implementation of the Faster R-CNN Algorithm for Object Detection (Part 2 – with Python codes)

Reply

prisilla says:
June 08, 2019 at 4:42 pm
Hi Pulkit, Can we train the above model for tumor detection using bounding boxes? But in a tumor image, we have one or two patches of tumor which is to be detected. The color of tumor varies in each image. How can we do that?
Reply

Haj_reserach says:
June 10, 2019 at 4:30 pm
Thank s for the wonderful article! I tried it and it worked well Now could you please guide me how to run this on AMD GPUs (is it possible to run the same code or some modifications should be needed)? Thanks in Advance!
Reply

SC says:
June 11, 2019 at 2:02 am
Would this object detection algorithm work if the images has objects with a resolution of about 6x6 or 10x10 pixels?
Reply

Elie says:
June 17, 2019 at 12:46 pm
Hi Pulkit , when I run the model in epoch 36/100 I receive the following exception error: "Exception: a must be non-empty". i've run this model on eight pictures (in TIF). the model ran for up to 35/1000 and then started throwing this exception. while, the other day i ran the same model with only 4 pictures (in PNG) and no exceptions where thrown. How is this possible, and would anyone have an idea what the reason would be. Thank you in advance for your help.
Reply

Pulkit Sharma says:
June 18, 2019 at 7:13 pm
Hi Elie, The error that you are getting might be due to memory issues. These are some heavy codes and will require high RAM and GPU power.
Reply

Pulkit Sharma says:
June 18, 2019 at 7:34 pm
Hi, Generally, Faster R-CNN works well enough while dealing with even small objects. But I can not confirm whether it will work well or not as I have not tried it myself for detecting small objects. So, I would suggest that you try it on images having smaller objects and share your insights here which will be helpful for the community.
Reply

Pulkit Sharma says:
June 18, 2019 at 7:37 pm
Hi, Glad you found it helpful. These codes are written to run on both GPU and CPU. I myself trained the model on GPU. So, you can go ahead and train the model on GPU. You won't have to change anything in the codes.
Reply

Pulkit Sharma says:
June 18, 2019 at 7:41 pm
Hi, You can use this for detecting tumor. For that you will require the labeled images first. Once you have the labeled images, i.e. you know the bounding boxes for all the tumor, you can train the model. Once the model is trained, you can use that trained model and make predictions for new images.
Reply

prisilla says:
June 25, 2019 at 5:07 pm
Thanks Pulkit
Reply

A Practical Implementation of the Faster R-CNN Algorithm for Object Detection (Part 2 – with Python codes)

Reply

Pulkit Sharma says:
June 28, 2019 at 4:51 pm

Hi Abisha, The dataset is provided here.

Reply

Leave a Reply

Your email address will not be published. Required fields are marked *

Comment

Name*

Email*

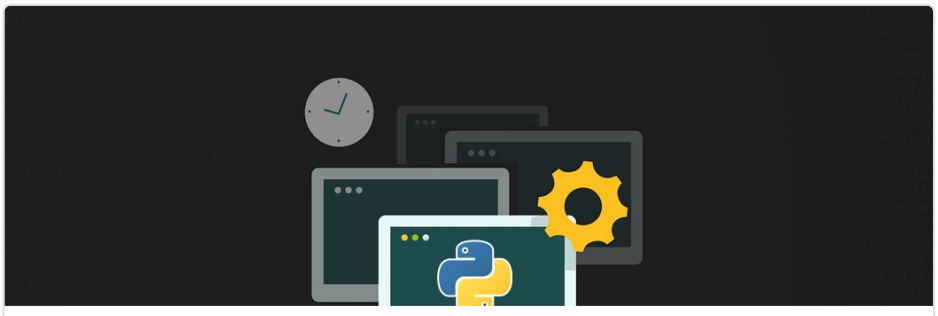
Website

Notify me of follow-up comments by email.

Notify me of new posts by email.

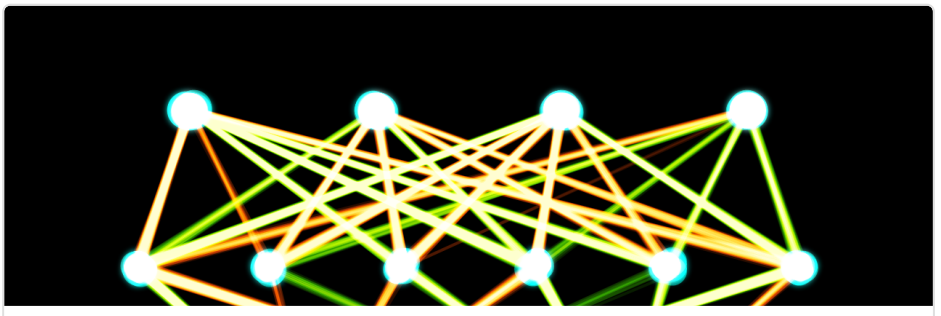
Submit

Top Resources



Python Tutorial: Working with CSV file for Data Science

Harika Bonthu - AUG 21, 2021



Boost Model Accuracy of Imbalanced COVID-19 Mortality Prediction Using GAN-based..

Bala Gangadhar Thilak Adiboina - OCT 07, 2020

Joins in Pandas: Master the Different Types of Joins in..

Understanding Random Forest

A Practical Implementation of the Faster R-CNN Algorithm for Object Detection (Part 2 – with Python codes)

Abhishek Sharma - FEB 27, 2020

Sruthi E R - JUN 17, 2021