Copy_of_HW4

November 4, 2022

0.1 Initializing and imports

[2]:
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

[3]:
```
from __future__ import print_function, division
!pip install pytorch_lightning &> /dev/null


from torchvision.utils import draw_bounding_boxes
import os
import copy
import torch
import pandas as pd
from skimage import io, transform
import numpy as np
import matplotlib.pyplot as plt
from torch.utils.data import Dataset, DataLoader, random_split, TensorDataset
from torchvision import transforms, utils
from  matplotlib.patches import Rectangle as rec
import numpy as np
import h5py
import cv2
import matplotlib.pyplot as plt
import torch
import torchvision
import torchvision.transforms as transforms
from PIL import Image
from scipy import ndimage
import torch.nn.functional as F

import torch.nn as nn
import pytorch_lightning as pl
import pytorch_lightning.loggers as pl_loggers
import pytorch_lightning.callbacks as pl_callbacks
```

```
# CUDA for PyTorch
use_cuda = torch.cuda.is_available()
device = torch.device("cuda:0" if use_cuda else "cpu")
seed = 17
torch.manual_seed(seed);
```

[60]:
```
from dataset_v1 import *
from utils import *
from rpn_v1 import *

from multiprocessing.connection import wait
import torch
from torch.nn import functional as F
from torchvision import transforms
from torch import nn, Tensor
import matplotlib.pyplot as plt
import torchvision
from scipy import stats as st


from torch.utils.data import Dataset, DataLoader
import h5py
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.patches as patches
```

0.2 Defining file path and building Dataset

[8]:
```
# file path and make a list
imgs_path   = '/content/drive/MyDrive/UPENN/SEM 3/CIS680/HW3/
 ↪hw3_mycocodata_img_comp_zlib.h5'
masks_path  = '/content/drive/MyDrive/UPENN/SEM 3/CIS680/HW3/
 ↪hw3_mycocodata_mask_comp_zlib.h5'
labels_path = '/content/drive/MyDrive/UPENN/SEM 3/CIS680/HW3/
 ↪hw3_mycocodata_labels_comp_zlib.npy'
bboxes_path = '/content/drive/MyDrive/UPENN/SEM 3/CIS680/HW3/
 ↪hw3_mycocodata_bboxes_comp_zlib.npy'
paths = [imgs_path, masks_path, labels_path, bboxes_path]
# load the data into data.Dataset
dataset = BuildDataset(paths)

# build the dataloader
# set 20% of the dataset as the training data
full_size = len(dataset)
```

```
train_size = int(full_size * 0.8)
test_size = full_size - train_size
# random split the dataset into training and testset

train_dataset, test_dataset = torch.utils.data.random_split(dataset,␣
↪[train_size, test_size])
rpn_net = RPNHead()
# push the randomized training data into the dataloader

# train_loader = DataLoader(train_dataset, batch_size=2, shuffle=True,␣
↪num_workers=0)
# test_loader  = DataLoader(test_dataset, batch_size=2, shuffle=False,␣
↪num_workers=0)
batch_size = 4
train_build_loader = BuildDataLoader(train_dataset, batch_size=batch_size,␣
↪shuffle=True, num_workers=0)
train_loader = train_build_loader.loader()
test_build_loader = BuildDataLoader(test_dataset, batch_size=batch_size,␣
↪shuffle=False, num_workers=0)
test_loader = test_build_loader.loader()
```

0.3 Training the Model

[10]:
```
from pytorch_lightning.callbacks import ModelCheckpoint
val_checkpoint_callback = ModelCheckpoint(
    monitor="val_loss",
    dirpath="./training_data_new_model_2",
    filename="val_loss{epoch:02d}-{val_loss:.2f}",
    save_top_k=3,
    mode="min",
)
train_checkpoint_callback = ModelCheckpoint(
    monitor="train_loss",
    dirpath="./training_data_new_model_2",
    filename="train_loss{epoch:02d}-{train_loss:.2f}",
    save_top_k=3,
    mode="min",
)
# model =
tb_logger = pl_loggers.TensorBoardLogger("logs2/")
trainer = pl.Trainer(gpus=1,logger=tb_logger,␣
↪max_epochs=40,callbacks=[val_checkpoint_callback,train_checkpoint_callback])
trainer.fit(rpn_net,train_loader,test_loader)
```

```
/usr/local/lib/python3.7/dist-
packages/pytorch_lightning/trainer/connectors/accelerator_connector.py:447:
```

```
LightningDeprecationWarning: Setting `Trainer(gpus=1)` is deprecated in v1.7 and
will be removed in v2.0. Please use `Trainer(accelerator='gpu', devices=1)`
instead.
  f"Setting `Trainer(gpus={gpus!r})` is deprecated in v1.7 and will be removed"
INFO:pytorch_lightning.utilities.rank_zero:GPU available: True (cuda), used:
True
INFO:pytorch_lightning.utilities.rank_zero:TPU available: False, using: 0 TPU
cores
INFO:pytorch_lightning.utilities.rank_zero:IPU available: False, using: 0 IPUs
INFO:pytorch_lightning.utilities.rank_zero:HPU available: False, using: 0 HPUs
WARNING:pytorch_lightning.loggers.tensorboard:Missing logger folder:
logs2/lightning_logs
INFO:pytorch_lightning.accelerators.cuda:LOCAL_RANK: 0 - CUDA_VISIBLE_DEVICES:
[0]
INFO:pytorch_lightning.callbacks.model_summary:
  | Name                      | Type       | Params
-------------------------------------------------------
0 | backbone                  | Sequential | 1.1 M
1 | intermediate_layer        | Sequential | 590 K
2 | proposal_classifier_head | Sequential | 257
3 | proposal_regressor_head  | Sequential | 1.0 K
-------------------------------------------------------
1.7 M     Trainable params
0         Non-trainable params
1.7 M     Total params
6.730     Total estimated model params size (MB)

Sanity Checking: 0it [00:00, ?it/s]


/usr/local/lib/python3.7/dist-
packages/pytorch_lightning/trainer/connectors/data_connector.py:229:
PossibleUserWarning: The dataloader, val_dataloader 0, does not have many
workers which may be a bottleneck. Consider increasing the value of the
`num_workers` argument` (try 12 which is the number of cpus on this machine) in
the `DataLoader` init to improve performance.
  category=PossibleUserWarning,
/content/rpn_3.py:323: UserWarning: __floordiv__ is deprecated, and its behavior
will change in a future version of pytorch. It currently rounds toward 0 (like
the 'trunc' function NOT 'floor'). This results in incorrect rounding for
negative values. To keep the current behavior, use torch.div(a, b,
rounding_mode='trunc'), or for actual floor division, use torch.div(a, b,
rounding_mode='floor').
  row          = invalid // 68
/content/rpn_3.py:330: UserWarning: __floordiv__ is deprecated, and its behavior
will change in a future version of pytorch. It currently rounds toward 0 (like
the 'trunc' function NOT 'floor'). This results in incorrect rounding for
negative values. To keep the current behavior, use torch.div(a, b,
rounding_mode='trunc'), or for actual floor division, use torch.div(a, b,
```

```
rounding_mode='floor').
  row_anc          = valid_anchor_idx // 68
/usr/local/lib/python3.7/dist-
packages/pytorch_lightning/trainer/connectors/data_connector.py:229:
PossibleUserWarning: The dataloader, train_dataloader, does not have many
workers which may be a bottleneck. Consider increasing the value of the
`num_workers` argument` (try 12 which is the number of cpus on this machine) in
the `DataLoader` init to improve performance.
  category=PossibleUserWarning,
Training: 0it [00:00, ?it/s]


Validation: 0it [00:00, ?it/s]


Validation: 0it [00:00, ?it/s]


Validation: 0it [00:00, ?it/s]


Validation: 0it [00:00, ?it/s]


Validation: 0it [00:00, ?it/s]


Validation: 0it [00:00, ?it/s]


Validation: 0it [00:00, ?it/s]


Validation: 0it [00:00, ?it/s]


Validation: 0it [00:00, ?it/s]


Validation: 0it [00:00, ?it/s]


Validation: 0it [00:00, ?it/s]


Validation: 0it [00:00, ?it/s]


Validation: 0it [00:00, ?it/s]


Validation: 0it [00:00, ?it/s]
```

```
Validation: 0it [00:00, ?it/s]

Validation: 0it [00:00, ?it/s]

Validation: 0it [00:00, ?it/s]

Validation: 0it [00:00, ?it/s]

Validation: 0it [00:00, ?it/s]

Validation: 0it [00:00, ?it/s]

Validation: 0it [00:00, ?it/s]

Validation: 0it [00:00, ?it/s]

Validation: 0it [00:00, ?it/s]

Validation: 0it [00:00, ?it/s]

Validation: 0it [00:00, ?it/s]

Validation: 0it [00:00, ?it/s]

Validation: 0it [00:00, ?it/s]

Validation: 0it [00:00, ?it/s]

Validation: 0it [00:00, ?it/s]

Validation: 0it [00:00, ?it/s]

Validation: 0it [00:00, ?it/s]

Validation: 0it [00:00, ?it/s]

Validation: 0it [00:00, ?it/s]

Validation: 0it [00:00, ?it/s]
```

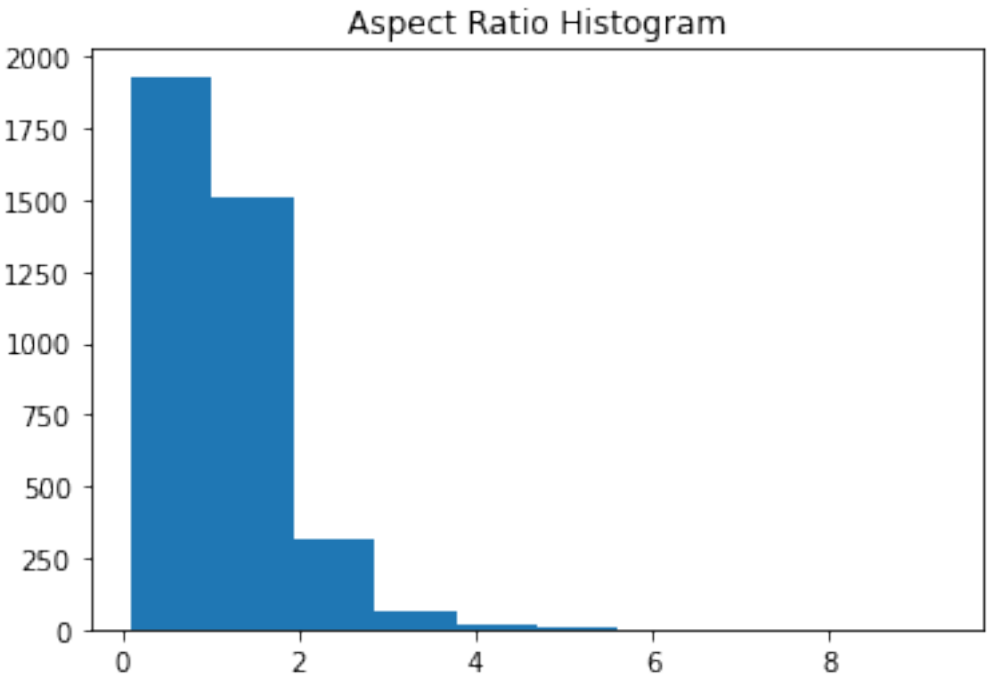
```
/usr/local/lib/python3.7/dist-packages/pytorch_lightning/trainer/call.py:48:
UserWarning: Detected KeyboardInterrupt, attempting graceful shutdown…
  rank_zero_warn("Detected KeyboardInterrupt, attempting graceful shutdown…")
```

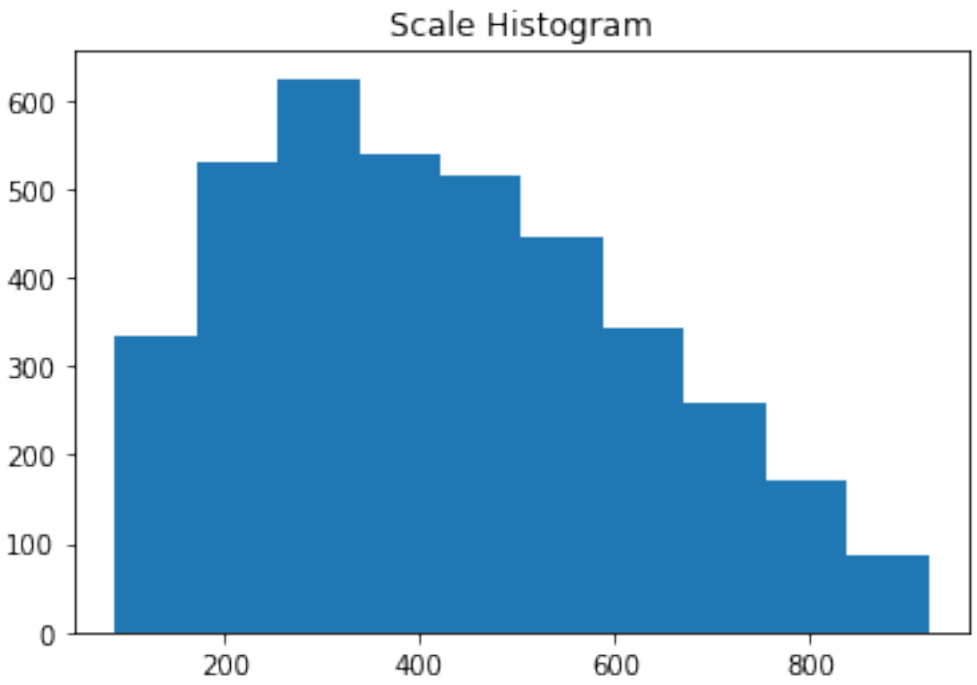
[ ]:
```
%load_ext tensorboard
%tensorboard --logdir logs2/lightning_logs/ # TODO1 : what is the x axis here ?␣
↪in the tensorboard
```

[11]:
```
torch.save(rpn_net.state_dict(), 'rpn_2.pth')
np.save('train_losses_rpn2.npy', rpn_net.train_losses)
np.save('val_losses_rpn2.npy', rpn_net.val_losses)
```

0.4 Historgram Plot

[61]:
```
histogram(bboxes_path)
```





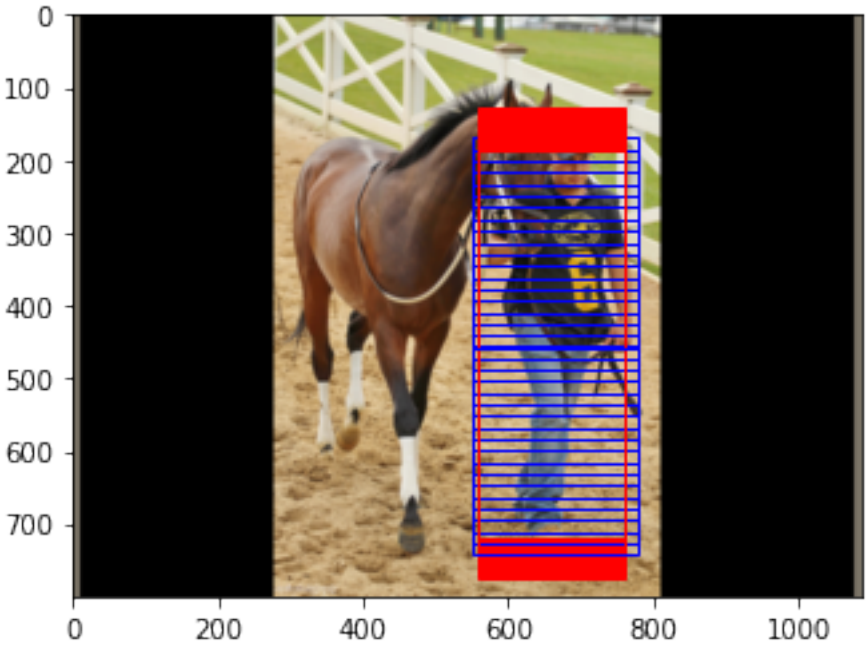
[61]:
```
([425.4656465848386, 1.1554366027804388],
 [404.3293045408205, 1.0111759884745717],
 [792.8099992678804, 1.0006253908692935])
```

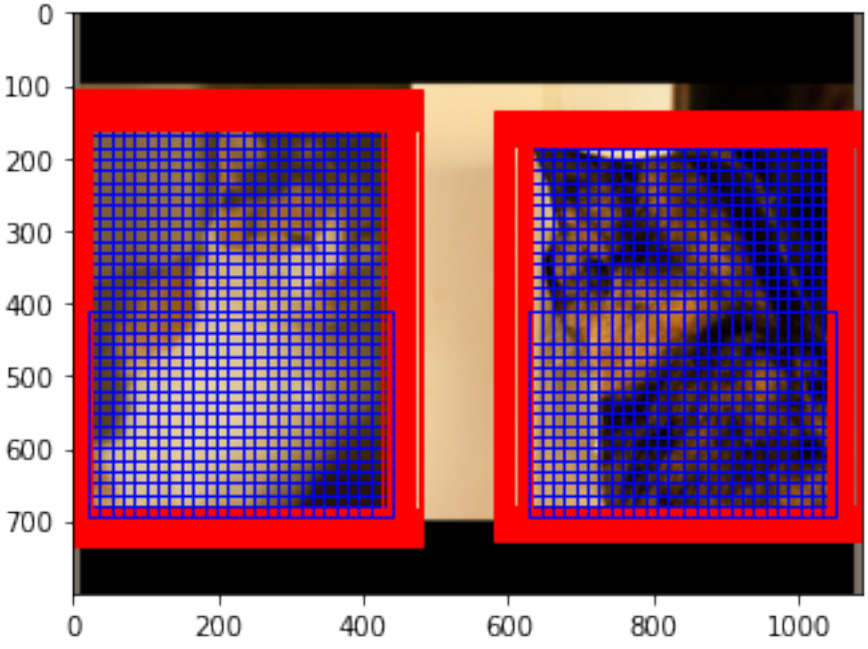
We chose the mode values from the scale and aspect ratio being 256 and 0.8 respectively

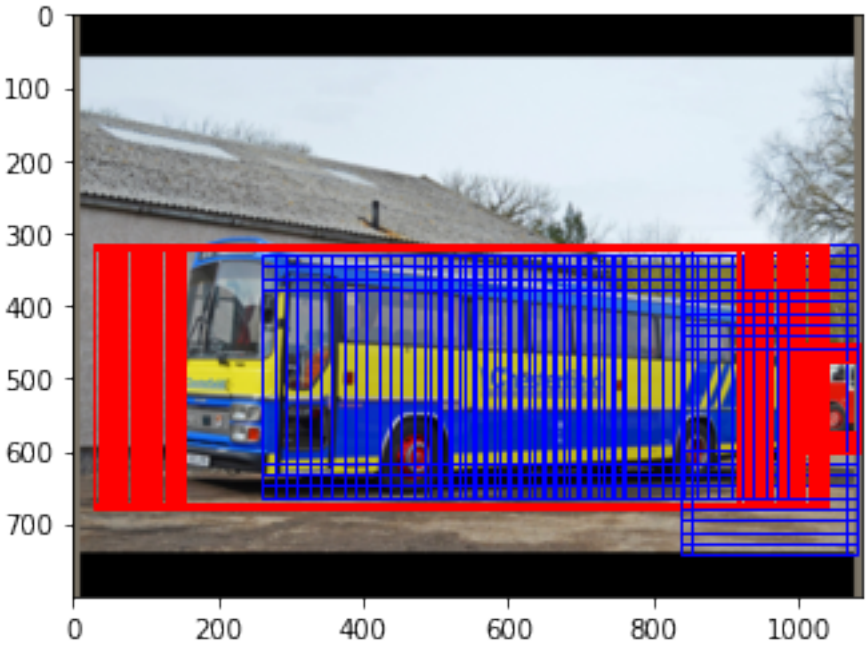
0.5 Images that show the positive anchors and the corresponding ground truth boxes

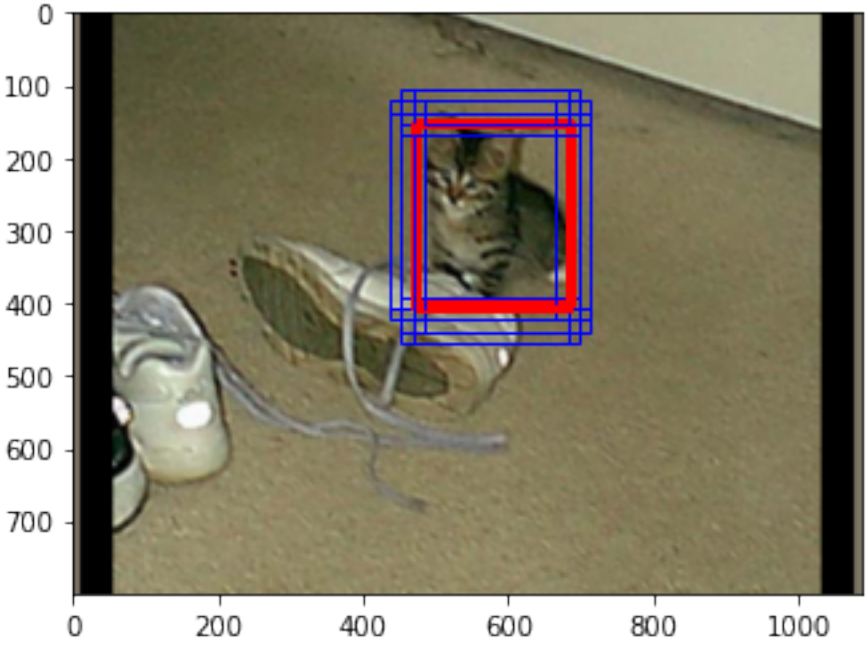
[52]:
```
def positive_anchors_gt_boxes():
  batch_size = 1
  train_build_loader = BuildDataLoader(train_dataset, batch_size=batch_size,␣
↪shuffle=True, num_workers=0)
  train_loader = train_build_loader.loader()
  test_build_loader = BuildDataLoader(test_dataset, batch_size=batch_size,␣
↪shuffle=False, num_workers=0)
  test_loader = test_build_loader.loader()

  for i,batch in enumerate(train_loader,0):
    for idx in range(len(images)):
        images = batch['images']
        images = torch.stack(images[:])
        indexes= batch['index']
        boxes  = batch['bbox']
```

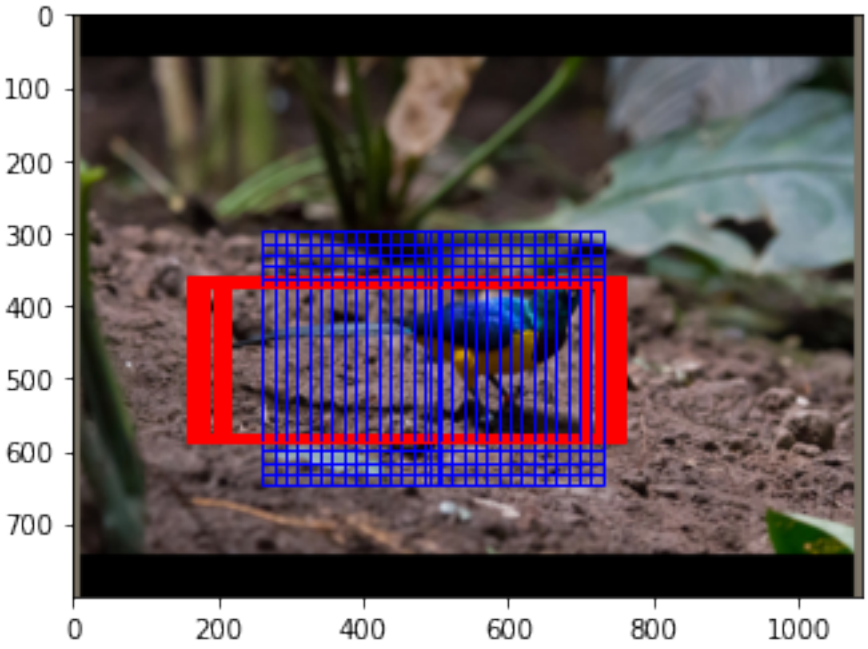
```
        # anchors = rpn_net.create_anchors(rpn_net.anchors_param["ratio"],␣
↪rpn_net.anchors_param["scale"], rpn_net.anchors_param["grid_size"], rpn_net.
↪anchors_param["stride"])
        gt,ground_coord = rpn_net.create_batch_truth(boxes, indexes, (800,1088))

        # Flatten the ground truth and the anchors
        flatten_coord,flatten_gt,flatten_anchors=output_flattening(ground_coord.
↪unsqueeze(0),gt.unsqueeze(0),rpn_net.anchors)

        # Decode the ground truth box to get the upper left and lower right␣
↪corners of the ground truth boxes
        decoded_coord=output_decoding(flatten_coord,flatten_anchors)

        # Plot the image and the anchor boxes with the positive labels and␣
↪their corresponding ground truth box
        images = transforms.functional.normalize(images,
                                                 [-0.485/0.229, -0.456/0.
↪224, -0.406/0.225],
                                                 [1/0.229, 1/0.224, 1/0.
↪225], inplace=False)
        fig,ax=plt.subplots(1,1)
        ax.imshow(images[0].permute(1,2,0))

        find_cor=(flatten_gt==1).nonzero()
        find_neg=(flatten_gt==-1).nonzero()

        for elem in find_cor:
            coord=decoded_coord[elem,:].view(-1)
            anchor=flatten_anchors[elem,:].view(-1)

            col='r'
            rect=patches.
↪Rectangle((coord[0],coord[1]),coord[2]-coord[0],coord[3]-coord[1],fill=False,color=col)
            ax.add_patch(rect)
            rect=patches.Rectangle((anchor[0]-anchor[2]/2,anchor[1]-anchor[3]/
↪2),anchor[2],anchor[3],fill=False,color='b')
            ax.add_patch(rect)

        plt.show()

        if(i < 10):
            break
```

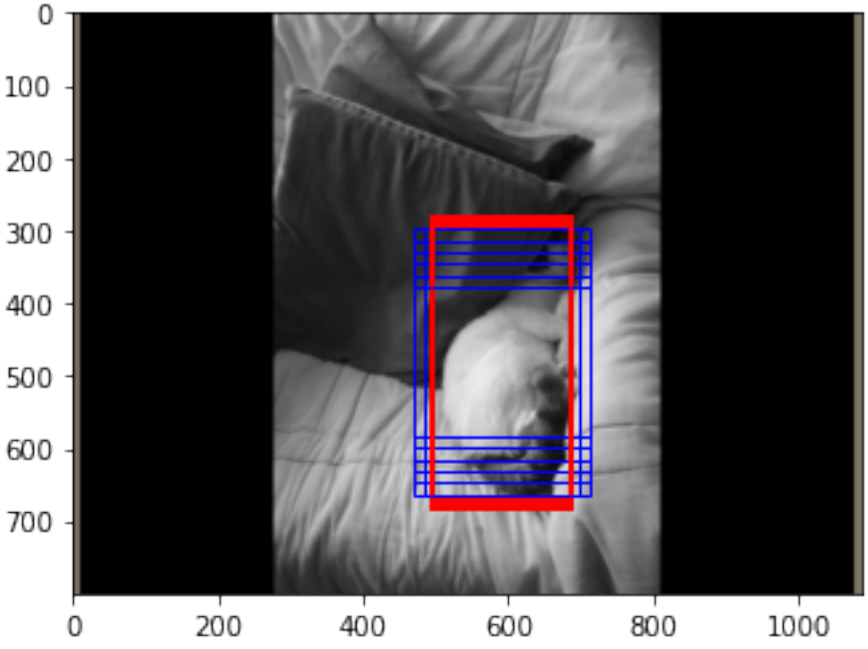

0
100
200
300
400
500
600
700
0
200
400
600
800
1000

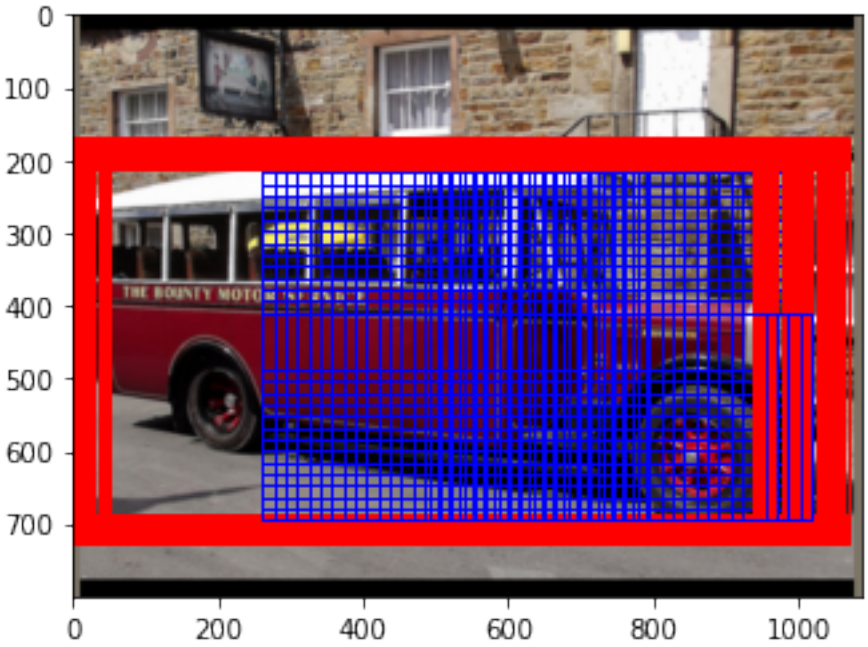

0
100
200
300
400
500
600
700
0
200
400
600
800
1000

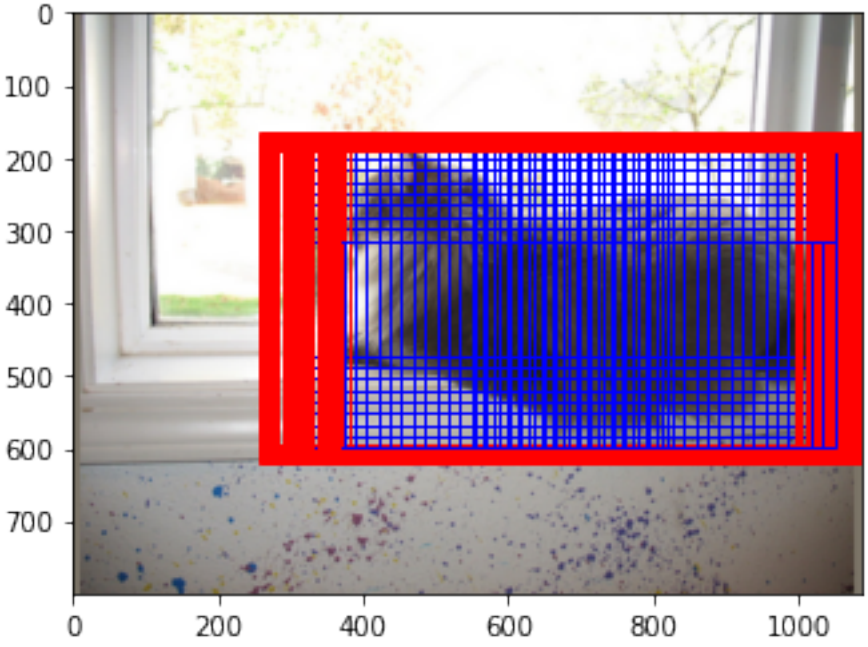

0
100
200
300
400
500
600
700
0
200
400
600
800
1000

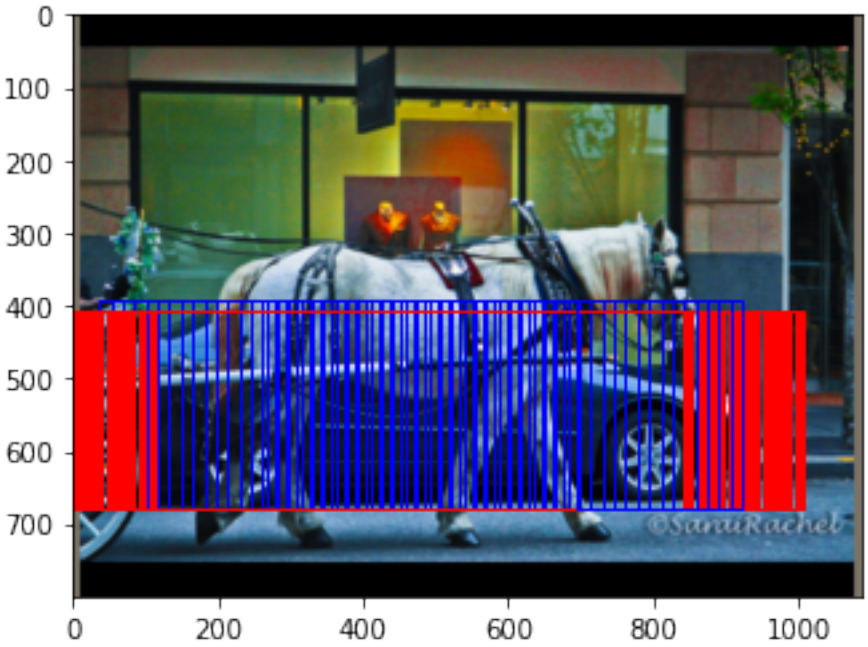

0
100
200
300
400
500
600
700
0
200
400
600
800
1000

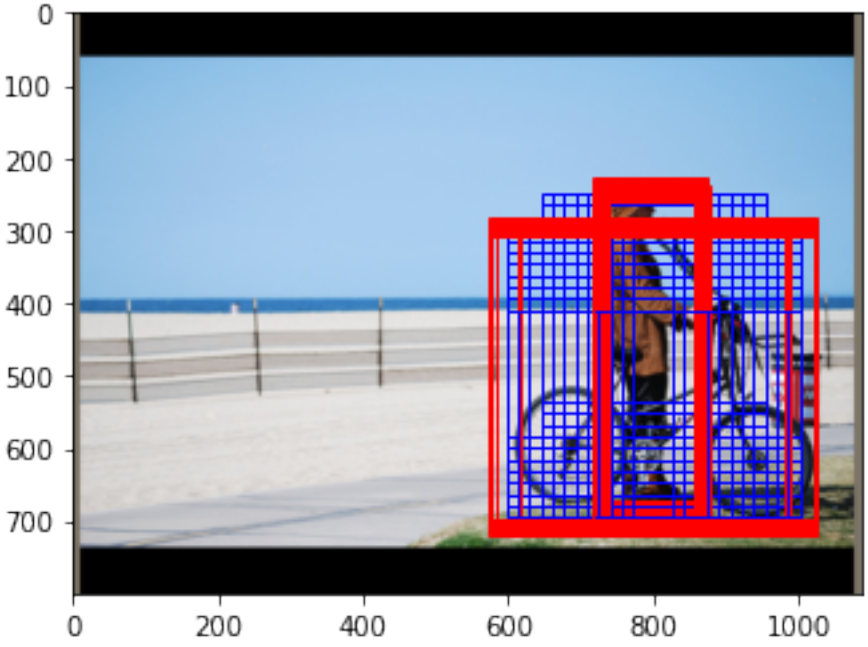

0
100
200
300
400
500
600
700
0
200
400
600
800
1000

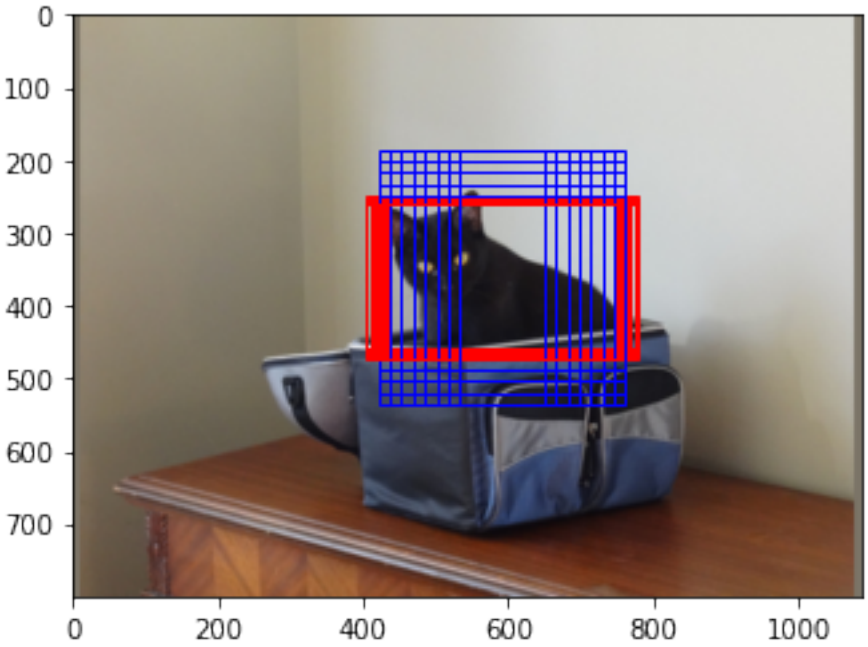

0
100
200
300
400
500
600
700
0
200
400
600
800
1000

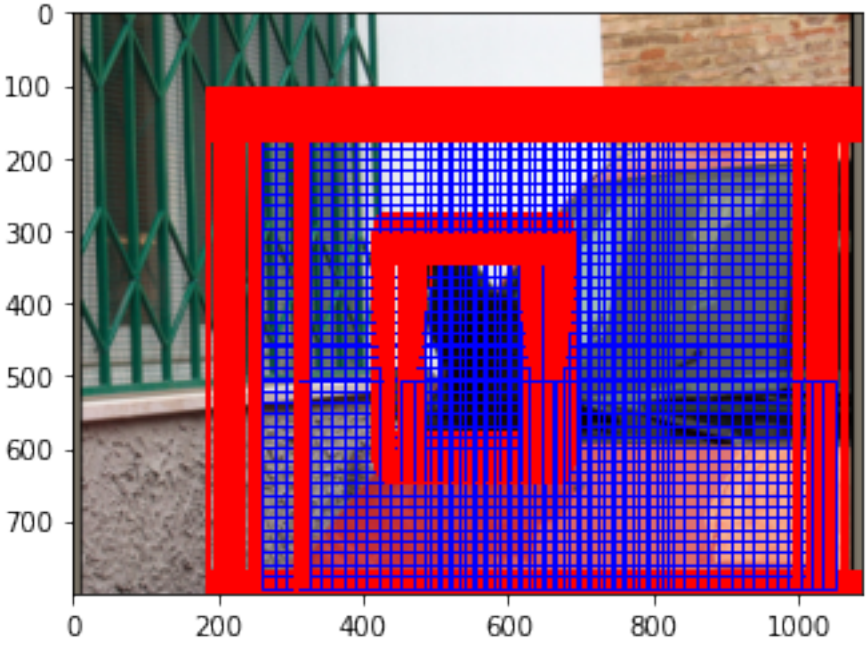

0
100
200
300
400
500
600
700
0
200
400
600
800
1000

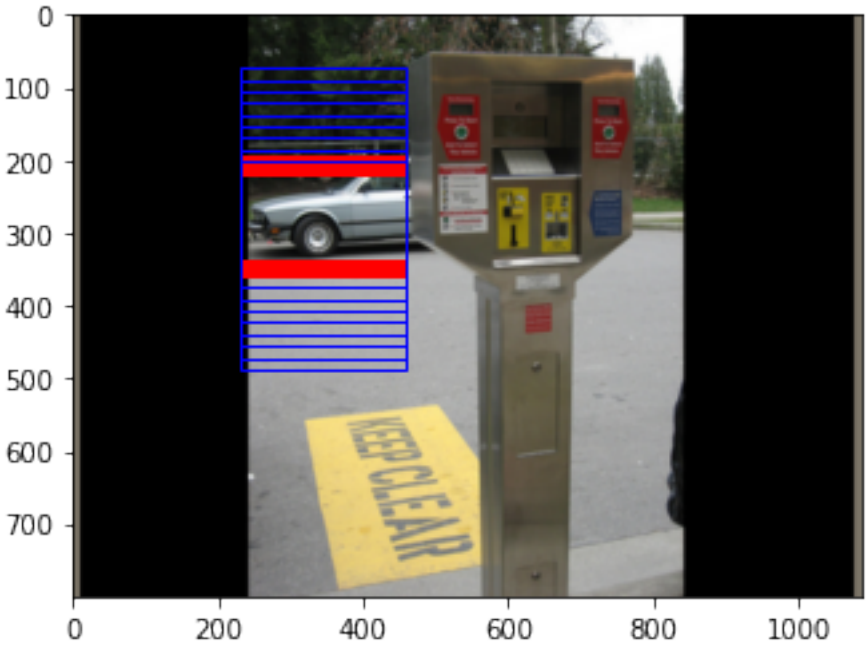

0
100
200
300
400
500
600
700
0
200
400
600
800
1000

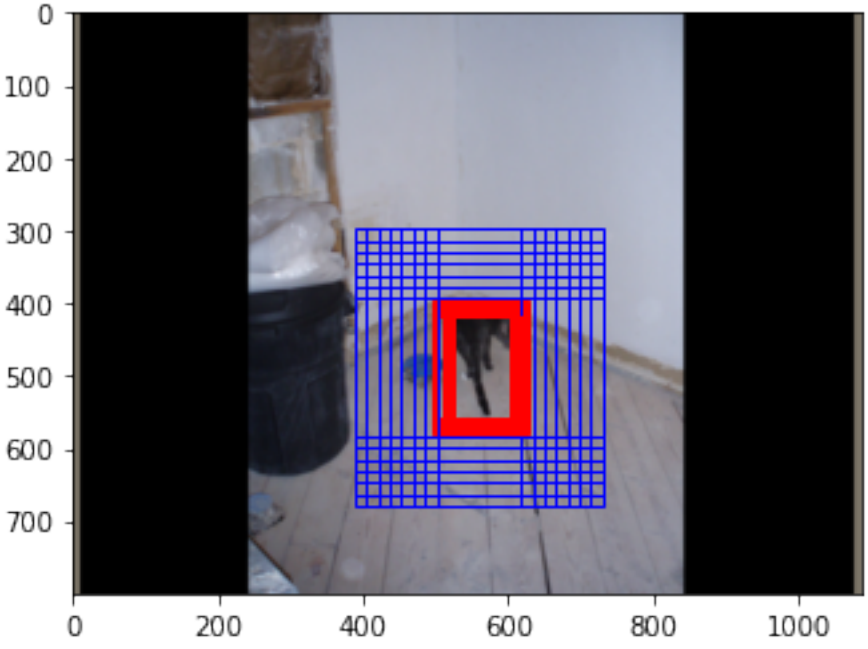

0
100
200
300
400
500
600
700
0
200
400
600
800
1000
©SanuRachel

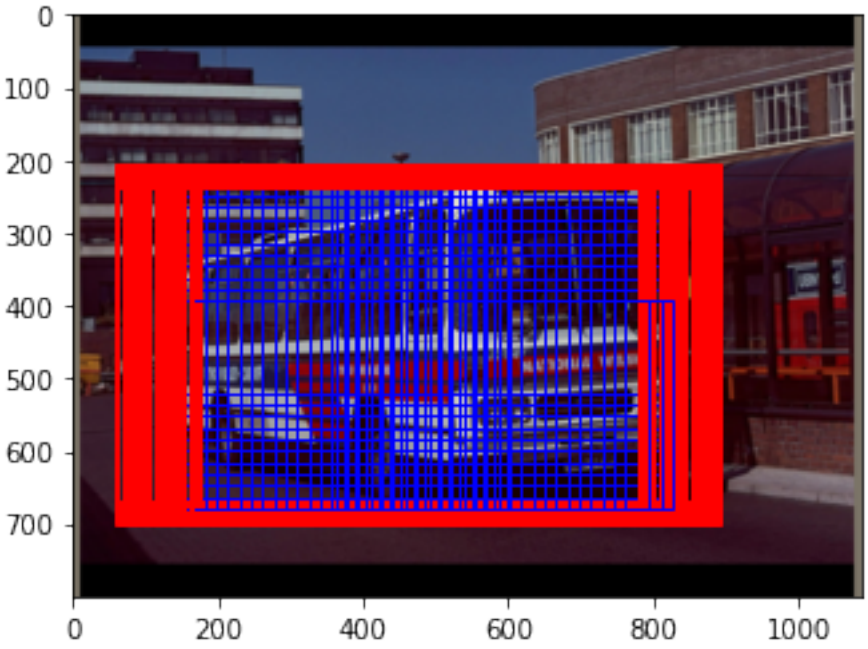

0
100
200
300
400
500
600
700
0
200
400
600
800
1000

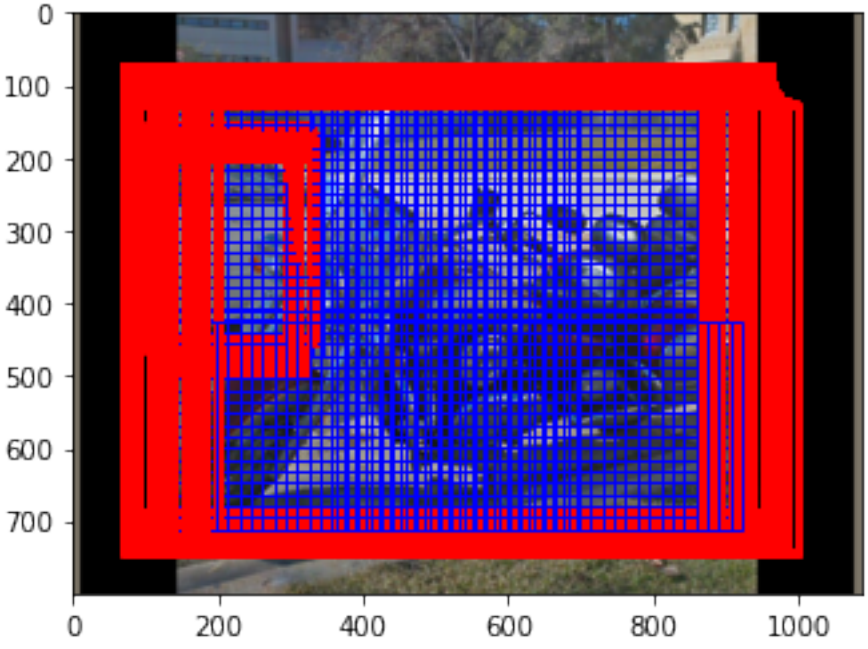

0
100
200
300
400
500
600
700
0
200
400
600
800
1000

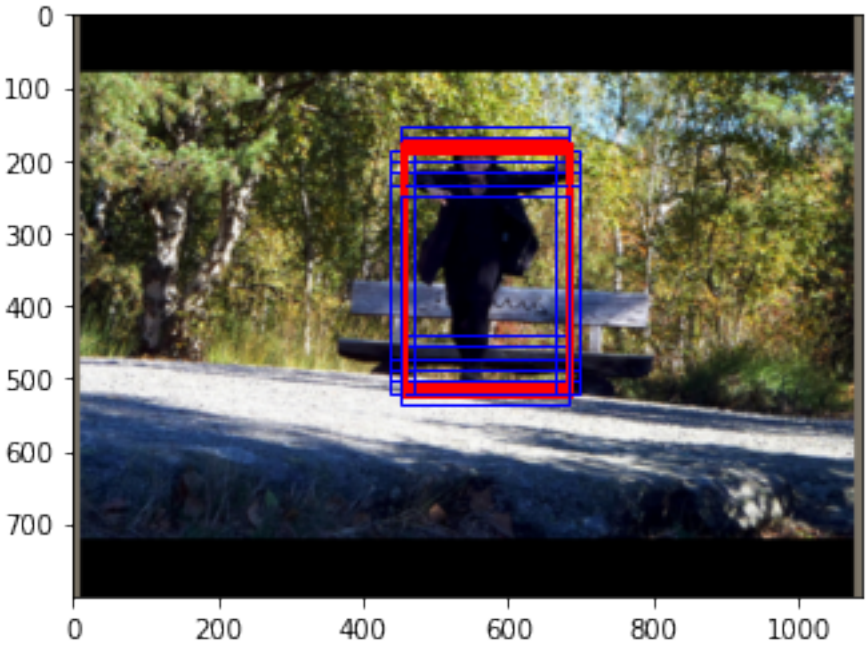

0
100
200
300
400
500
600
700
0
200
400
600
800
1000

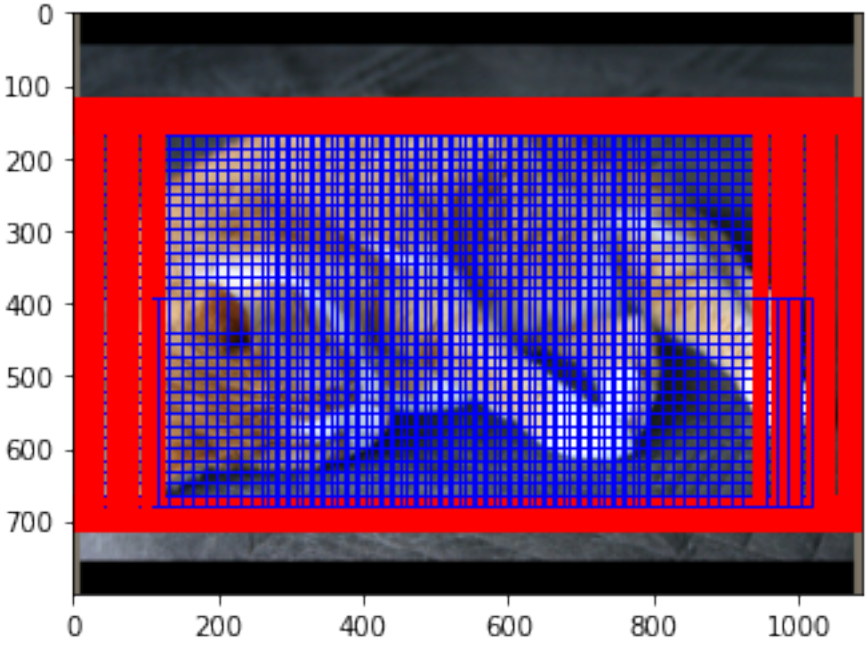

0
100
200
300
400
500
600
700
0
200
400
600
800
1000
KEEP CLEAR

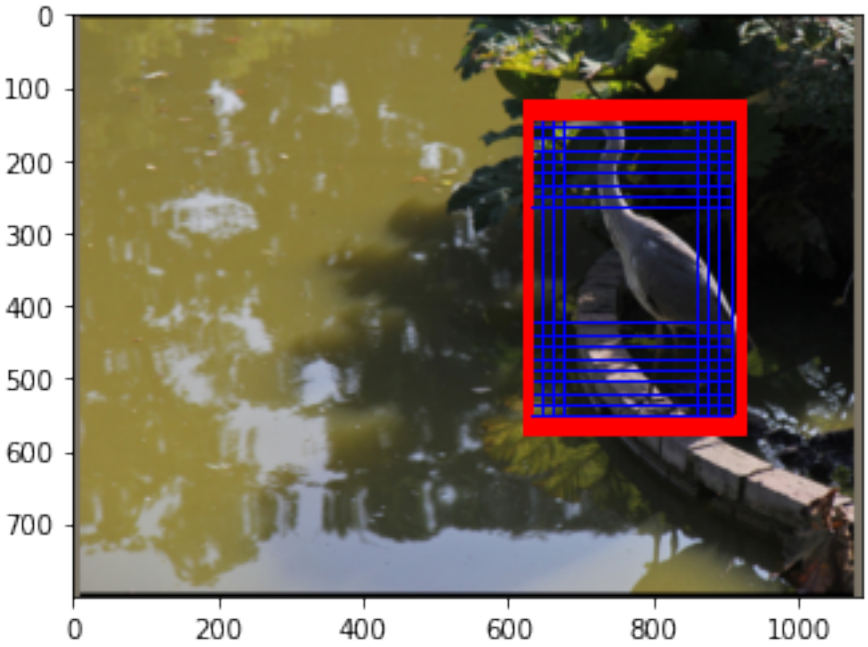

0
100
200
300
400
500
600
700
0
200
400
600
800
1000

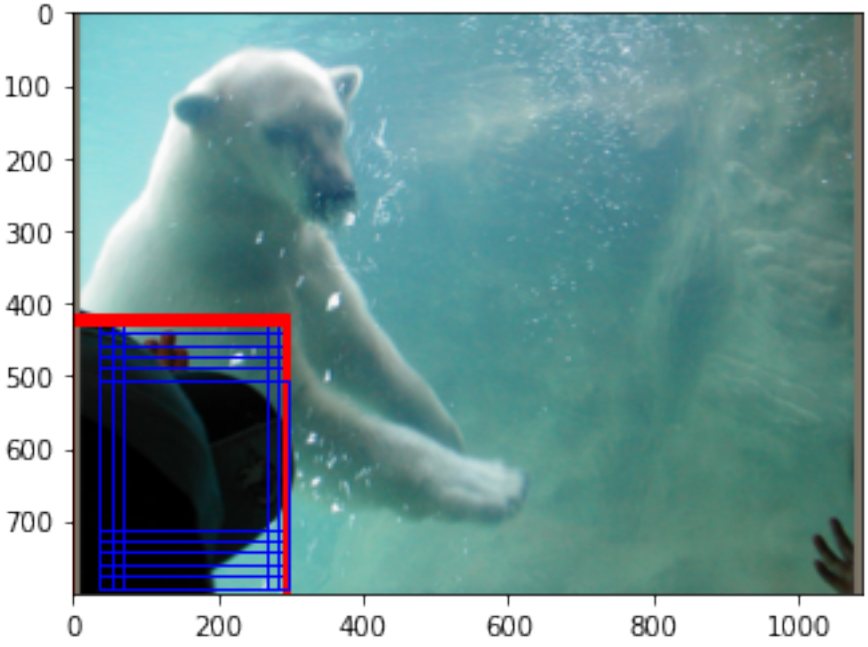

0
100
200
300
400
500
600
700
0
200
400
600
800
1000

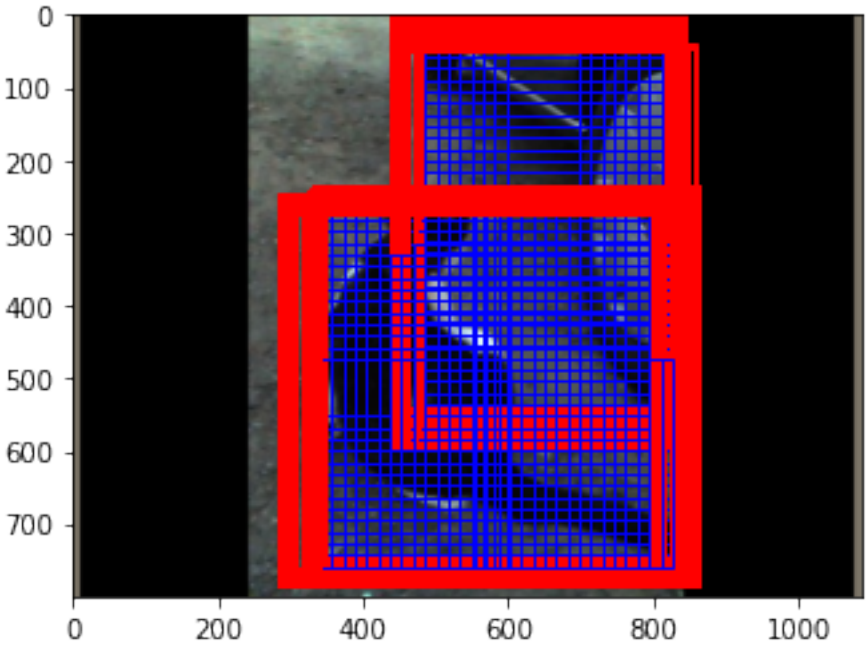

0
100
200
300
400
500
600
700
0
200
400
600
800
1000

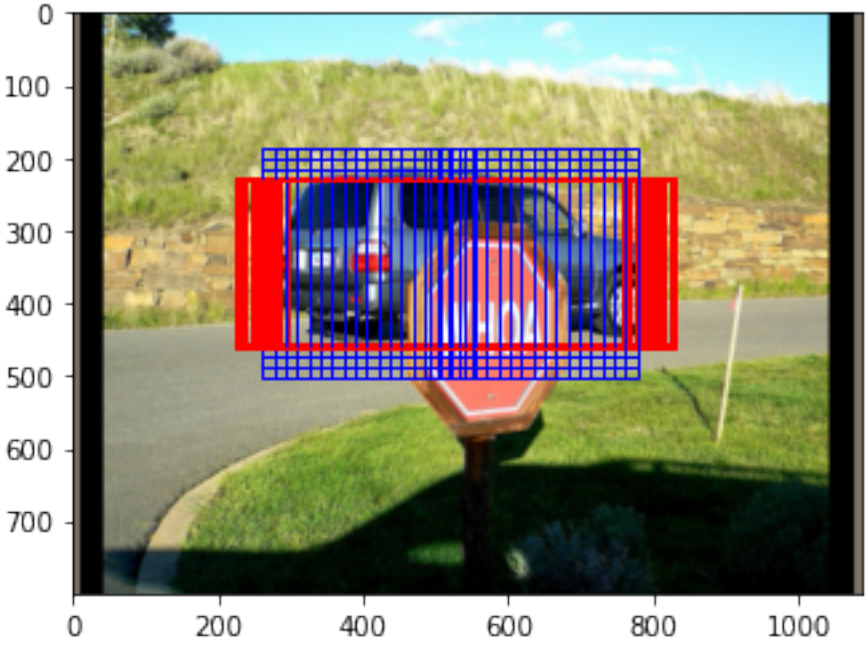

0
100
200
300
400
500
600
700
0
200
400
600
800
1000

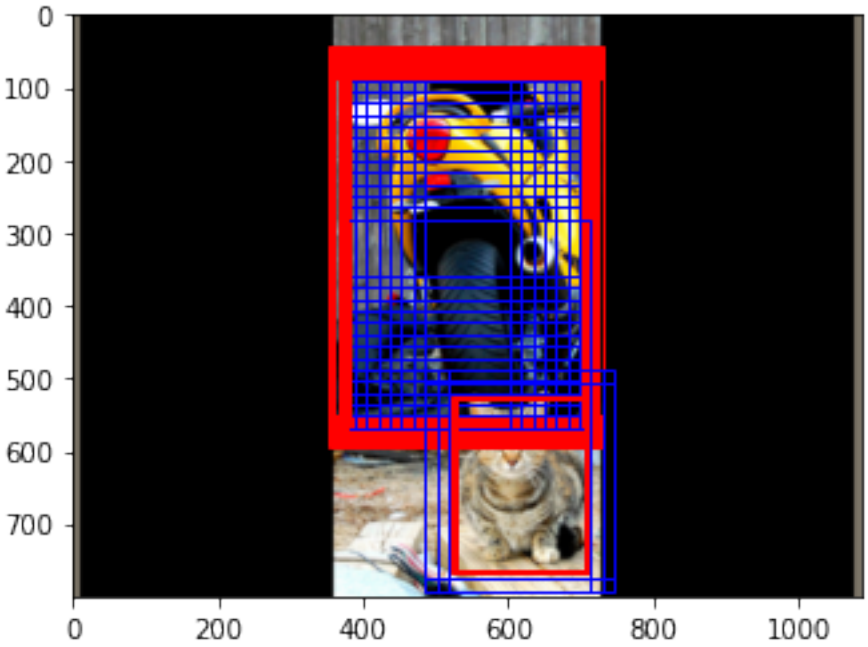

0
100
200
300
400
500
600
700
0
200
400
600
800
1000

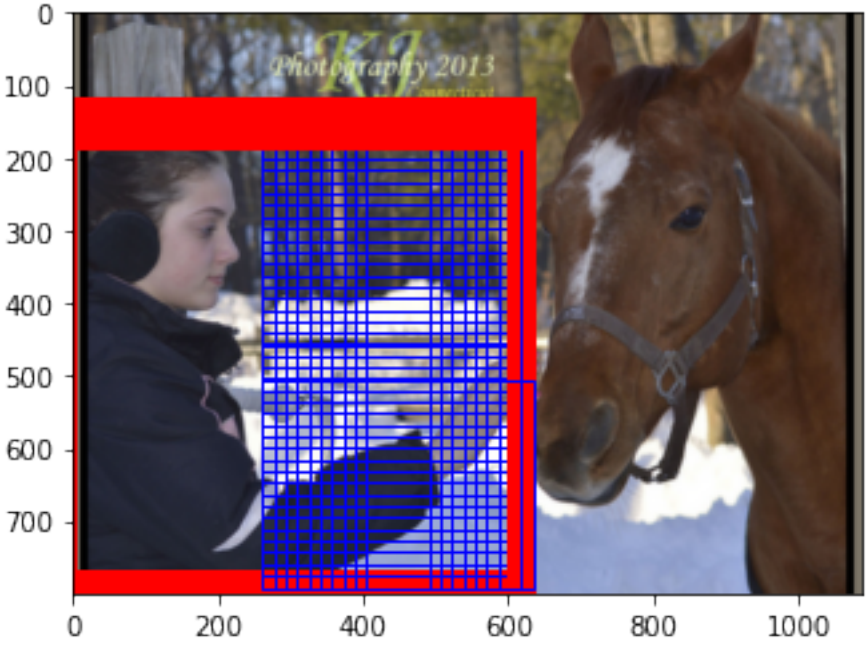

0
100
200
300
400
500
600
700
0
200
400
600
800
1000

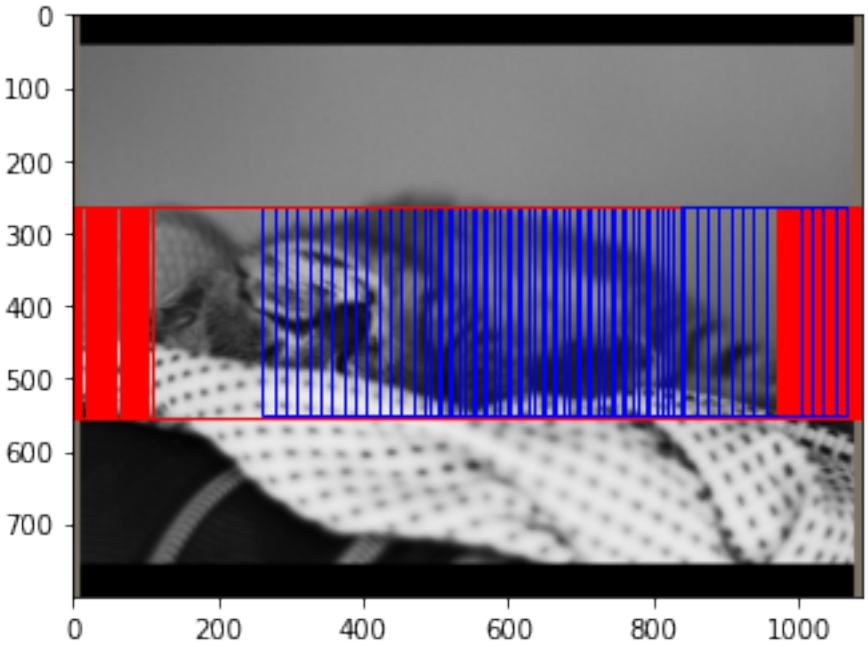

0
100
200
300
400
500
600
700
0
200
400
600
800
1000

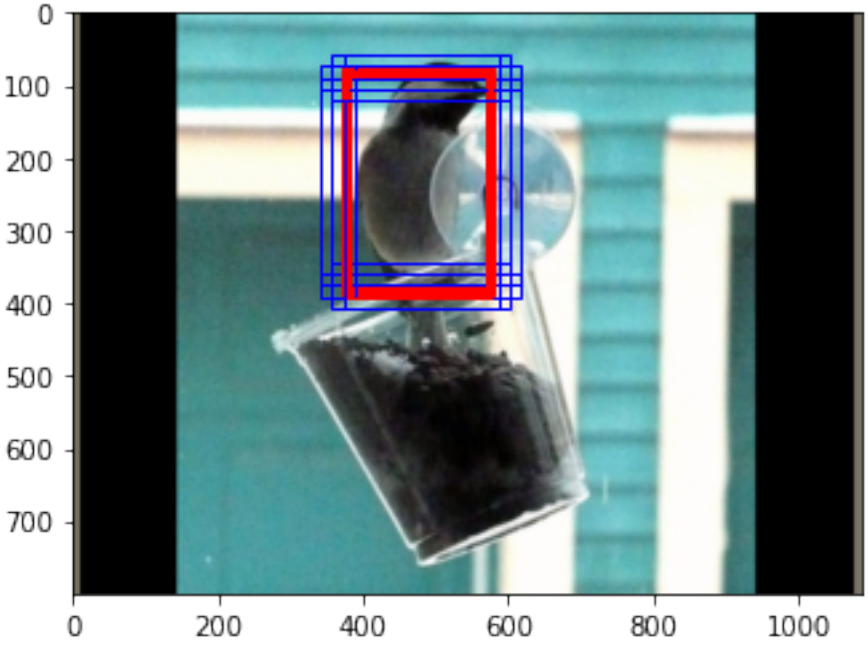

0
100
200
300
400
500
600
700
0
200
400
600
800
1000

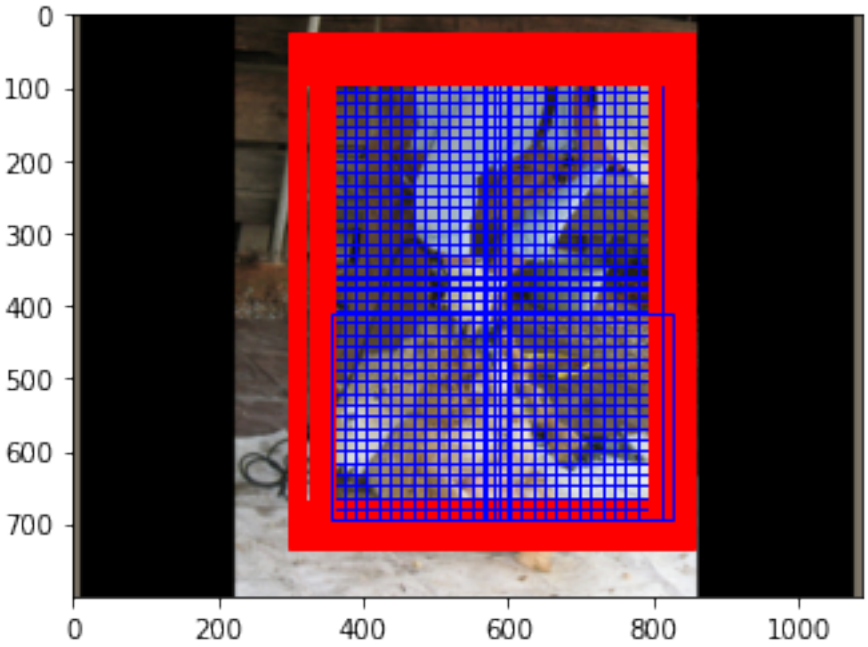

0
100
200
300
400
500
600
700
0
200
400
600
800
1000

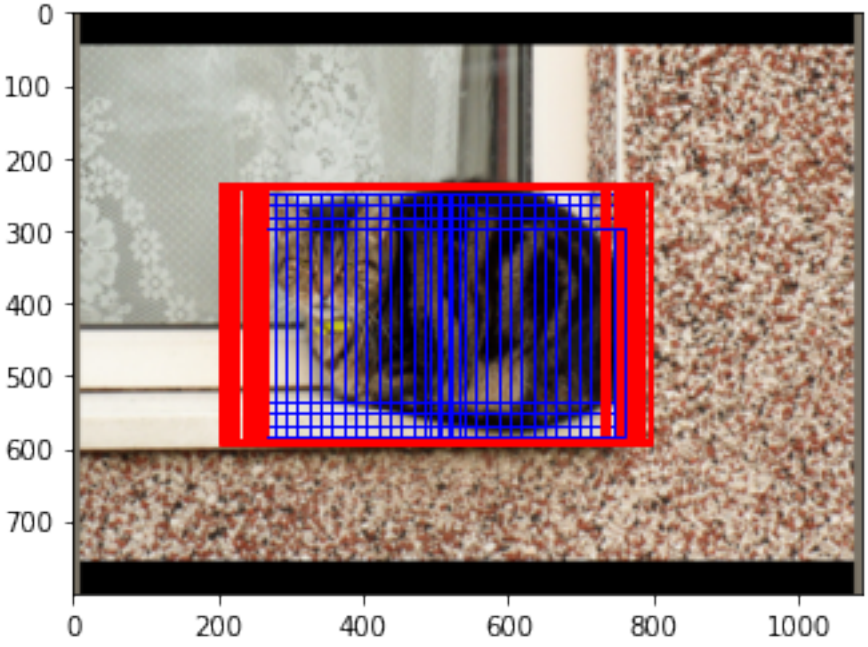

0
100
200
300
400
500
600
700
0
200
400
600
800
1000

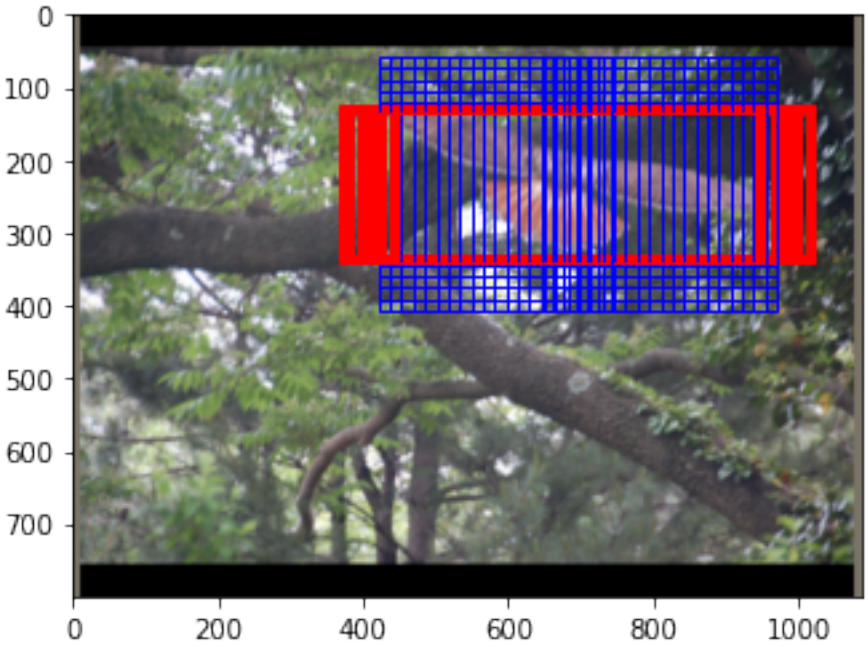

Photography 2013
0
100
200
300
400
500
600
700
0
200
400
600
800
1000

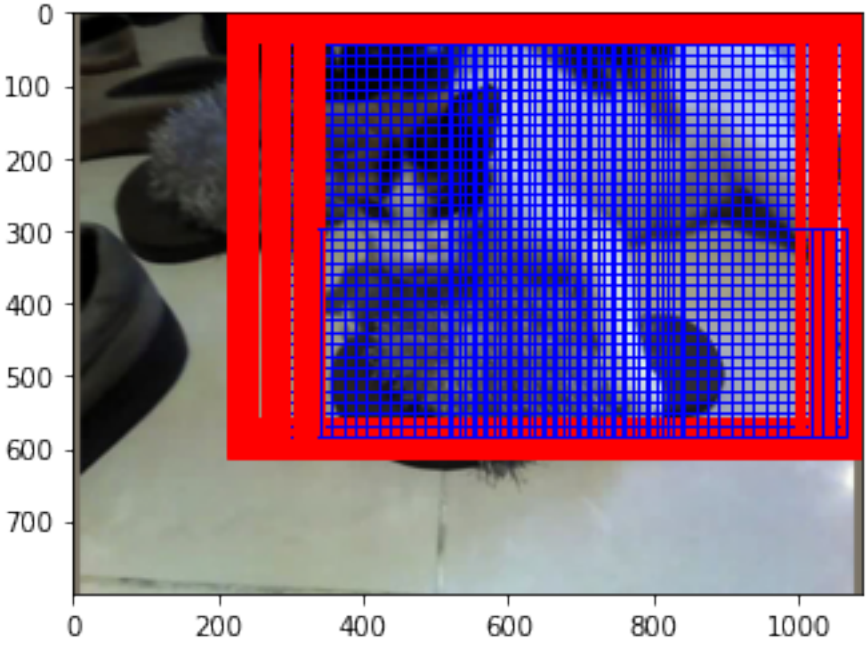

0
100
200
300
400
500
600
700
0
200
400
600
800
1000


0
100
200
300
400
500
600
700
0
200
400
600
800
1000


0
100
200
300
400
500
600
700
0
200
400
600
800
1000


0
100
200
300
400
500
600
700
0
200
400
600
800
1000


0
100
200
300
400
500
600
700
0
200
400
600
800
1000


0
100
200
300
400
500
600
700
0
200
400
600
800
1000

```
␣
↪----------------------------------------------------------------------------
```

```
       KeyboardInterrupt                         Traceback (most recent call␣
↪last)

       <ipython-input-52-74898d6a17c5> in <module>
        41           ax.add_patch(rect)
        42
   ---> 43       plt.show()
        44
        45       if(i < 10):


       /usr/local/lib/python3.7/dist-packages/matplotlib/pyplot.py in␣
↪show(*args, **kw)
       270     """
       271     global _show
   --> 272     return _show(*args, **kw)
       273
       274


       /usr/local/lib/python3.7/dist-packages/ipykernel/pylab/backend_inline.py␣
↪in show(close, block)
        41             display(
        42                 figure_manager.canvas.figure,
   ---> 43                 metadata=_fetch_figure_metadata(figure_manager.
↪canvas.figure)
        44             )
        45     finally:


       /usr/local/lib/python3.7/dist-packages/IPython/core/display.py in␣
↪display(include, exclude, metadata, transient, display_id, *objs, **kwargs)
       311             publish_display_data(data=obj, metadata=metadata,␣
↪**kwargs)
       312         else:
   --> 313             format_dict, md_dict = format(obj, include=include,␣
↪exclude=exclude)
       314             if not format_dict:
       315                 # nothing to display (e.g. _ipython_display_ took␣
↪over)


       /usr/local/lib/python3.7/dist-packages/IPython/core/formatters.py in␣
↪format(self, obj, include, exclude)
       178             md = None
       179             try:
   --> 180                 data = formatter(obj)
```

```
       181             except:
       182                 # FIXME: log the exception


       <decorator-gen-2> in __call__(self, obj)


       /usr/local/lib/python3.7/dist-packages/IPython/core/formatters.py in␣
↪catch_format_error(method, self, *args, **kwargs)
       222     """show traceback on failed format call"""
       223     try:
   --> 224         r = method(self, *args, **kwargs)
       225     except NotImplementedError:
       226         # don't warn on NotImplementedErrors


       /usr/local/lib/python3.7/dist-packages/IPython/core/formatters.py in␣
↪__call__(self, obj)
       339                 pass
       340             else:
   --> 341                 return printer(obj)
       342             # Finally look for special method names
       343             method = get_real_method(obj, self.print_method)


       /usr/local/lib/python3.7/dist-packages/IPython/core/pylabtools.py in␣
↪<lambda>(fig)
       242
       243     if 'png' in formats:
   --> 244         png_formatter.for_type(Figure, lambda fig: print_figure(fig,␣
↪'png', **kwargs))
       245     if 'retina' in formats or 'png2x' in formats:
       246         png_formatter.for_type(Figure, lambda fig:␣
↪retina_figure(fig, **kwargs))


       /usr/local/lib/python3.7/dist-packages/IPython/core/pylabtools.py in␣
↪print_figure(fig, fmt, bbox_inches, **kwargs)
       126
       127     bytes_io = BytesIO()
   --> 128     fig.canvas.print_figure(bytes_io, **kw)
       129     data = bytes_io.getvalue()
       130     if fmt == 'svg':
```

```
      /usr/local/lib/python3.7/dist-packages/matplotlib/backend_bases.py in␣
↪print_figure(self, filename, dpi, facecolor, edgecolor, orientation, format,␣
↪bbox_inches, **kwargs)
     2124                     orientation=orientation,
     2125                     bbox_inches_restore=_bbox_inches_restore,
  -> 2126                     **kwargs)
     2127             finally:
     2128                 if bbox_inches and restore_bbox:


      /usr/local/lib/python3.7/dist-packages/matplotlib/backends/backend_agg.
↪py in print_png(self, filename_or_obj, metadata, pil_kwargs, *args, **kwargs)
      512         }
      513
  --> 514         FigureCanvasAgg.draw(self)
      515         if pil_kwargs is not None:
      516             from PIL import Image


      /usr/local/lib/python3.7/dist-packages/matplotlib/backends/backend_agg.
↪py in draw(self)
      391              (self.toolbar._wait_cursor_for_draw_cm() if self.toolbar
      392               else nullcontext()):
  --> 393             self.figure.draw(self.renderer)
      394             # A GUI class may be need to update a window using this␣
↪draw, so
      395             # don't forget to call the superclass.


      /usr/local/lib/python3.7/dist-packages/matplotlib/artist.py in␣
↪draw_wrapper(artist, renderer, *args, **kwargs)
       36                 renderer.start_filter()
       37
  ---> 38             return draw(artist, renderer, *args, **kwargs)
       39         finally:
       40             if artist.get_agg_filter() is not None:


      /usr/local/lib/python3.7/dist-packages/matplotlib/figure.py in␣
↪draw(self, renderer)
     1734             self.patch.draw(renderer)
     1735             mimage._draw_list_compositing_images(
  -> 1736                 renderer, self, artists, self.suppressComposite)
     1737
     1738             renderer.close_group('figure')
```

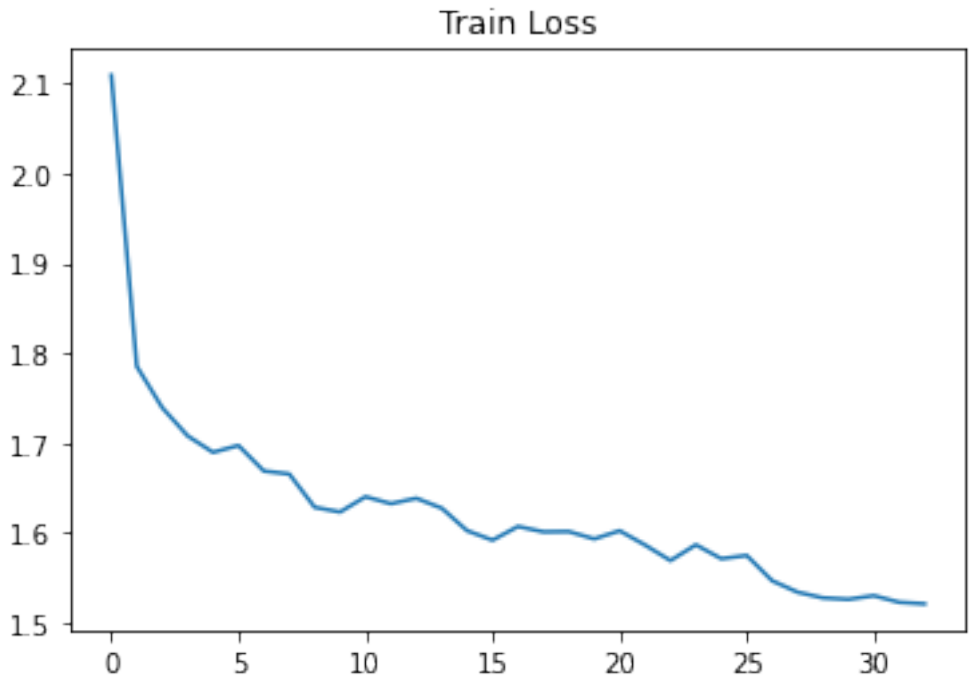
```
      /usr/local/lib/python3.7/dist-packages/matplotlib/image.py in␣
↪_draw_list_compositing_images(renderer, parent, artists, suppress_composite)
      135     if not_composite or not has_images:
      136         for a in artists:
  --> 137             a.draw(renderer)
      138     else:
      139         # Composite any adjacent images together


      /usr/local/lib/python3.7/dist-packages/matplotlib/artist.py in␣
↪draw_wrapper(artist, renderer, *args, **kwargs)
       36                 renderer.start_filter()
       37
  ---> 38             return draw(artist, renderer, *args, **kwargs)
       39         finally:
       40             if artist.get_agg_filter() is not None:


      /usr/local/lib/python3.7/dist-packages/matplotlib/axes/_base.py in␣
↪draw(self, renderer, inframe)
     2628             renderer.stop_rasterizing()
     2629
  -> 2630         mimage._draw_list_compositing_images(renderer, self, artists)
     2631
     2632         renderer.close_group('axes')


      /usr/local/lib/python3.7/dist-packages/matplotlib/image.py in␣
↪_draw_list_compositing_images(renderer, parent, artists, suppress_composite)
      135     if not_composite or not has_images:
      136         for a in artists:
  --> 137             a.draw(renderer)
      138     else:
      139         # Composite any adjacent images together


      /usr/local/lib/python3.7/dist-packages/matplotlib/artist.py in␣
↪draw_wrapper(artist, renderer, *args, **kwargs)
       36                 renderer.start_filter()
       37
  ---> 38             return draw(artist, renderer, *args, **kwargs)
       39         finally:
       40             if artist.get_agg_filter() is not None:


      /usr/local/lib/python3.7/dist-packages/matplotlib/image.py in draw(self,␣
↪renderer, *args, **kwargs)
```

```
      624         else:
      625             im, l, b, trans = self.make_image(
  --> 626                 renderer, renderer.get_image_magnification())
      627             if im is not None:
      628                 renderer.draw_image(gc, l, b, im)


      /usr/local/lib/python3.7/dist-packages/matplotlib/image.py in␣
↪make_image(self, renderer, magnification, unsampled)
      915             self._A, bbox, transformed_bbox,
      916             self.get_clip_box() or self.axes.bbox,
  --> 917             magnification, unsampled=unsampled)
      918
      919     def _check_unsampled_image(self, renderer):


      /usr/local/lib/python3.7/dist-packages/matplotlib/image.py in␣
↪_make_image(self, A, in_bbox, out_bbox, clip_bbox, magnification, unsampled,␣
↪round_to_pixel_border)
      522                     self, A[…, 3], out_shape, t, alpha=alpha)
      523                 output = _resample(  # resample rgb channels
  --> 524                     self, _rgb_to_rgba(A[…, :3]), out_shape, t,␣
↪alpha=alpha)
      525                 output[…, 3] = output_alpha  # recombine rgb and␣
↪alpha
      526


      /usr/local/lib/python3.7/dist-packages/matplotlib/image.py in␣
↪_resample(image_obj, data, out_shape, transform, resample, alpha)
      200                     alpha,
      201                     image_obj.get_filternorm(),
  --> 202                     image_obj.get_filterrad())
      203     return out
      204


      KeyboardInterrupt:
```

0.6 Training and Validation curves that show the total loss, the loss of the classifier and the loss of the regressor.

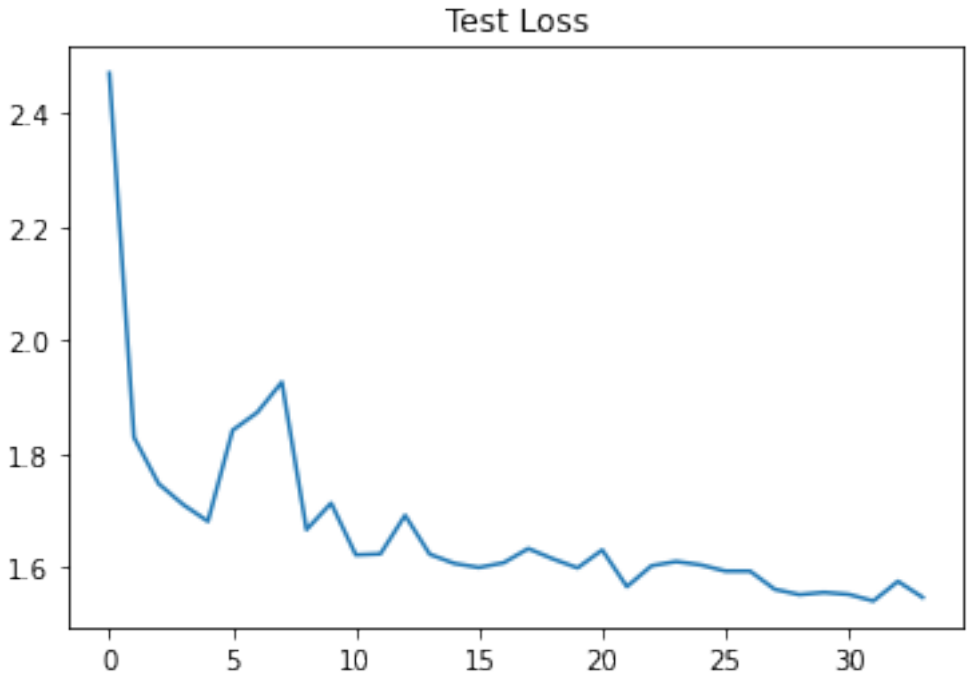
[20]:
```
def loss_curves():
  t_loss = []
  t_cat_loss = []
  t_reg_loss = []
  for idx in range(len(rpn_net.train_losses)):
    t_loss.append(rpn_net.train_losses[idx][0])
    t_cat_loss.append(rpn_net.train_losses[idx][1])
    t_reg_loss.append(rpn_net.train_losses[idx][2])

  plt.plot(t_loss)
  plt.title("Train Loss")
  plt.show()
  plt.plot(t_cat_loss)
  plt.title("Train Classifier Loss")
  plt.show()
  plt.plot(t_reg_loss)
  plt.title("Train Regressor Loss")
  plt.show()

  v_loss = []
  v_cat_loss = []
  v_reg_loss = []
  for idx in range(len(rpn_net.val_losses)):
    v_loss.append(rpn_net.val_losses[idx][0])
    v_cat_loss.append(rpn_net.val_losses[idx][1])
    v_reg_loss.append(rpn_net.val_losses[idx][2])

  plt.plot(v_loss)
  plt.xlabel('Epochs')
  plt.ylabel('Epochs')
  plt.title("Test Loss")
  plt.show()
  plt.plot(v_cat_loss)
  plt.title("Test Classifier Loss")
  plt.show()
  plt.plot(v_reg_loss)
  plt.title("Test Regressor Loss")
  plt.show()
```

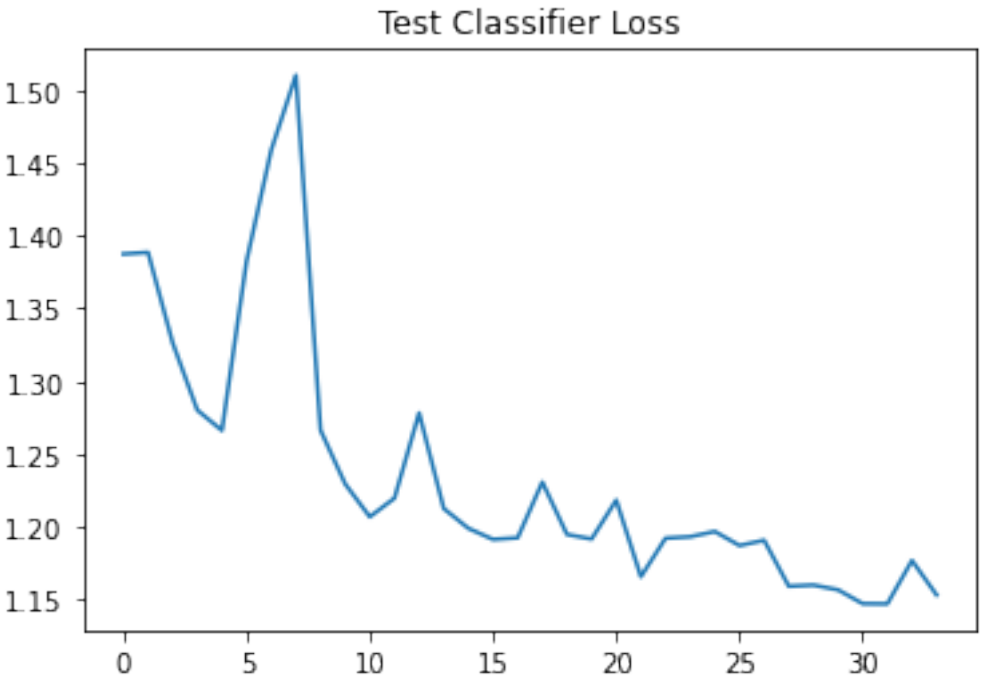
[22]:
```
loss_curves()
```

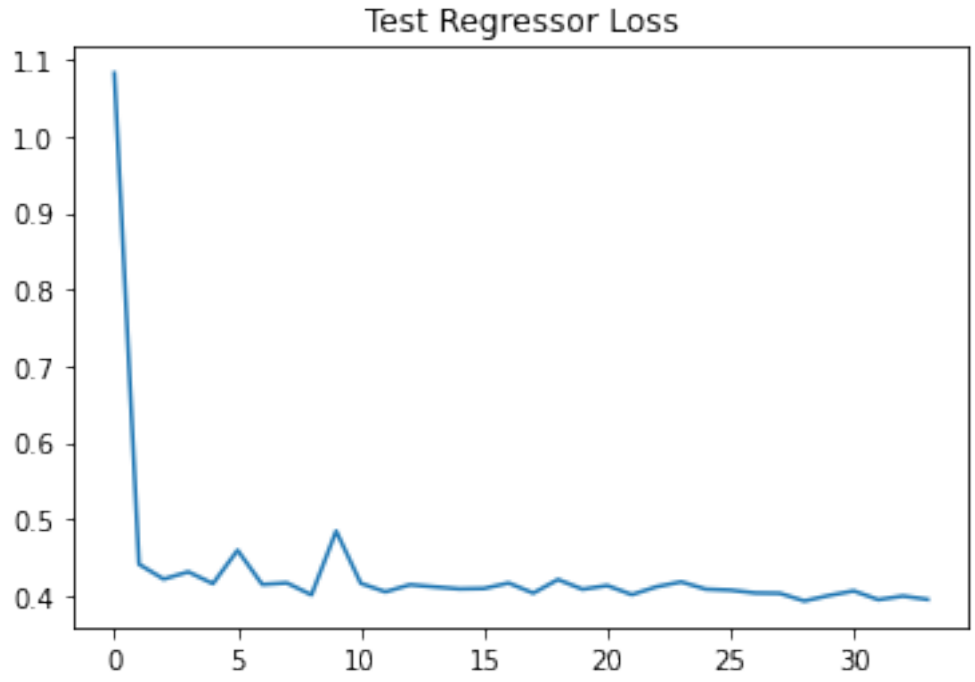

Train Loss
2.1
2.0
1.9
1.8
1.7
1.6
1.5
0
5
10
15
20
25
30


Train Classifier Loss
1.40
1.35
1.30
1.25
1.20
1.15
0
5
10
15
20
25
30


Train Regressor Loss
0.70
0.65
0.60
0.55
0.50
0.45
0.40
0
5
10
15
20
25
30


Test Loss
2.4
2.2
2.0
1.8
1.6
0
5
10
15
20
25
30


Test Classifier Loss
1.50
1.45
1.40
1.35
1.30
1.25
1.20
1.15
0
5
10
15
20
25
30


Test Regressor Loss
1.1
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0
5
10
15
20
25
30

0.7 5. Report of the point-wise accuracy of the proposal classifier. Plots of the top 20 proposals for some images of the test set.

[81]:
```
def accuracy_value():
  for i,batch in enumerate(test_loader,0):
      images = batch['images']
      images = torch.stack(images[:])
      indexes= batch['index']
      boxes  = batch['bbox']

      gt,ground_coord     = rpn_net.create_batch_truth(boxes, indexes,␣
↪(800,1088))
      logits, bbox_regs   = rpn_net.forward(images.to(device))

      indexes = torch.where(gt == 1)
      num = torch.sum(logits[indexes])
      accuracy = num/len(indexes[0])

      # if i == 10:
      print(accuracy)
      if i == 2:
        break
```

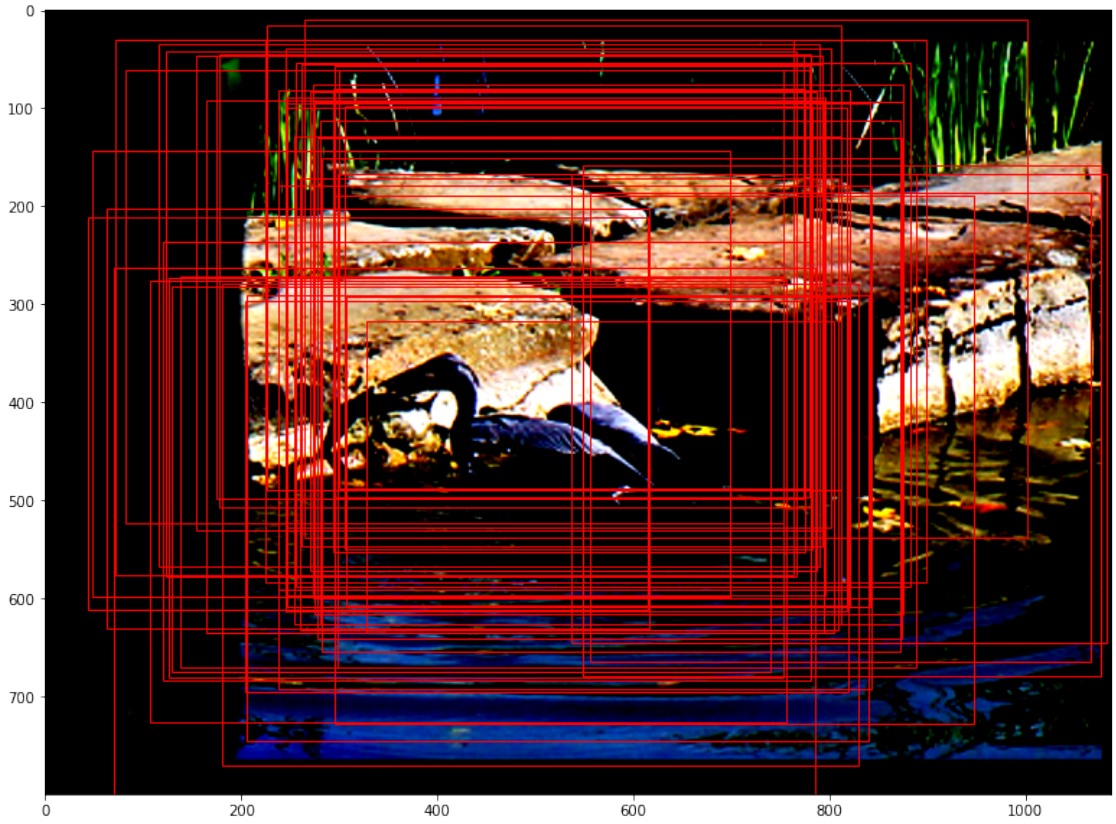
[82]:
```
accuracy_value()
```

```
tensor(0.6098, device='cuda:0', grad_fn=<DivBackward0>)
tensor(0.7384, device='cuda:0', grad_fn=<DivBackward0>)
tensor(0.6123, device='cuda:0', grad_fn=<DivBackward0>)
```

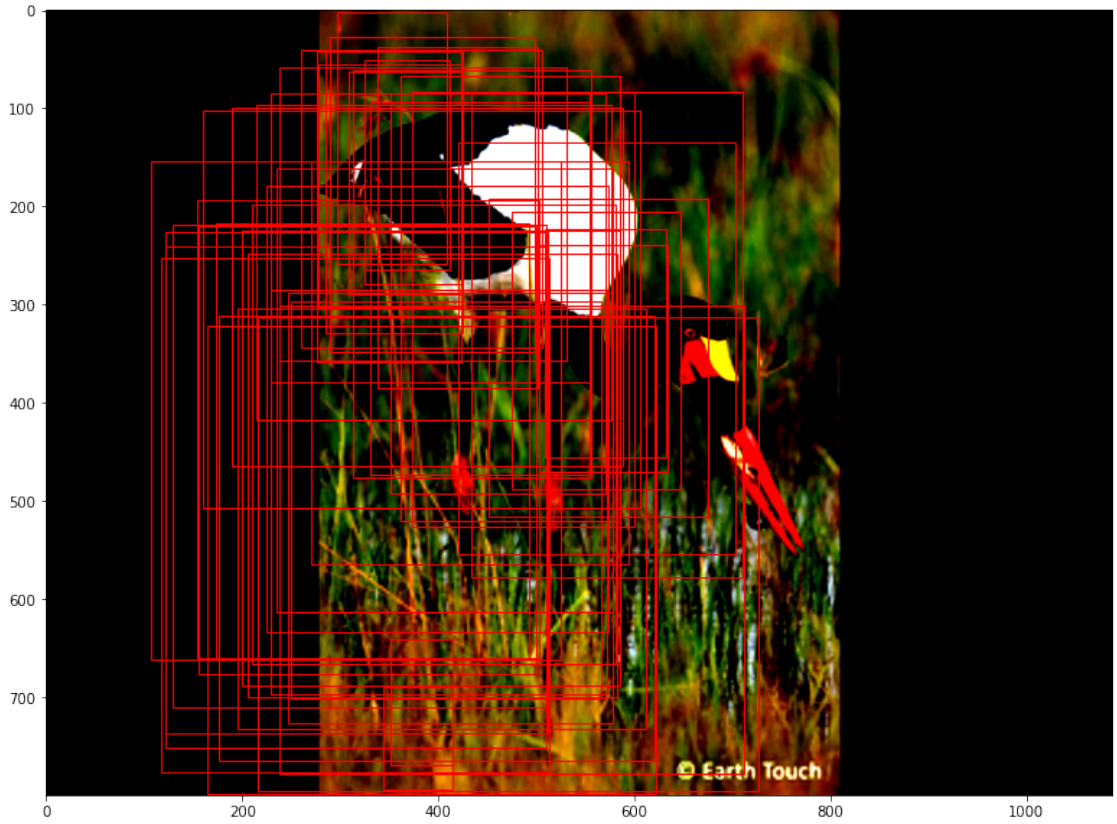
[65]:
```
def point_wise_proposal():
  for i,batch in enumerate(test_loader,0):
      images = batch['images']
      images = torch.stack(images[:])

      logits, bbox_regs = rpn_net.forward(images.to(device))
      scores_sorted_list, pre_nms_matrix_list, nms_clas_list, nms_prebox_list =␣
↪rpn_net.postprocess(logits.detach().cpu(), bbox_regs.detach().cpu(),␣
↪IOU_thresh = 0.5 ,keep_num_preNMS = 20, keep_num_postNMS = 10)

      plot_bounding_box(images[0].permute(1,2,0), pre_nms_matrix_list[0].
↪detach().cpu().numpy())
      # plt.title("Top 20 proposals of Test Images")
      plt.show()

      if i == 5:
          break
```

[66]:
```
point_wise_proposal()
```

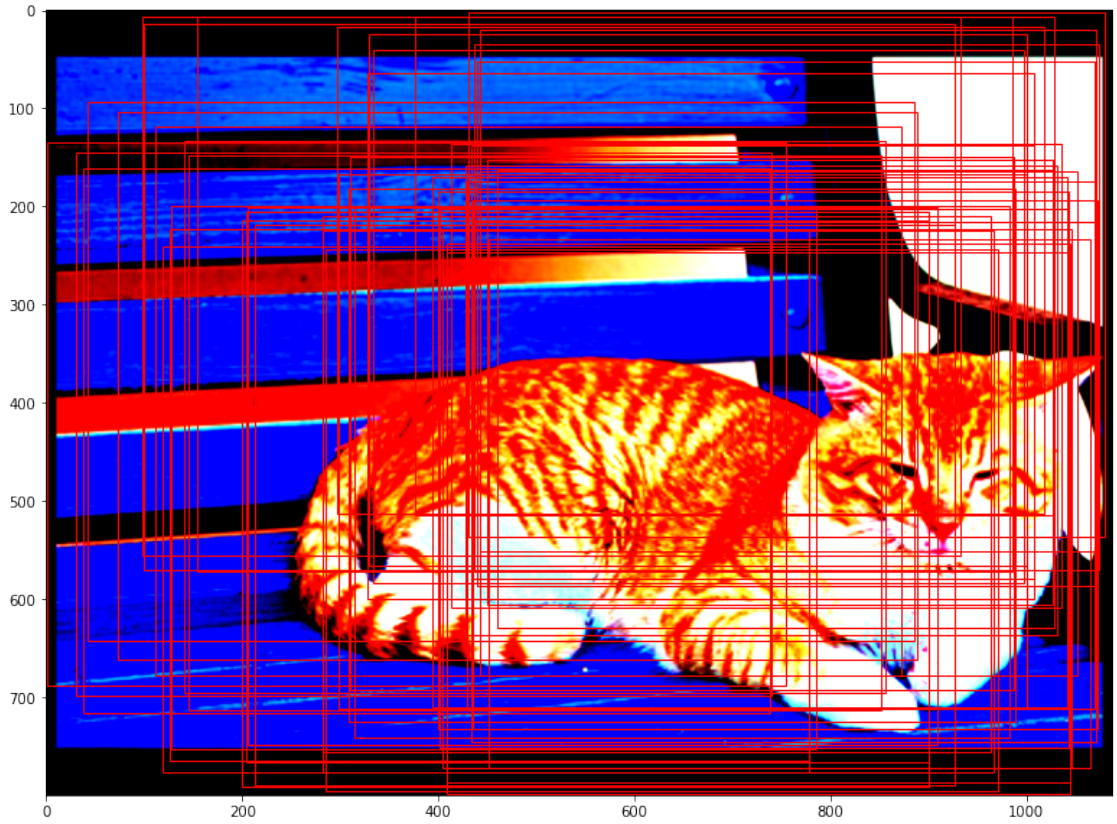
```
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
```



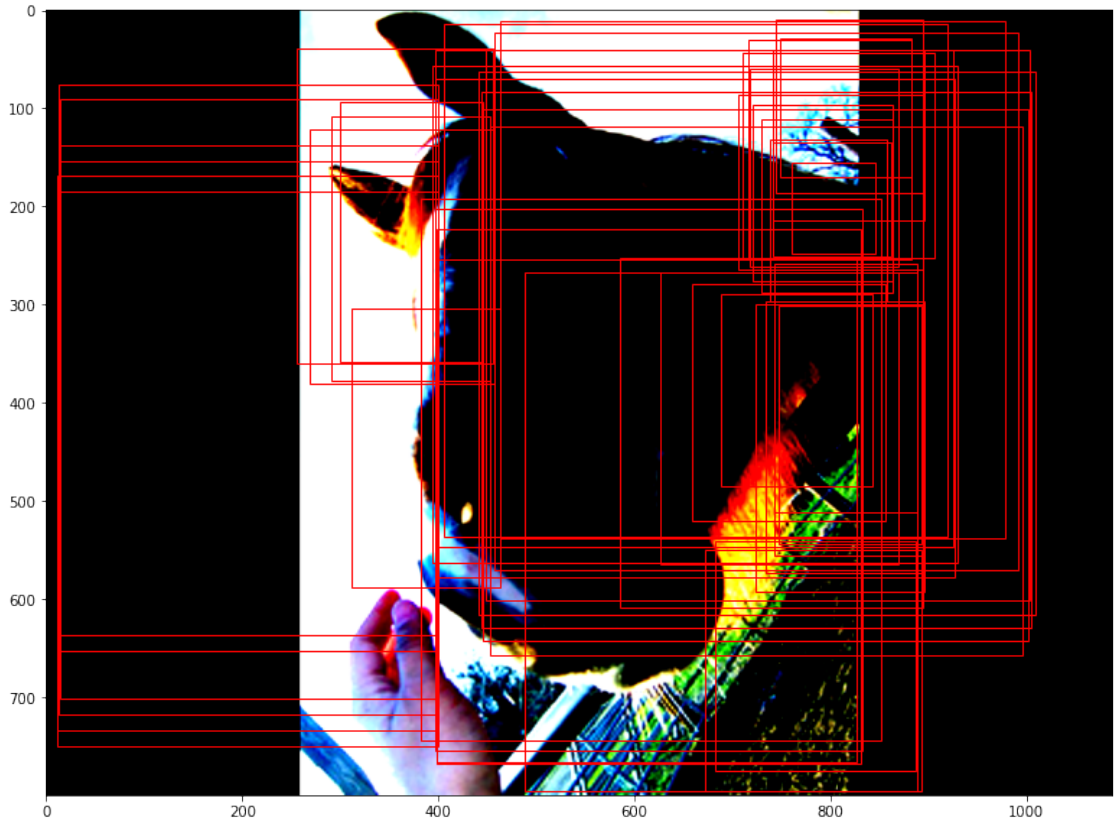

```
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
```



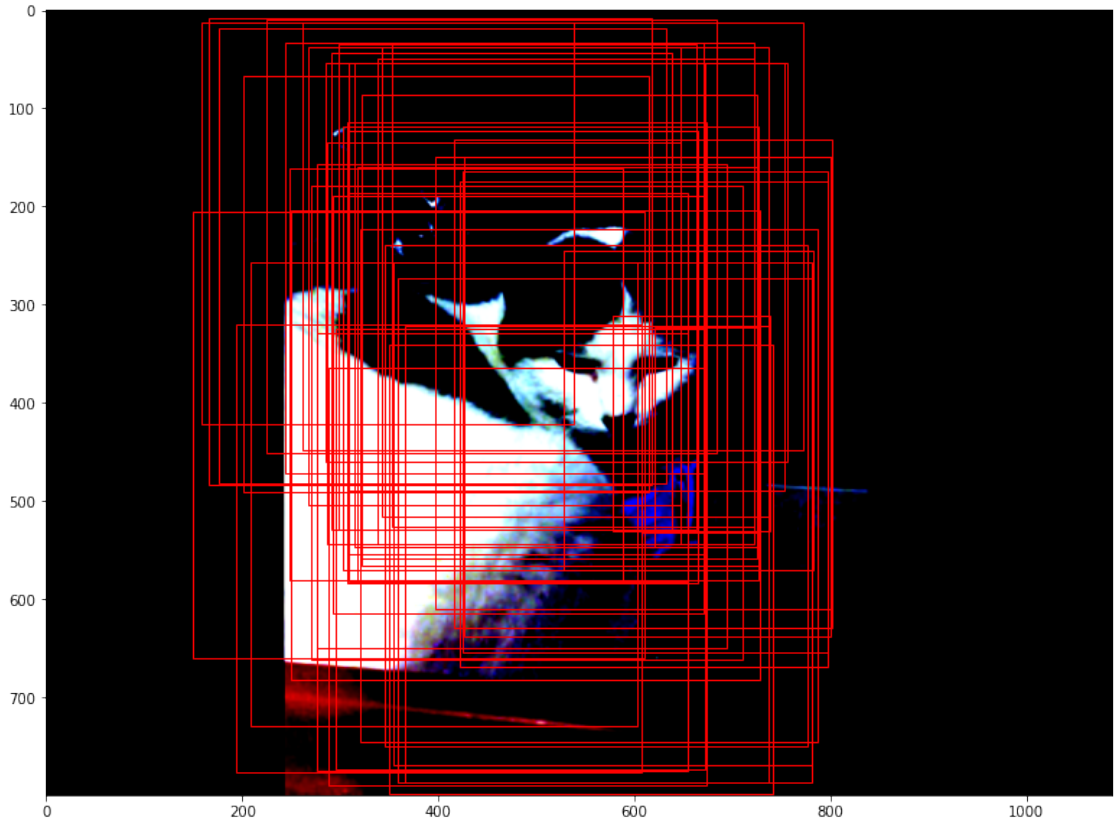

```
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
```



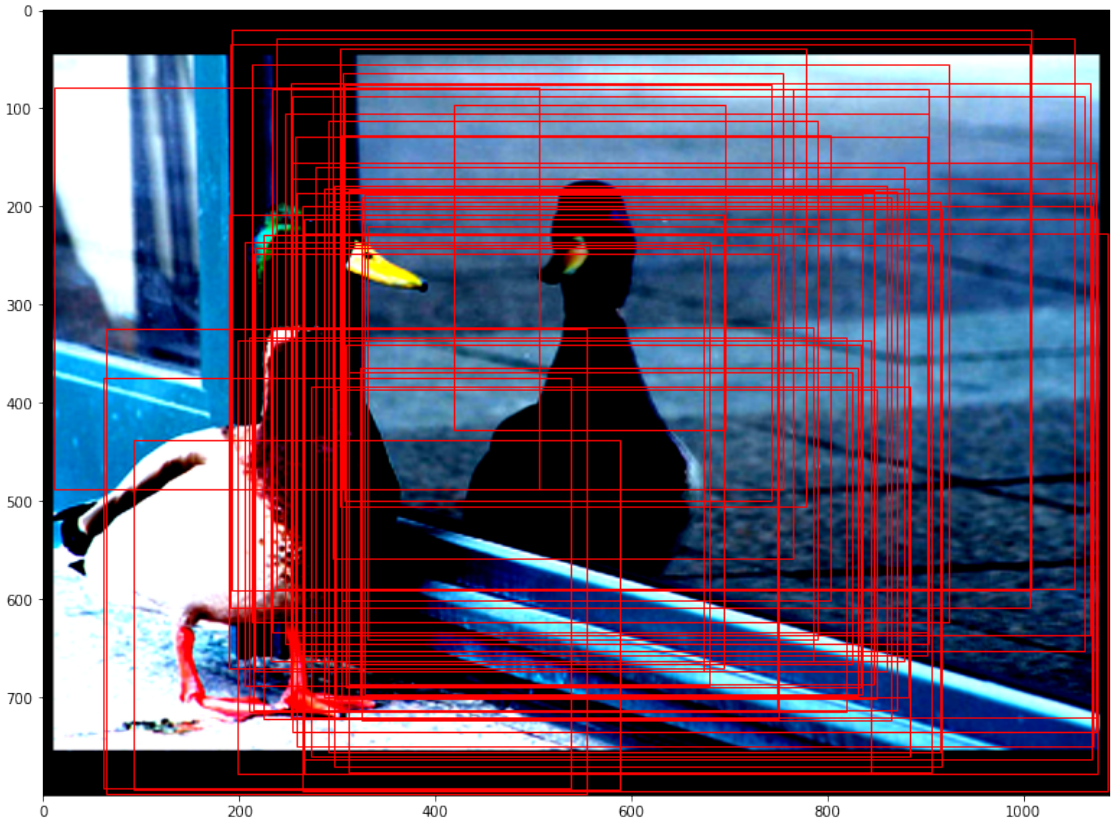

```
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
```




```
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
```




```
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
```



0.8 Visualization of the proposed boxes before and after the NMS

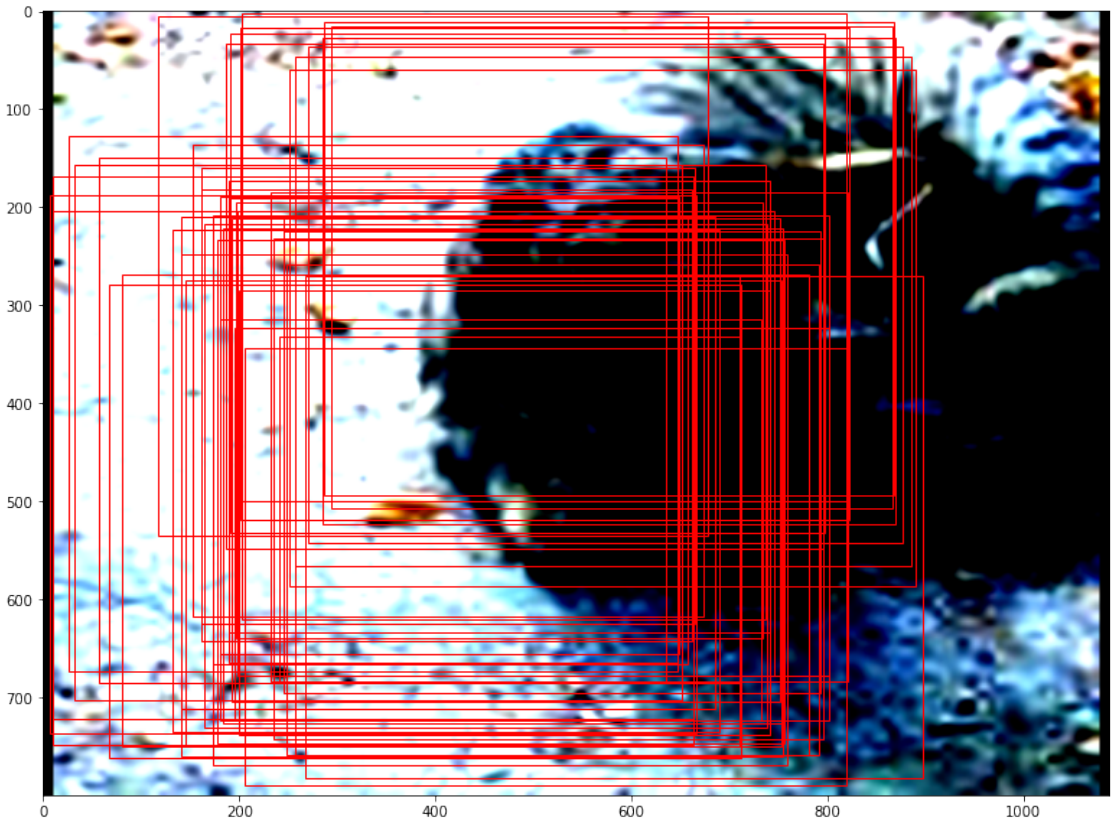
Post Training Results

[71]:
```
def preposrnms():
  for i,batch in enumerate(train_loader,0):
      images = batch['images']
      images = torch.stack(images[:])

      logits, bbox_regs = rpn_net.forward(images.to(device))
      scores_sorted_list, pre_nms_matrix_list, nms_clas_list, nms_prebox_list =␣
↪rpn_net.postprocess(logits.detach().cpu(), bbox_regs.detach().cpu(),␣
↪IOU_thresh = 0.5 ,keep_num_preNMS = 20, keep_num_postNMS = 10)

      plot_bounding_box(images[0].permute(1,2,0), pre_nms_matrix_list[0].
↪detach().cpu().numpy())
      plot_bounding_box(images[0].permute(1,2,0), nms_prebox_list[0].detach().
↪cpu().numpy())

      if i == 5:
```

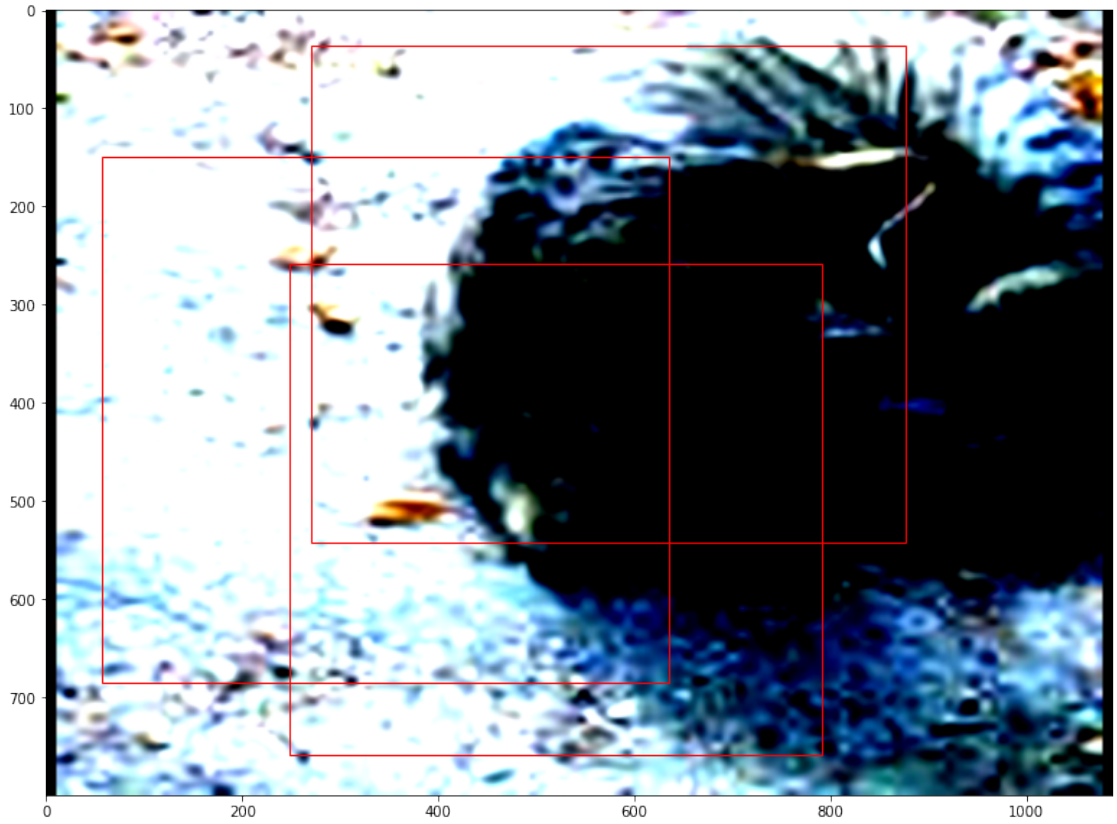
```
            break
```

[72]:
```
preposrnms()
```

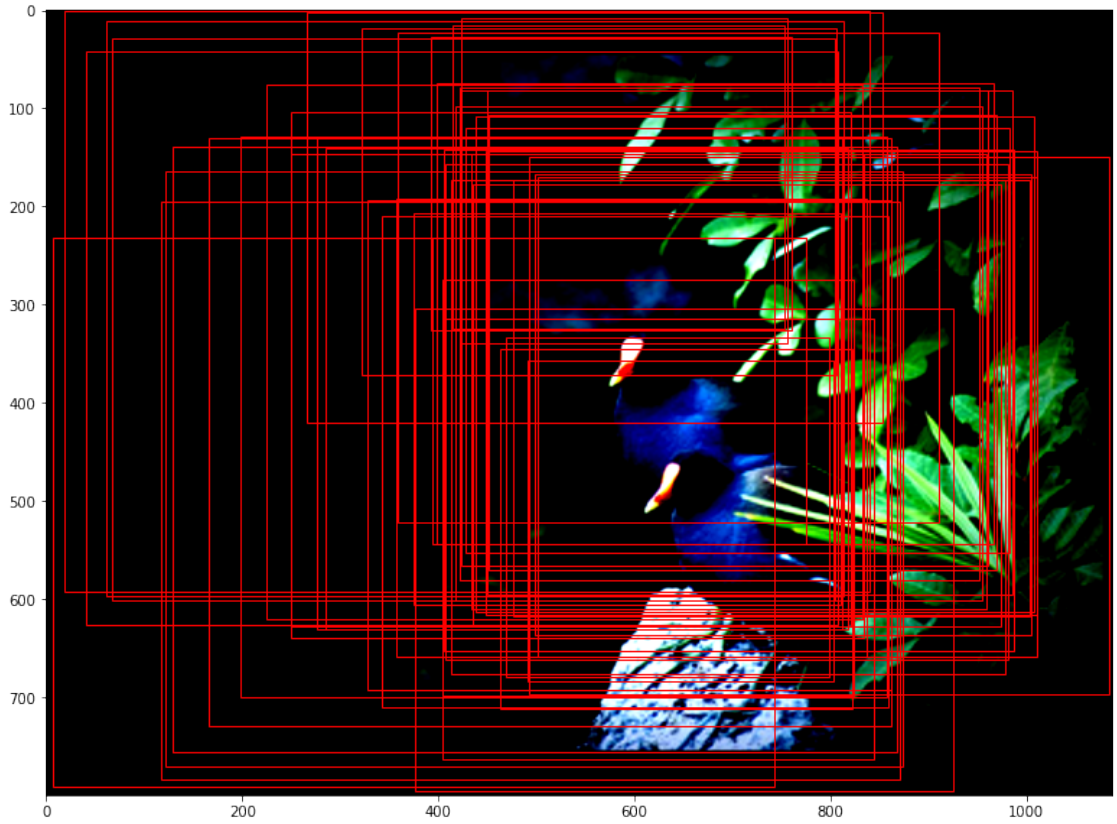
```
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
```



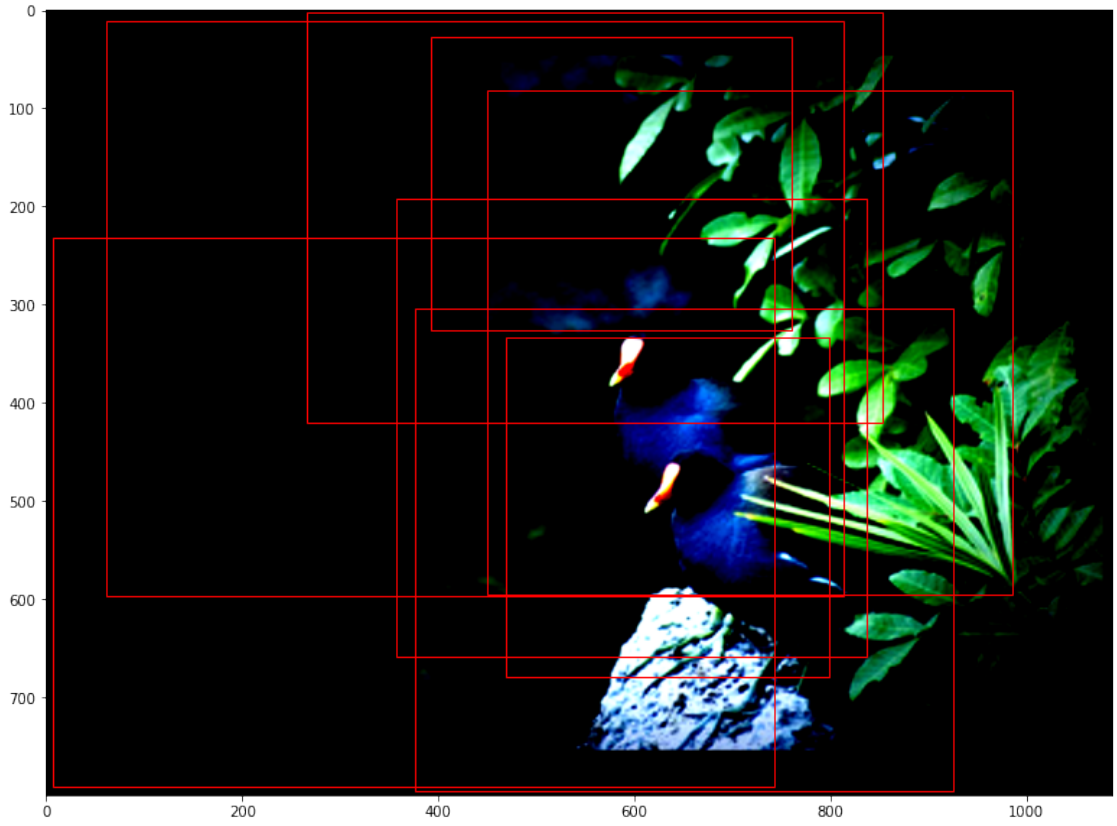

```
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
```



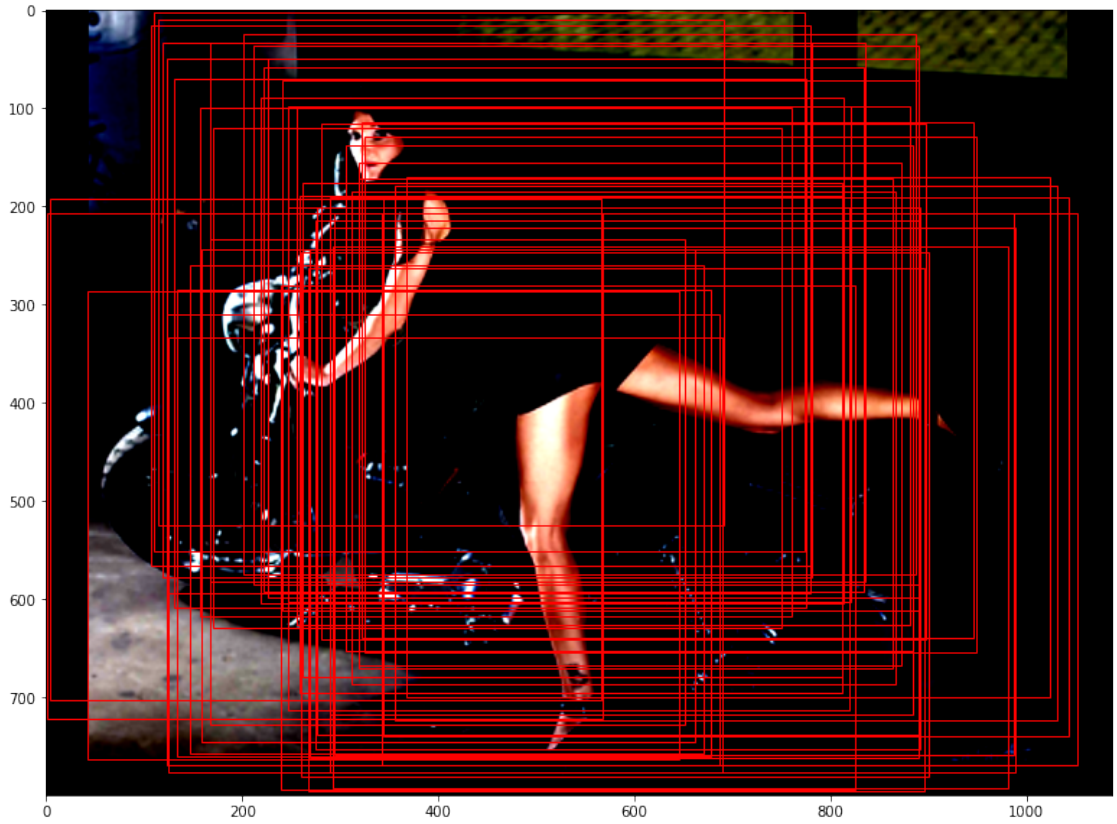

```
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
```



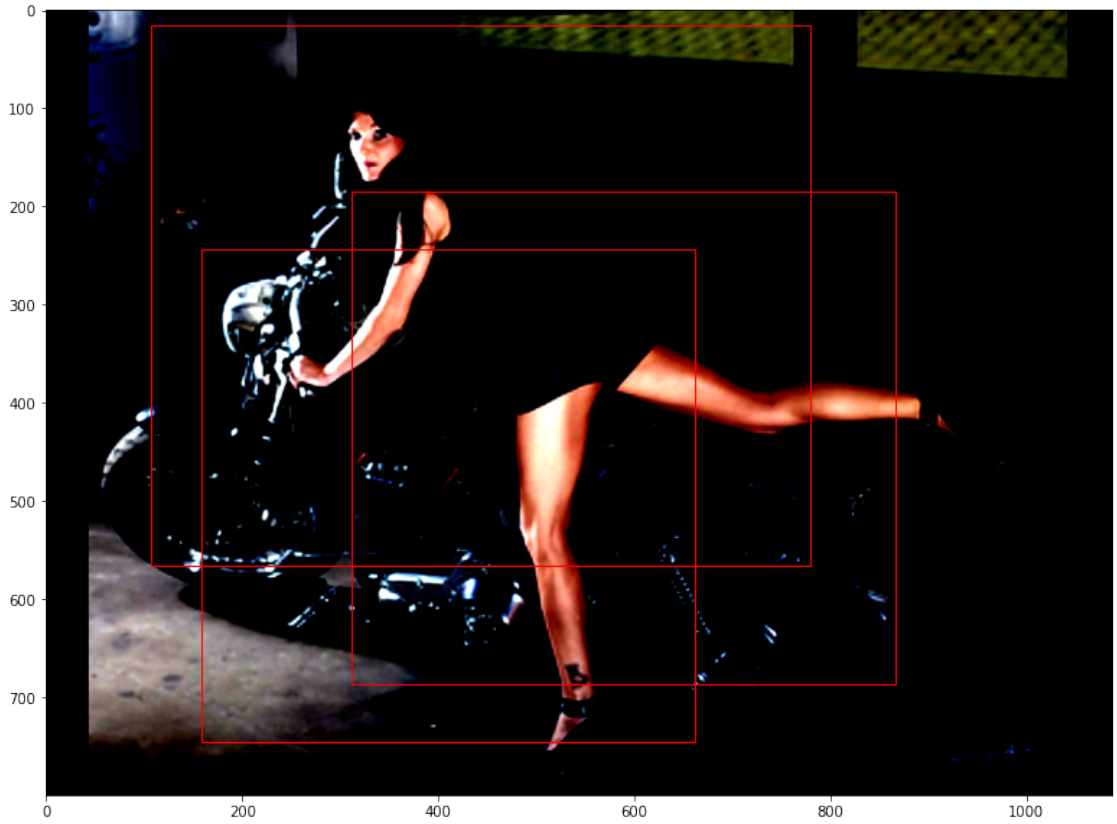

```
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
```



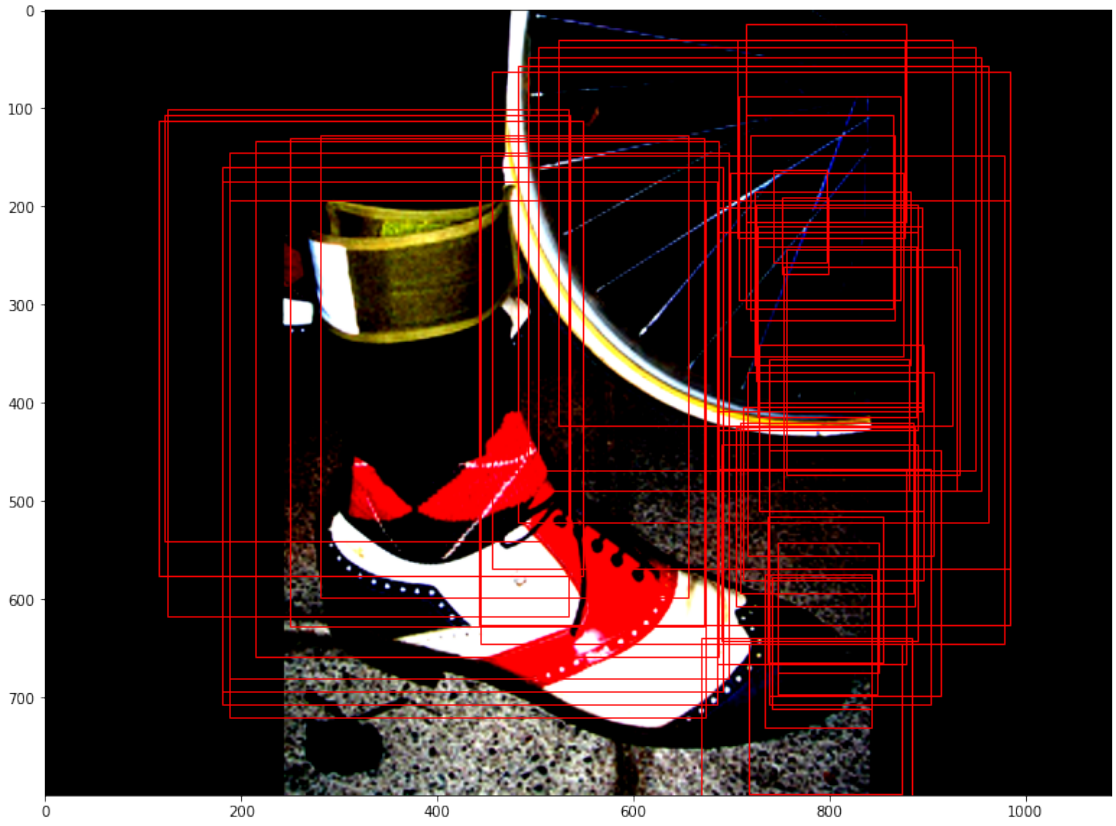

```
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
```



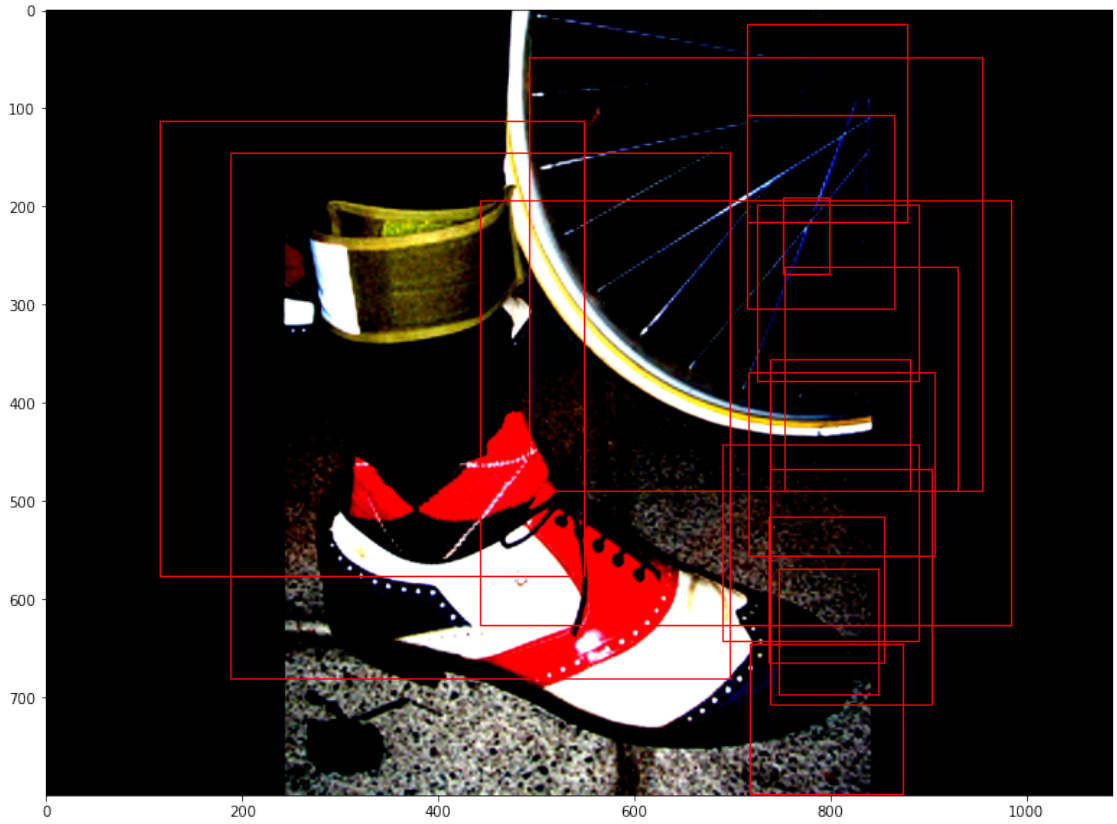

```
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
```



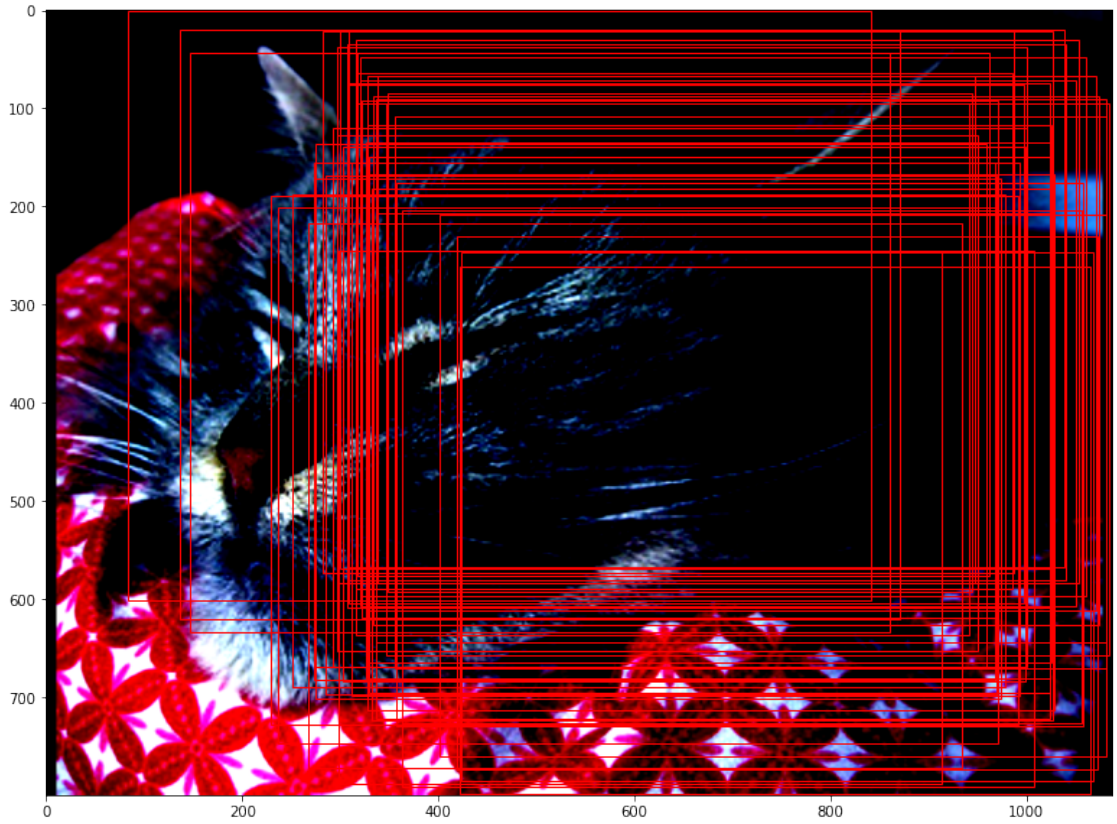

```
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
```



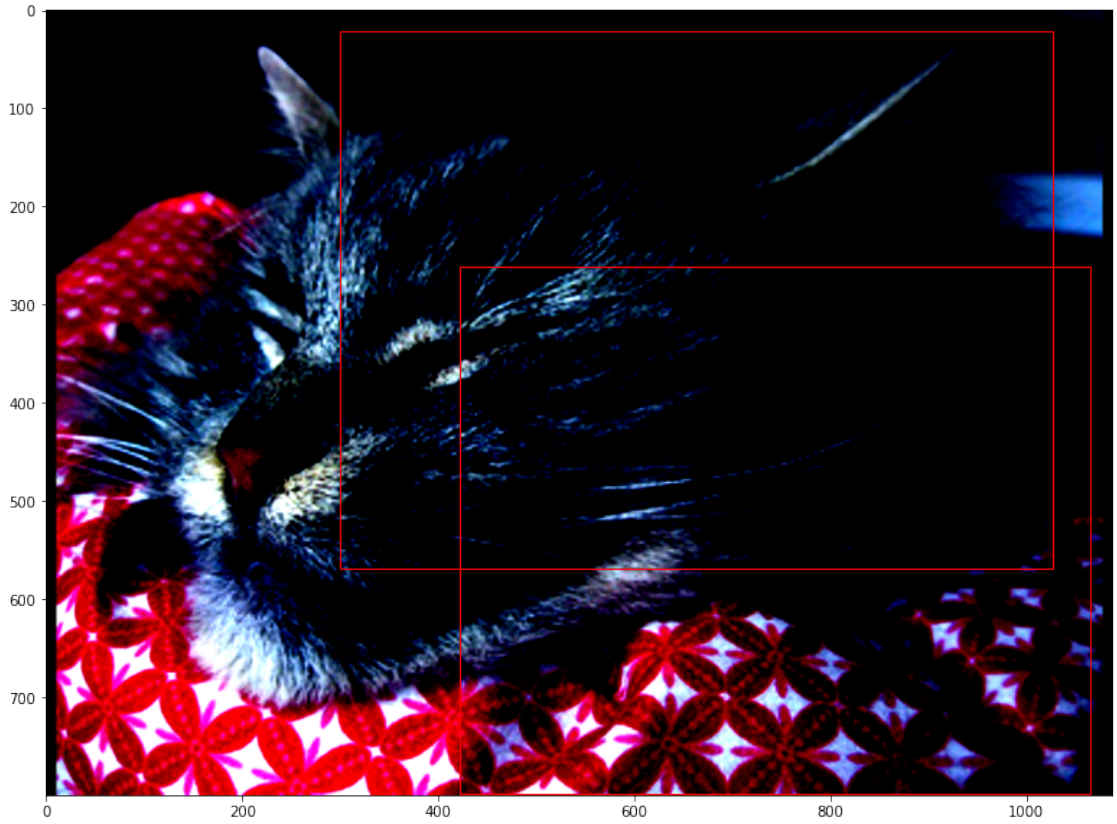

```
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
```



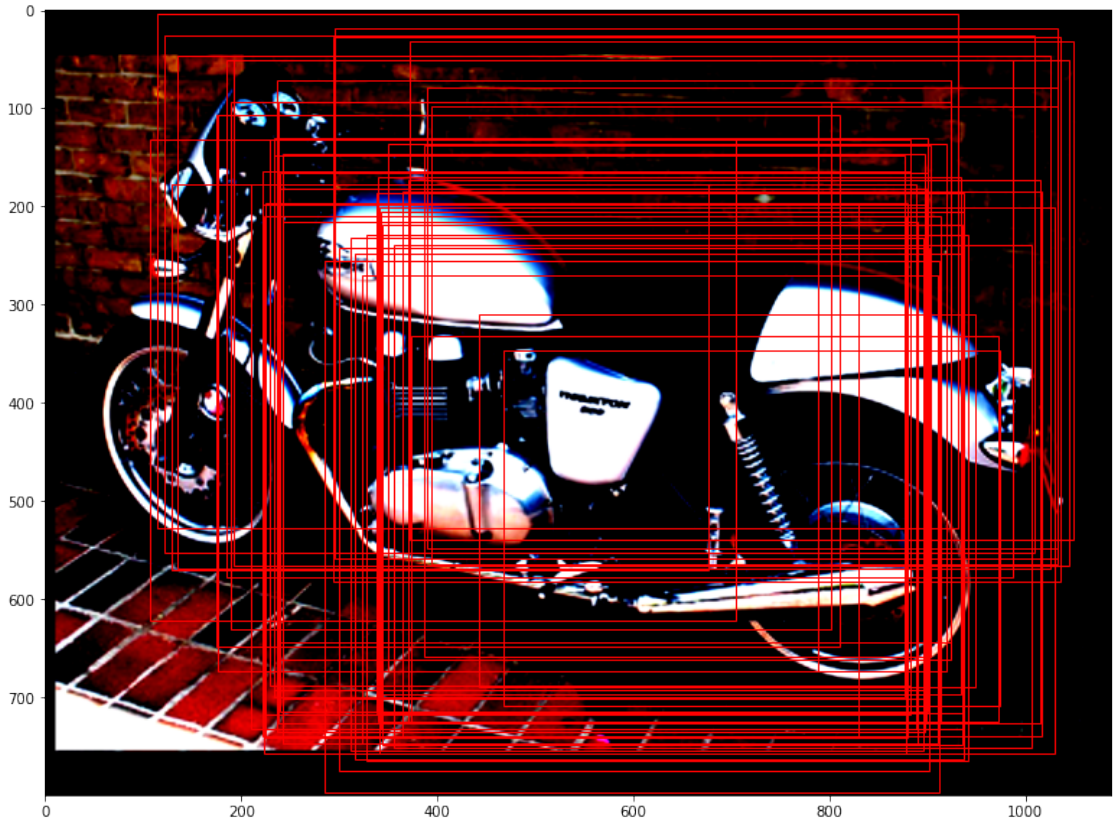

```
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
```



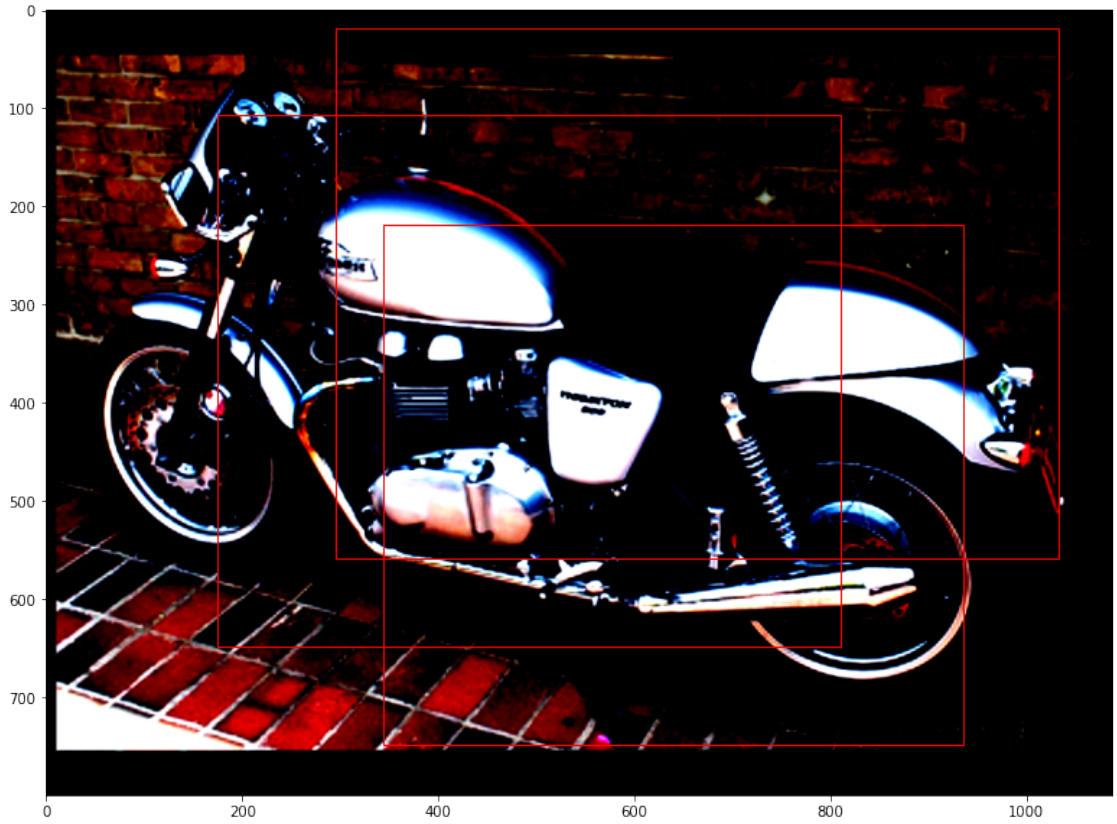

```
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
```




```
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
```




```
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with
RGB data ([0..1] for floats or [0..255] for integers).
```



0.9 CELL TO RUN, FOR YOUR PERUSAL

[ ]:
```
# file path and make a list
##########################PLEASE CHANGE THE FILE PATH␣
 ↪#########################################################3
imgs_path   = '/content/drive/MyDrive/UPENN/SEM 3/CIS680/HW3/
 ↪hw3_mycocodata_img_comp_zlib.h5'
masks_path  = '/content/drive/MyDrive/UPENN/SEM 3/CIS680/HW3/
 ↪hw3_mycocodata_mask_comp_zlib.h5'
labels_path = '/content/drive/MyDrive/UPENN/SEM 3/CIS680/HW3/
 ↪hw3_mycocodata_labels_comp_zlib.npy'
bboxes_path = '/content/drive/MyDrive/UPENN/SEM 3/CIS680/HW3/
 ↪hw3_mycocodata_bboxes_comp_zlib.npy'
paths = [imgs_path, masks_path, labels_path, bboxes_path]
# load the data into data.Dataset
dataset = BuildDataset(paths)

# build the dataloader
# set 20% of the dataset as the training data
```

```
full_size = len(dataset)
train_size = int(full_size * 0.8)
test_size = full_size - train_size
# random split the dataset into training and testset

train_dataset, test_dataset = torch.utils.data.random_split(dataset,␣
↪[train_size, test_size])
rpn_net = RPNHead()
# push the randomized training data into the dataloader

# train_loader = DataLoader(train_dataset, batch_size=2, shuffle=True,␣
↪num_workers=0)
# test_loader  = DataLoader(test_dataset, batch_size=2, shuffle=False,␣
↪num_workers=0)
batch_size = 4
train_build_loader = BuildDataLoader(train_dataset, batch_size=batch_size,␣
↪shuffle=True, num_workers=0)
train_loader = train_build_loader.loader()
test_build_loader = BuildDataLoader(test_dataset, batch_size=batch_size,␣
↪shuffle=False, num_workers=0)
test_loader = test_build_loader.loader()

#Model Checkpoints
val_checkpoint_callback = ModelCheckpoint(
    monitor="val_loss",
    dirpath="./training_data_new_model_2",
    filename="val_loss{epoch:02d}-{val_loss:.2f}",
    save_top_k=3,
    mode="min",
)
train_checkpoint_callback = ModelCheckpoint(
    monitor="train_loss",
    dirpath="./training_data_new_model_2",
    filename="train_loss{epoch:02d}-{train_loss:.2f}",
    save_top_k=3,
    mode="min",
)

#Train
tb_logger = pl_loggers.TensorBoardLogger("logs2/")
trainer = pl.Trainer(gpus=1,logger=tb_logger,␣
↪max_epochs=40,callbacks=[val_checkpoint_callback,train_checkpoint_callback])
trainer.fit(rpn_net,train_loader,test_loader)

##########################################################################################
```

```
# HISTOGRAM O/P
histogram(bboxes_path)

#########################################################################################################

#Positive anchors and Ground truth boxes
positive_anchors_gt_boxes()

#########################################################################################################

# LOSS CURVES
loss_curves()

#########################################################################################################

#Report of the point-wise accuracy of the proposal classifier.
point_wise_proposal()

#########################################################################################################

# PRE AND POST NMS
preposrnms()
```