notebook_Mask_RCNN_car_damage_pred_nasir

August 22, 2019

0.1 DamageDetection with MaskRCNN

- Nasir Uddin
- Data visualization of car damage images and automated car damage detection example.

[1]:
```
#import all the packages including custom functions of Matterport Mask R-CNN␣
 ↪repository
import os
import sys
import itertools
import math
import logging
import json
import re
import random
from collections import OrderedDict
import numpy as np
import matplotlib
import matplotlib.pyplot as plt
import matplotlib.patches as patches
import matplotlib.lines as lines
from matplotlib.patches import Polygon

# Import Mask RCNN
#sys.path.append(ROOT_DIR)  # To find local version of the library
from mrcnn import utils
from mrcnn import visualize
from mrcnn.visualize import display_images
from mrcnn import model
import mrcnn.model as modellib
from mrcnn.model import log
import cv2
import custom,custom_1
import imgaug,h5py,IPython

%matplotlib inline
```

```
Using TensorFlow backend.
```

Setting up the configuration - root directory,data path setting up the ,log file path and model object(weight matrix)for inference (prediction)

[2]:
```
# Root directory of the project
ROOT_DIR = os.getcwd()
sys.path.append(ROOT_DIR)  # To find local version of the library
MODEL_DIR = os.path.join(ROOT_DIR, "logs")
custom_WEIGHTS_PATH = "mask_rcnn_coco.h5"  # TODO: update this path for best␣
 ↪performing iteration weights
config = custom.CustomConfig()
custom_DIR = os.path.join(ROOT_DIR, "custom/")
custom_DIR
```

[2]:
```
'/home/nasir/Desktop/carcnn/car-damage-detection-using-CNN/custom/'
```

loading the data

[3]:
```
# Load dataset
dataset = custom_1.CustomDataset()
dataset.load_custom(custom_DIR, "train")

# Must call before using the dataset
dataset.prepare()

print("Image Count: {}".format(len(dataset.image_ids)))
print("Class Count: {}".format(dataset.num_classes))
for i, info in enumerate(dataset.class_info):
    print("{:3}. {:50}".format(i, info['name']))
```

```
Image Count: 49
Class Count: 2
  0. BG
  1. scratch
```

We will visualize few car damage(scratch) images

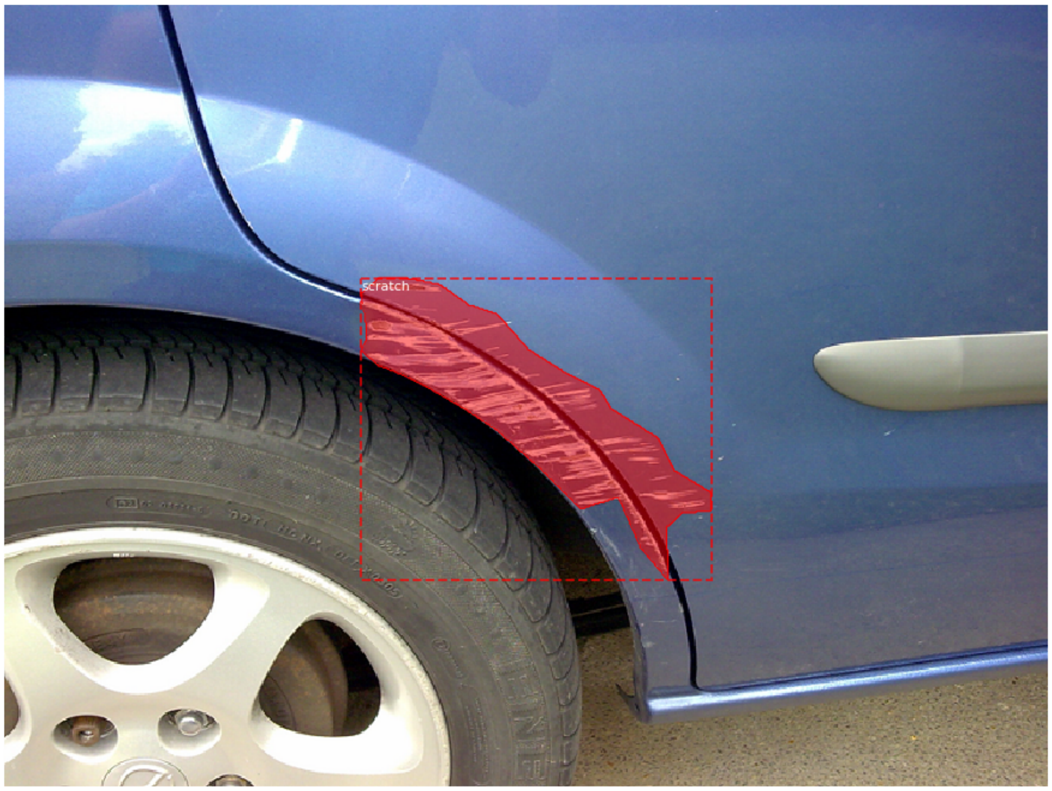
[4]:
```
# Load and display random samples
image_ids = np.random.choice(dataset.image_ids, 5)
for image_id in image_ids:
    image = dataset.load_image(image_id)
    mask, class_ids = dataset.load_mask(image_id)
    visualize.display_top_masks(image, mask, class_ids, dataset.class_names)
```

Make Bounding Box(BB)with annotated damage mask for a typical car image.

[5]:
```
image_id = random.choice(dataset.image_ids)
image = dataset.load_image(image_id)
```

```
mask, class_ids = dataset.load_mask(image_id)
# Compute Bounding box
bbox = utils.extract_bboxes(mask)

# Display image and additional stats
print("image_id ", image_id, dataset.image_reference(image_id))
log("image", image)
log("mask", mask)
log("class_ids", class_ids)
log("bbox", bbox)
# Display image and instances
visualize.display_instances(image, bbox, mask, class_ids, dataset.class_names)
```

```
image_id  41 /home/nasir/Desktop/carcnn/car-damage-detection-using-
CNN/custom/train/image46.png
image                    shape: (528, 705, 3)         min:    0.00000  max:
255.00000  uint8
mask                     shape: (528, 705, 1)         min:    0.00000  max:
1.00000  bool
class_ids                shape: (1,)                  min:    1.00000  max:
1.00000  int32
bbox                     shape: (1, 4)                min:  184.00000  max:
478.00000  int32
```



We see some the components of image annotations. Mainly it has x and y co-ordinate of all labeled damages('polygon') and class name(here 'scratch') for respective car image.

[7]:
```
#Annotation file load
annotations1 = json.load(open(os.path.join(ROOT_DIR, "custom/train/
↪via_region_data.json"),encoding="utf8"))
annotations = list(annotations1.values())
annotations = [a for a in annotations if a['regions']]
annotations[0]
```

[7]:
```
{'fileref': '',
 'size': 46041,
 'filename': 'image2.jpg',
 'base64_img_data': '',
 'file_attributes': {},
 'regions': {'0': {'shape_attributes': {'name': 'polygon',
    'all_points_x': [428,
     429,
     480,
     518,
     557,
     577,
     610,
     660,
     642,
     578,
     579,
     585,
     590,
     574,
     580,
     516,
     507,
     474,
     427,
     426,
     412,
     412,
     430,
     470,
     452,
     428],
    'all_points_y': [232,
     216,
     198,
     193,
      212,
      238,
      237,
      242,
      248,
      248,
      260,
      292,
      343,
      409,
      417,
      441,
      443,
      427,
      413,
      381,
      324,
      301,
      288,
      249,
      231,
      232]},
   'region_attributes': {'Scratch': 'scratch'}},
  '1': {'shape_attributes': {'name': 'polygon',
    'all_points_x': [470, 500, 578, 718, 670, 594, 553, 510, 469, 448, 470],
    'all_points_y': [516, 548, 562, 557, 569, 595, 587, 600, 576, 552, 516]},
   'region_attributes': {'Scratch': 'scratch'}}}}
```

If we have to quantify a car damage,we need to know the x and y coordinates of the polygon to calculate area of the marked/detected damage.This is for 2nd damage polygon of 'image2.jpg'

[8]:
```
annotations[1]['regions']['0']['shape_attributes']
l = []
for d in annotations[1]['regions']['0']['shape_attributes'].values():
    l.append(d)
display('x co-ordinates of the damage:',l[1])
display('y co-ordinates of the damage:',l[2])
```

```
'x co-ordinates of the damage:'


[293, 360, 349, 308, 293]


'y co-ordinates of the damage:'


[303, 330, 314, 302, 303]
```

For prediction or damage detection we need to use the model as inference mode. Model description is consists of important model information like CNN architecture name('resnet101'), ROI threshold(0.9 as defined),configuration description, weightage of different loss components, mask shape, WEIGHT_DECAY etc.

[9]:
```
config = custom.CustomConfig()
ROOT_DIR = "/home/nasir/Desktop/carcnn/car-damage-detection-using-CNN"
CUSTOM_DIR = os.path.join(ROOT_DIR + "/custom/")
print(CUSTOM_DIR)
class InferenceConfig(config.__class__):
    # Run detection on one image at a time
    GPU_COUNT = 1
    IMAGES_PER_GPU = 1

config = InferenceConfig()
config.display()

# Device to load the neural network on.
# Useful if you're training a model on the same
# machine, in which case use CPU and leave the
# GPU for training.
DEVICE = "/cpu:0"  # /cpu:0 or /gpu:0

# Inspect the model in training or inference modes
# values: 'inference' or 'training'
# TODO: code for 'training' test mode not ready yet
TEST_MODE = "inference"
```

```
/home/nasir/Desktop/carcnn/car-damage-detection-using-CNN/custom/

Configurations:
BACKBONE                       resnet101
BACKBONE_STRIDES               [4, 8, 16, 32, 64]
BATCH_SIZE                     1
BBOX_STD_DEV                   [0.1 0.1 0.2 0.2]
COMPUTE_BACKBONE_SHAPE         None
DETECTION_MAX_INSTANCES        100
DETECTION_MIN_CONFIDENCE       0.9
DETECTION_NMS_THRESHOLD        0.3
FPN_CLASSIF_FC_LAYERS_SIZE     1024
GPU_COUNT                      1
GRADIENT_CLIP_NORM             5.0
IMAGES_PER_GPU                 1
IMAGE_CHANNEL_COUNT            3
IMAGE_MAX_DIM                  1024
IMAGE_META_SIZE                14
IMAGE_MIN_DIM                  800
IMAGE_MIN_SCALE                0
```

```
IMAGE_RESIZE_MODE              square
IMAGE_SHAPE                    [1024 1024    3]
LEARNING_MOMENTUM              0.9
LEARNING_RATE                  0.001
LOSS_WEIGHTS                   {'rpn_class_loss': 1.0, 'rpn_bbox_loss': 1.0,
'mrcnn_class_loss': 1.0, 'mrcnn_bbox_loss': 1.0, 'mrcnn_mask_loss': 1.0}
MASK_POOL_SIZE                 14
MASK_SHAPE                     [28, 28]
MAX_GT_INSTANCES               100
MEAN_PIXEL                     [123.7 116.8 103.9]
MINI_MASK_SHAPE                (56, 56)
NAME                           damage
NUM_CLASSES                    2
POOL_SIZE                      7
POST_NMS_ROIS_INFERENCE        1000
POST_NMS_ROIS_TRAINING         2000
PRE_NMS_LIMIT                  6000
ROI_POSITIVE_RATIO             0.33
RPN_ANCHOR_RATIOS              [0.5, 1, 2]
RPN_ANCHOR_SCALES              (32, 64, 128, 256, 512)
RPN_ANCHOR_STRIDE              1
RPN_BBOX_STD_DEV               [0.1 0.1 0.2 0.2]
RPN_NMS_THRESHOLD              0.7
RPN_TRAIN_ANCHORS_PER_IMAGE    256
STEPS_PER_EPOCH                100
TOP_DOWN_PYRAMID_SIZE          256
TRAIN_BN                       False
TRAIN_ROIS_PER_IMAGE           200
USE_MINI_MASK                  True
USE_RPN_ROIS                   True
VALIDATION_STEPS               50
WEIGHT_DECAY                   0.0001
```

Helper function to visualize predicted damage masks and loading the model weights for prediction

[11]:
```python
def get_ax(rows=1, cols=1, size=16):
    """Return a Matplotlib Axes array to be used in
    all visualizations in the notebook. Provide a
    central point to control graph sizes.

    Adjust the size attribute to control how big to render images
    """
    _, ax = plt.subplots(rows, cols, figsize=(size*cols, size*rows))
    return ax
```

```
from importlib import reload # was constantly changin the visualization, so I␣
 ↪decided to reload it instead of notebook
reload(visualize)

# Create model in inference mode
import tensorflow as tf
with tf.device(DEVICE):
    model = modellib.MaskRCNN(mode="inference", model_dir=MODEL_DIR,
                              config=config)

# load the last best model you trained
# weights_path = model.find_last()[1]
custom_WEIGHTS_PATH = '/home/nasir/Desktop/carcnn/
 ↪car-damage-detection-using-CNN/logs/scratch20190822T1650/
 ↪mask_rcnn_scratch_0015.h5'
# Load weights
print("Loading weights ", custom_WEIGHTS_PATH)
model.load_weights(custom_WEIGHTS_PATH, by_name=True)
```

```
Loading weights  /home/nasir/Desktop/carcnn/car-damage-detection-using-
CNN/logs/scratch20190822T1650/mask_rcnn_scratch_0015.h5
Re-starting from epoch 15
```

Loading validation data-set for prediction

[12]:
```
dataset = custom_1.CustomDataset()
dataset.load_custom(CUSTOM_DIR,'val')
dataset.prepare()
print('Images: {}\nclasses: {}'.format(len(dataset.image_ids), dataset.
 ↪class_names))
```

```
Images: 6
classes: ['BG', 'scratch']
```

Visualize model weight matrix descriptive statistics(shapes, histograms)

[13]:
```
visualize.display_weight_stats(model)
```

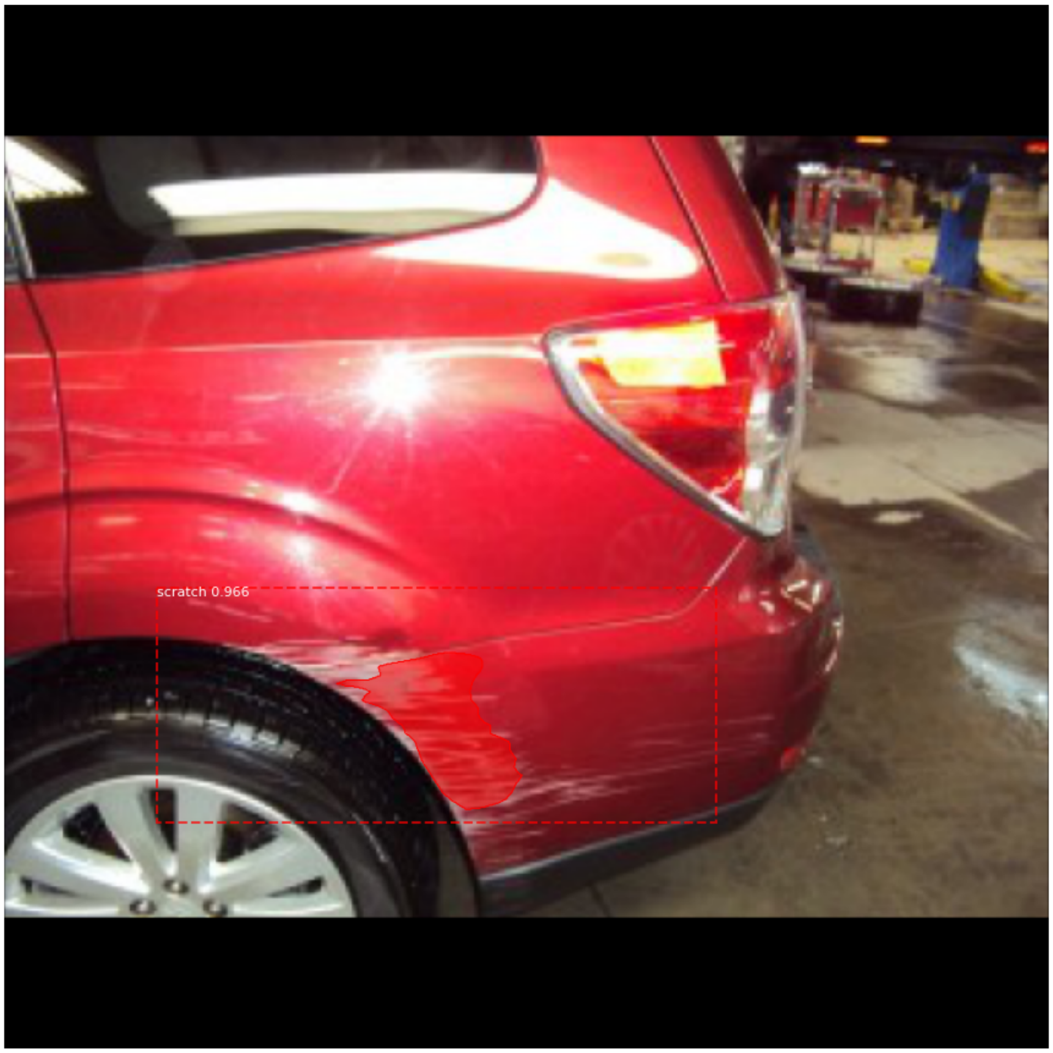
```
<IPython.core.display.HTML object>
```

Prediction on a random validation image(image53.jpeg)

[14]:
```
image_id = random.choice(dataset.image_ids)
image, image_meta, gt_class_id, gt_bbox, gt_mask =\
    modellib.load_image_gt(dataset, config, image_id, use_mini_mask=False)
info = dataset.image_info[image_id]
print("image ID: {}.{} ({}) {}".format(info["source"], info["id"], image_id,
                                               dataset.image_reference(image_id)))

# Run object detection
results = model.detect([image], verbose=1)

# Display results
ax = get_ax(1)
r = results[0]
visualize.display_instances(image, r['rois'], r['masks'], r['class_ids'],
                            dataset.class_names, r['scores'], ax=ax,
                            title="Predictions")
log("gt_class_id", gt_class_id)
log("gt_bbox", gt_bbox)
log("gt_mask", gt_mask)
print('The car has:{} damages'.format(len(dataset.
 ↪image_info[image_id]['polygons'])))
```

```
image ID: scratch.image54.jpg (3) /home/nasir/Desktop/carcnn/car-damage-
detection-using-CNN/custom/val/image54.jpg
Processing 1 images
image                    shape: (1024, 1024, 3)       min:    0.00000  max:
255.00000  uint8
molded_images            shape: (1, 1024, 1024, 3)    min: -123.70000  max:
151.10000  float64
image_metas              shape: (1, 14)               min:    0.00000  max:
1024.00000  int64
anchors                  shape: (1, 261888, 4)        min:   -0.35390  max:
1.29134  float32
gt_class_id              shape: (1,)                  min:    1.00000  max:
1.00000  int32
gt_bbox                  shape: (1, 4)                min:  221.00000  max:
856.00000  int32
gt_mask                  shape: (1024, 1024, 1)       min:    0.00000  max:
1.00000  bool
The car has:1 damages
```

Predictions



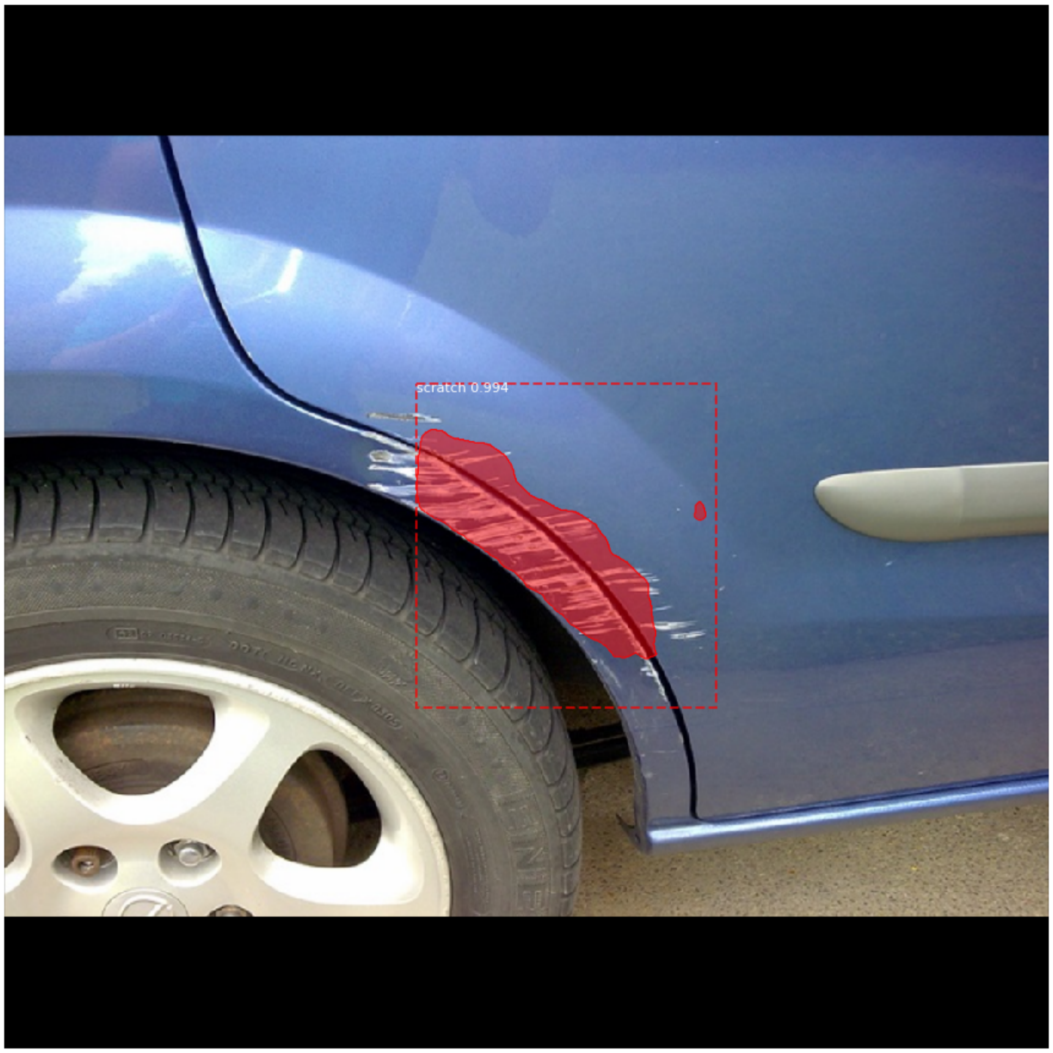
On another image

[15]:
```
image_id = random.choice(dataset.image_ids)
image, image_meta, gt_class_id, gt_bbox, gt_mask =\
    modellib.load_image_gt(dataset, config, image_id, use_mini_mask=False)
info = dataset.image_info[image_id]
print("image ID: {}.{} ({}) {}".format(info["source"], info["id"], image_id,
                                       dataset.image_reference(image_id)))

# Run object detection
results = model.detect([image], verbose=1)

# Display results
```

```
ax = get_ax(1)
r = results[0]
visualize.display_instances(image, r['rois'], r['masks'], r['class_ids'],
                            dataset.class_names, r['scores'], ax=ax,
                            title="Predictions")
log("gt_class_id", gt_class_id)
log("gt_bbox", gt_bbox)
log("gt_mask", gt_mask)
print('The car has:{} damages'.format(len(dataset.
 ↪image_info[image_id]['polygons'])))
```

```
image ID: scratch.image51.png (0) /home/nasir/Desktop/carcnn/car-damage-
detection-using-CNN/custom/val/image51.png
Processing 1 images
image                    shape: (1024, 1024, 3)       min:    0.00000  max:
255.00000  uint8
molded_images            shape: (1, 1024, 1024, 3)    min: -123.70000  max:
151.10000  float64
image_metas              shape: (1, 14)               min:    0.00000  max:
1024.00000  int64
anchors                  shape: (1, 261888, 4)        min:   -0.35390  max:
1.29134  float32
gt_class_id              shape: (1,)                  min:    1.00000  max:
1.00000  int32
gt_bbox                  shape: (1, 4)                min:  351.00000  max:
689.00000  int32
gt_mask                  shape: (1024, 1024, 1)       min:    0.00000  max:
1.00000  bool
The car has:1 damages
```

Predictions



Good prediction considering training with only 49 images and 15 epochs.
Thanks!

[ ]: