Mask R-CNN - Visualize Trash detection

In [31]:
```
import os
import sys
import random
import math
import re
import time
import glob
import skimage
import numpy as np
import tensorflow as tf
import matplotlib
import matplotlib.pyplot as plt
import matplotlib.patches as patches


ROOT_DIR = os.getcwd()
print(ROOT_DIR)

# Import Mask RCNN
sys.path.append(ROOT_DIR)
from mrcnn import utils
from mrcnn import visualize
from mrcnn.visualize import display_images
import mrcnn.model as modellib
from mrcnn.model import log

from trash import trash

%matplotlib inline

# Directory to save logs and trained model
MODEL_DIR = os.path.join(ROOT_DIR, "logs")

# Path to Trash trained weights
TRASH_WEIGHTS_PATH = "weights/mask_rcnn_trash_0200_030519_large.h5" #the best

print('Weights being used: ', TRASH_WEIGHTS_PATH)
```

```
C:\Users\91740\Desktop\wade-ai-master\Trash_Detection
Weights being used:  weights/mask_rcnn_trash_0200_030519_large.h5
```

Configurations

In [32]:
```
config = trash.TrashConfig()
TRASH_DIR = 'trash'
TRASH_DIR
```

Out[32]:
```
'trash'
```

In [33]:
```
# Override the training configurations with a few
# changes for inferencing.
class InferenceConfig(config.__class__):
    # Run detection on one image at a time
    GPU_COUNT = 1
```

```
    IMAGES_PER_GPU = 1

config = InferenceConfig()
config.display()
```

```
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       0.95
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                  1024
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                           trash
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                17
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
```

Notebook Preferences

In [34]:

```
# 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"
```

In [35]:
```
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
```

Load Validation Dataset

In [36]:
```
# Load validation dataset
dataset = trash.TrashDataset()
dataset.load_trash(TRASH_DIR, "val")

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

print("Images: {}\nClasses: {}".format(len(dataset.image_ids), dataset.class_names))
```

```
Images: 0
Classes: ['BG', 'trash']
```

Load Model

In [37]:
```
# Create model in inference mode
with tf.device(DEVICE):
    model = modellib.MaskRCNN(mode="inference", model_dir=MODEL_DIR,config=config)
```

In [38]:
```
# Load the weights you trained
weights_path = os.path.join(ROOT_DIR, TRASH_WEIGHTS_PATH)
model.load_weights(weights_path, by_name=True)
print("Loading weights ", TRASH_WEIGHTS_PATH)
```

```
Loading weights  weights/mask_rcnn_trash_0200_030519_large.h5
```

Select test images

In [39]:
```
# Get images from the directory of all the test images
jpg = glob.glob("images/*.jpg")
jpeg = glob.glob("images/*.jpeg")
jpg.extend(jpeg)
jpg
```

Out[39]:
```
['images\\319.jpg',
 'images\\roadside_garbage.jpg',
 'images\\sample_test.jpg',
 'images\\urban_cctv_garbage.jpg']
```

Run detection on images

In [40]:

```
for image in jpg:
    print(image)
    image = skimage.io.imread('{}'.format(image))

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

    print("-------------------------------------------")
    print("Number of Trash elements detected: ", len(r['scores']))
    for j in range(len(r['scores'])):
        print("Element",j,r['scores'][j])
    print("-------------------------------------------")

    # 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")
```

```
images\319.jpg
Processing 1 images
image                    shape: (640, 640, 3)         min:    0.00000  max:  255.000
00  uint8
molded_images            shape: (1, 1024, 1024, 3)    min: -123.70000  max:  147.100
00  float64
image_metas              shape: (1, 14)               min:    0.00000  max: 1024.000
00  float64
anchors                  shape: (1, 261888, 4)        min:   -0.35390  max:    1.291
34  float32
-------------------------------------------
Number of Trash elements detected:  1
Element 0 0.99703836
-------------------------------------------

*** No instances to display ***

images\roadside_garbage.jpg
Processing 1 images
image                    shape: (1334, 2000, 3)       min:    0.00000  max:  255.000
00  uint8
molded_images            shape: (1, 1024, 1024, 3)    min: -123.70000  max:  151.100
00  float64
image_metas              shape: (1, 14)               min:    0.00000  max: 2000.000
00  float64
anchors                  shape: (1, 261888, 4)        min:   -0.35390  max:    1.291
34  float32
-------------------------------------------
Number of Trash elements detected:  0
-------------------------------------------
images\sample_test.jpg
Processing 1 images
image                    shape: (1907, 4031, 3)       min:    0.00000  max:  255.000
00  uint8
molded_images            shape: (1, 1024, 1024, 3)    min: -123.70000  max:  151.100
00  float64
image_metas              shape: (1, 14)               min:    0.00000  max: 4031.000
00  float64
anchors                  shape: (1, 261888, 4)        min:   -0.35390  max:    1.291
34  float32
-------------------------------------------
Number of Trash elements detected:  1
Element 0 0.98225456
```

```
------------------------------------------------
images\urban_cctv_garbage.jpg
Processing 1 images
image                    shape: (533, 800, 3)         min:    0.00000  max:  255.000
00  uint8
molded_images            shape: (1, 1024, 1024, 3)    min: -123.70000  max:  151.100
00  float64
image_metas              shape: (1, 14)               min:    0.00000  max: 1024.000
00  float64
anchors                  shape: (1, 261888, 4)        min:   -0.35390  max:    1.291
34  float32
------------------------------------------------
Number of Trash elements detected:  4
Element 0 0.9974147
Element 1 0.99541014
Element 2 0.99172735
Element 3 0.9753823
------------------------------------------------
```

Predictions

Predictions

Predictions

Predictions

In [ ]: