Capstone 3: Using Mask R-CNN for Weapons Identification
Author: Nantawat Samermit

Model Metrics:

Data set:

- Total size: 1470 images
- Train set: 999 images
- Test set: 471 images
- Annotation format: PASCAL VOC using .xml

CNN Model:

- Model Used: Mask Regional-Convolutional Neural Network (R-CNN)
- Configuration name: 'weapons_faster_cfg'
- Number of classes 1 + 3 (background + number of classes)
- Steps per epoch: 489
- Code:

```
from mrcnn.config import Config
from mrcnn.model import MaskRCNN
#define a configuration for the model
class WeaponsConfig(Config):
  #give config a recognizable name
  NAME='weapons_faster_cfg'
  #Number of classes (background + knife + cash + pistol)
  NUM_CLASSES = 1 + 3
  #Number of training steps per epoch
  STEPS_PER_EPOCH = 489
```

- Configuration:
    - Backbone: Resnet101
    - Strides: [4, 8, 16, 32, 64]
    - Batch_size: 2
    - BBOX_STD_DEV: [0.1, 0.1, 0.2, 0.2]
    - 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
    - Image:
        - Per GPU: 2
        - Max Dim: 1024
        - Meta Size: 16

        - Min Dim: 800
        - Min Scale: 0
        - Resize Mode: square
        - Shape: 1024 x 1024 x 3
    - Mask:
        - Pool Size: 14
        - Shape: 28 x 28
        - Mean Pixel: 123.7 x 116.8 x 103.9
        - Mini Mask Shape: 56 x 56
- Weights:
    - MSCOCO: ‘mask_rcnn_coco.h5’
- Training:
    - Train set: 999 images
    - Test Set: 471 images
    - Learning Rate: 0.001
    - Epochs: 10
    - Layers: ‘heads’

Evaluation Metrics:

- Metric Used: Mean Average Precision
- Training MAP: 0.645695
- Testing MAP: 0.149661

- Code:

```
def evaluate_model(dataset, model, cfg):
    APs = list()
    for image_id in dataset.image_ids:
        # load image, bounding boxes and masks for the image id.
        # load_image_gt() returns np.ndarrays

        image, image_meta, gt_class_id, gt_bbox, gt_mask = load_image_gt(dataset,
                                                                         cfg,
                                                                         image_id,
                                                                         use_mini_mask=False)


        # convert pixel values (e.g. center)
        scaled_image = mold_image(image, cfg)

        # convert image into one sample
        sample = expand_dims(scaled_image, 0)

        # make prediction
        yhat = model.detect(sample, verbose=0)

        # extract results for first sample
        r = yhat[0]

        # calculate statistics, including AP
        #when using on training data, since it may not have an image of a weapon
        #the prediction becomes an empty list, this will throw a ValueError
        #Using try, except control flow to circumvent
        try:
          AP, precision, recalls, overlaps = compute_ap(gt_bbox, gt_class_id,
                                                        gt_mask, r["rois"],
                                                        r["class_ids"],
                                                        r["scores"],
                                                        r['masks'])
          APs.append(AP)
        except ValueError:
          continue
    return mean(APs)
```