Maskrcnn_seven_class5 (3)

December 31, 2019

[4]:
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Go to this URL in a browser: https://accounts.google.com/o/oauth2/auth?client_id
=947318989803-6bn6qk8qdgf4n4g3pfee6491hc0brc4i.apps.googleusercontent.com&redire
ct_uri=urn%3aietf%3awg%3aoauth%3a2.0%3aoob&response_type=code&scope=email%20http
s%3a%2f%2fwww.googleapis.com%2fauth%2fdocs.test%20https%3a%2f%2fwww.googleapis.c
om%2fauth%2fdrive%20https%3a%2f%2fwww.googleapis.com%2fauth%2fdrive.photos.reado
nly%20https%3a%2f%2fwww.googleapis.com%2fauth%2fpeopleapi.readonly

Enter your authorization code:
··········
Mounted at /content/drive
```

[0]:
```
import os
import sys
import json
import numpy as np
import time
from PIL import Image, ImageDraw
import tensorflow.compat.v1 as tf
tf.disable_v2_behavior()
```

[6]:
```
# Set the ROOT_DIR variable to the root directory of the Mask_RCNN git repo
ROOT_DIR = '/content/drive/My Drive/'
assert os.path.exists(ROOT_DIR), 'ROOT_DIR does not exist. Did you forget to␣
 ↪read the instructions above? ;)'

# Import mrcnn libraries
sys.path.append(ROOT_DIR)
from mrcnn.config import Config
import mrcnn.utils as utils
from mrcnn import visualize
import mrcnn.model as modellib
```

```
Using TensorFlow backend.
```

[0]:
```
# Directory to save logs and trained model
MODEL_DIR = os.path.join(ROOT_DIR, "cervic_logs")

# Local path to trained weights file
# COCO_MODEL_PATH = os.path.join(ROOT_DIR, "mask_rcnn_cig_butts_0008.h5")


COCO_MODEL_PATH = os.path.join(ROOT_DIR, "mask_rcnn_coco.h5")
# Download COCO trained weights from Releases if needed
if not os.path.exists(COCO_MODEL_PATH):
    utils.download_trained_weights(COCO_MODEL_PATH)
```

[8]:
```
print(MODEL_DIR )
print(COCO_MODEL_PATH)
```

```
/content/drive/My Drive/cervic_logs
/content/drive/My Drive/mask_rcnn_coco.h5
```

[9]:
```
class Cervic_seven_classConfig(Config):
    """Configuration for training on the cigarette butts dataset.
    Derives from the base Config class and overrides values specific
    to the cigarette butts dataset.
    """
    # Give the configuration a recognizable name
    NAME = "Cervic_seven_class"

    # Train on 1 GPU and 1 image per GPU. Batch sizoure is 1 (GPUs * images/
 ↪GPU).
    GPU_COUNT = 1
    IMAGES_PER_GPU = 1

    # Number of classes (including background)
    NUM_CLASSES = 1 + 7  # background + 1 (cig_butt)

    # All of our training images are 512x512
    IMAGE_MIN_DIM = 512
    IMAGE_MAX_DIM = 512

    # You can experiment with this number to see if it improves training
    STEPS_PER_EPOCH = 500
    LEARNING_RATE= 5e-4
    # This is how often validation is run. If you are using too much hard drive␣
 ↪space
    # on saved models (in the MODEL_DIR), try making this value larger.
    VALIDATION_STEPS = 5

    # Matterport originally used resnet101, but I downsized to fit it on my␣
↪graphics card
    BACKBONE = 'resnet50'

    # To be honest, I haven't taken the time to figure out what these do
    RPN_ANCHOR_SCALES = (8, 16, 32, 64, 128)
    TRAIN_ROIS_PER_IMAGE = 32
    MAX_GT_INSTANCES = 50
    POST_NMS_ROIS_INFERENCE = 500
    POST_NMS_ROIS_TRAINING = 1000

config = Cervic_seven_classConfig()
config.display()
```

```
Configurations:
BACKBONE                       resnet50
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       0.7
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                  512
IMAGE_META_SIZE                20
IMAGE_MIN_DIM                  512
IMAGE_MIN_SCALE                0
IMAGE_RESIZE_MODE              square
IMAGE_SHAPE                    [512 512   3]
LEARNING_MOMENTUM              0.9
LEARNING_RATE                  0.0005
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               50
MEAN_PIXEL                     [123.7 116.8 103.9]
MINI_MASK_SHAPE                (56, 56)
NAME                           Cervic_seven_class
NUM_CLASSES                    8
POOL_SIZE                      7
```

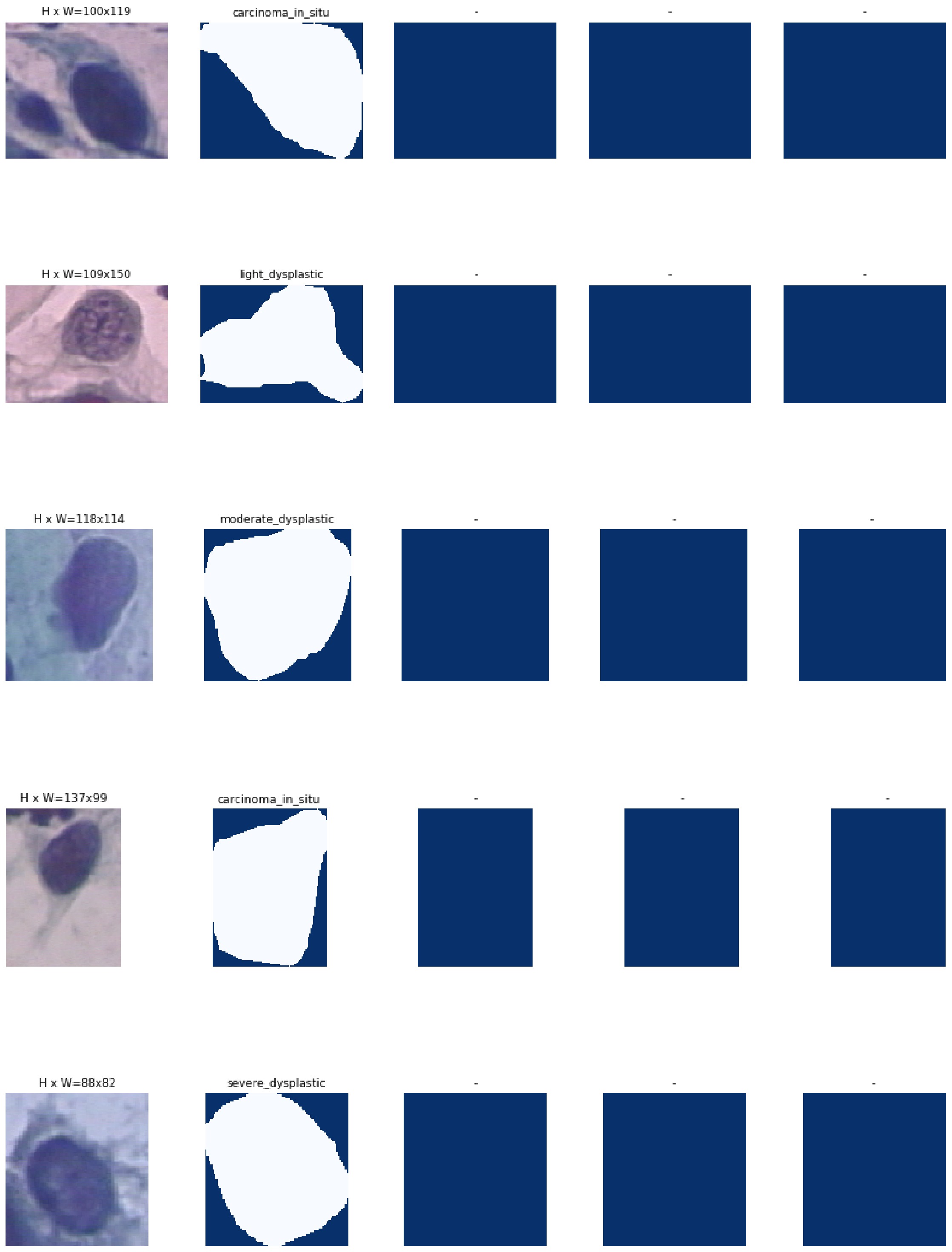
```
POST_NMS_ROIS_INFERENCE        500
POST_NMS_ROIS_TRAINING         1000
PRE_NMS_LIMIT                  6000
ROI_POSITIVE_RATIO             0.33
RPN_ANCHOR_RATIOS              [0.5, 1, 2]
RPN_ANCHOR_SCALES              (8, 16, 32, 64, 128)
RPN_ANCHOR_STRIDE              1
RPN_BBOX_STD_DEV               [0.1 0.1 0.2 0.2]
RPN_NMS_THRESHOLD              0.7
RPN_TRAIN_ANCHORS_PER_IMAGE    256
STEPS_PER_EPOCH                500
TOP_DOWN_PYRAMID_SIZE          256
TRAIN_BN                       False
TRAIN_ROIS_PER_IMAGE           32
USE_MINI_MASK                  True
USE_RPN_ROIS                   True
VALIDATION_STEPS               5
WEIGHT_DECAY                   0.0001
```

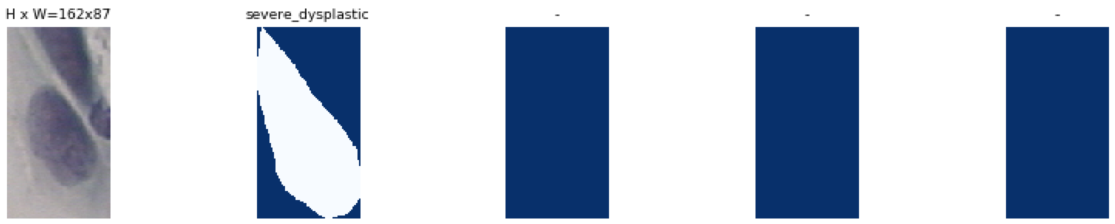
[0]:

```python
class CocoLikeDataset(utils.Dataset):
    """ Generates a COCO-like dataset, i.e. an image dataset annotated in the␣
↪style of the COCO dataset.
        See http://cocodataset.org/#home for more information.
    """
    def load_data(self, annotation_json, images_dir):
        """ Load the coco-like dataset from json
        Args:
            annotation_json: The path to the coco annotations json file
            images_dir: The directory holding the images referred to by the␣
↪json file
        """
        # Load json from file
        json_file = open(annotation_json)
        coco_json = json.load(json_file)
        json_file.close()

        # Add the class names using the base method from utils.Dataset
        source_name = "coco_like"
        for category in coco_json['categories']:
            class_id = category['category_id']
            # class_id = 4
            class_name = category['name']
            # class_name = 'Severe_dysplastic'
            if class_id < 1:
                print('Error: Class id for "{}" cannot be less than one. (0 is␣
↪reserved for the background)'.format(class_name))
                return

            self.add_class(source_name, class_id, class_name)

        # Get all annotations
        annotations = {}
        for annotation in coco_json['annotations']:
            image_id = annotation['image_id']
            if image_id not in annotations:
                annotations[image_id] = []
            annotations[image_id].append(annotation)

        # Get all images and add them to the dataset
        seen_images = {}
        for image in coco_json['images']:
            image_id = image['id']
            if image_id in seen_images:
                print("Warning: Skipping duplicate image id: {}".format(image))
            else:
                seen_images[image_id] = image
                try:
                    image_file_name = image['filename']
                    image_width = image['width']
                    image_height = image['height']
                except KeyError as key:
                    print("Warning: Skipping image (id: {}) with missing key:␣
↪{}".format(image_id, key))

                image_path = os.path.abspath(os.path.join(images_dir,␣
↪image_file_name))
                image_annotations = annotations[image_id]

                # Add the image using the base method from utils.Dataset
                self.add_image(
                    source=source_name,
                    image_id=image_id,
                    path=image_path,
                    width=image_width,
                    height=image_height,
                    annotations=image_annotations
                )

    def load_mask(self, image_id):
        """ Load instance masks for the given image.
        MaskRCNN expects masks in the form of a bitmap [height, width,␣
↪instances].
        Args:
            image_id: The id of the image to load masks for
        Returns:
            masks: A bool array of shape [height, width, instance count] with
                one mask per instance.
            class_ids: a 1D array of class IDs of the instance masks.
        """
        image_info = self.image_info[image_id]
        annotations = image_info['annotations']
        instance_masks = []
        class_ids = []

        for annotation in annotations:
            class_id = annotation['category_id']
            mask = Image.new('1', (image_info['width'], image_info['height']))
            mask_draw = ImageDraw.ImageDraw(mask, '1')
            for segmentation in annotation['segmentation']:
                mask_draw.polygon(segmentation, fill=1)
                bool_array = np.array(mask) > 0
                instance_masks.append(bool_array)
                class_ids.append(class_id)

        mask = np.dstack(instance_masks)
        class_ids = np.array(class_ids, dtype=np.int32)

        return mask, class_ids
```

[0]:
```
dataset_train = CocoLikeDataset()
dataset_train.load_data('/content/drive/My Drive/cervic_train/
↪cervic_all_class_train1.json', '/content/drive/My Drive/')
dataset_train.prepare()

dataset_val = CocoLikeDataset()
dataset_val.load_data('/content/drive/My Drive/cervic_validation/
↪cervic_all_class_validation1.json', '/content/drive/My Drive/')
dataset_val.prepare()
```

[12]:
```
dataset = dataset_train
image_ids = np.random.choice(dataset.image_ids,6)
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)
```


H x W=100x119
carcinoma_in_situ
-
-
-
H x W=109x150
light_dysplastic
-
-
-
H x W=118x114
moderate_dysplastic
-
-
-
H x W=137x99
carcinoma_in_situ
-
-
-
H x W=88x82
severe_dysplastic
-
-
-



[13]:
```python
# Create model in training mode
model = modellib.MaskRCNN(mode="training", config=config,
                          model_dir=MODEL_DIR)
```

```
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/keras/backend/tensorflow_backend.py:541: The name tf.placeholder is
deprecated. Please use tf.compat.v1.placeholder instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/keras/backend/tensorflow_backend.py:66: The name tf.get_default_graph
is deprecated. Please use tf.compat.v1.get_default_graph instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/keras/backend/tensorflow_backend.py:4432: The name tf.random_uniform is
deprecated. Please use tf.random.uniform instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/keras/backend/tensorflow_backend.py:2139: The name
tf.nn.fused_batch_norm is deprecated. Please use
tf.compat.v1.nn.fused_batch_norm instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/keras/backend/tensorflow_backend.py:4267: The name tf.nn.max_pool is
deprecated. Please use tf.nn.max_pool2d instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/keras/backend/tensorflow_backend.py:2239: The name
tf.image.resize_nearest_neighbor is deprecated. Please use
tf.compat.v1.image.resize_nearest_neighbor instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/tensorflow_core/python/ops/array_ops.py:1475: where (from
tensorflow.python.ops.array_ops) is deprecated and will be removed in a future
version.
Instructions for updating:
Use tf.where in 2.0, which has the same broadcast rule as np.where
WARNING:tensorflow:From /content/drive/My Drive/mrcnn/model.py:553: The name
tf.random_shuffle is deprecated. Please use tf.random.shuffle instead.
```

```
WARNING:tensorflow:From /content/drive/My Drive/mrcnn/utils.py:202: The name
tf.log is deprecated. Please use tf.math.log instead.

WARNING:tensorflow:From /content/drive/My Drive/mrcnn/model.py:600: calling
crop_and_resize_v1 (from tensorflow.python.ops.image_ops_impl) with box_ind is
deprecated and will be removed in a future version.
Instructions for updating:
box_ind is deprecated, use box_indices instead
```

[14]:
```
# Which weights to start with?
# init_with = "coco"  # imagenet, coco, or last
init_with = "imagenet"
if init_with == "imagenet":
    model.load_weights(model.get_imagenet_weights(), by_name=True)
elif init_with == "coco":
    # Load weights trained on MS COCO, but skip layers that
    # are different due to the different number of classes
    # See README for instructions to download the COCO weights
    model.load_weights(COCO_MODEL_PATH, by_name=True,
                       exclude=["mrcnn_class_logits", "mrcnn_bbox_fc",
                                "mrcnn_bbox", "mrcnn_mask"])
elif init_with == "last":
    # Load the last model you trained and continue training
    model.load_weights(model.find_last(), by_name=True)
```

```
Downloading data from https://github.com/fchollet/deep-learning-models/releases/
download/v0.2/resnet50_weights_tf_dim_ordering_tf_kernels_notop.h5
94658560/94653016 [==============================] - 1s 0us/step
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/keras/backend/tensorflow_backend.py:190: The name
tf.get_default_session is deprecated. Please use
tf.compat.v1.get_default_session instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/keras/backend/tensorflow_backend.py:197: The name tf.ConfigProto is
deprecated. Please use tf.compat.v1.ConfigProto instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/keras/backend/tensorflow_backend.py:203: The name tf.Session is
deprecated. Please use tf.compat.v1.Session instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/keras/backend/tensorflow_backend.py:207: The name tf.global_variables
is deprecated. Please use tf.compat.v1.global_variables instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/keras/backend/tensorflow_backend.py:216: The name
```

```
tf.is_variable_initialized is deprecated. Please use
tf.compat.v1.is_variable_initialized instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/keras/backend/tensorflow_backend.py:223: The name
tf.variables_initializer is deprecated. Please use
tf.compat.v1.variables_initializer instead.
```

[15]:
```
# Train the head branches
# Passing layers="heads" freezes all layers except the head
# layers. You can also pass a regular expression to select
# which layers to train by name pattern.
start_train = time.time()
model.train(dataset_train, dataset_val,
            learning_rate=config.LEARNING_RATE,
            epochs=20,
            layers='heads')
end_train = time.time()
minutes = round((end_train - start_train) / 60, 2)
print(f'Training took {minutes} minutes')
```

```
Starting at epoch 0. LR=0.0005

Checkpoint Path: /content/drive/My Drive/cervic_logs/cervic_seven_class20191231T
0757/mask_rcnn_cervic_seven_class_{epoch:04d}.h5
Selecting layers to train
fpn_c5p5               (Conv2D)
fpn_c4p4               (Conv2D)
fpn_c3p3               (Conv2D)
fpn_c2p2               (Conv2D)
fpn_p5                 (Conv2D)
fpn_p2                 (Conv2D)
fpn_p3                 (Conv2D)
fpn_p4                 (Conv2D)
In model:  rpn_model
    rpn_conv_shared        (Conv2D)
    rpn_class_raw          (Conv2D)
    rpn_bbox_pred          (Conv2D)
mrcnn_mask_conv1       (TimeDistributed)
mrcnn_mask_bn1         (TimeDistributed)
mrcnn_mask_conv2       (TimeDistributed)
mrcnn_mask_bn2         (TimeDistributed)
mrcnn_class_conv1      (TimeDistributed)
mrcnn_class_bn1        (TimeDistributed)
mrcnn_mask_conv3       (TimeDistributed)
mrcnn_mask_bn3         (TimeDistributed)
```

```
mrcnn_class_conv2      (TimeDistributed)
mrcnn_class_bn2        (TimeDistributed)
mrcnn_mask_conv4       (TimeDistributed)
mrcnn_mask_bn4         (TimeDistributed)
mrcnn_bbox_fc          (TimeDistributed)
mrcnn_mask_deconv      (TimeDistributed)
mrcnn_class_logits     (TimeDistributed)
mrcnn_mask             (TimeDistributed)
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/keras/optimizers.py:793: The name tf.train.Optimizer is deprecated.
Please use tf.compat.v1.train.Optimizer instead.


/usr/local/lib/python3.6/dist-
packages/tensorflow_core/python/framework/indexed_slices.py:424: UserWarning:
Converting sparse IndexedSlices to a dense Tensor of unknown shape. This may
consume a large amount of memory.
  "Converting sparse IndexedSlices to a dense Tensor of unknown shape. "
/usr/local/lib/python3.6/dist-
packages/tensorflow_core/python/framework/indexed_slices.py:424: UserWarning:
Converting sparse IndexedSlices to a dense Tensor of unknown shape. This may
consume a large amount of memory.
  "Converting sparse IndexedSlices to a dense Tensor of unknown shape. "
/usr/local/lib/python3.6/dist-
packages/tensorflow_core/python/framework/indexed_slices.py:424: UserWarning:
Converting sparse IndexedSlices to a dense Tensor of unknown shape. This may
consume a large amount of memory.
  "Converting sparse IndexedSlices to a dense Tensor of unknown shape. "

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/keras/backend/tensorflow_backend.py:1033: The name tf.assign_add is
deprecated. Please use tf.compat.v1.assign_add instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/keras/backend/tensorflow_backend.py:1020: The name tf.assign is
deprecated. Please use tf.compat.v1.assign instead.


/usr/local/lib/python3.6/dist-packages/keras/engine/training_generator.py:49:
UserWarning: Using a generator with `use_multiprocessing=True` and multiple
workers may duplicate your data. Please consider using the `keras.utils.Sequence
class.
  UserWarning('Using a generator with `use_multiprocessing=True`'

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/keras/callbacks.py:1122: The name tf.summary.merge_all is deprecated.
Please use tf.compat.v1.summary.merge_all instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/keras/callbacks.py:1125: The name tf.summary.FileWriter is deprecated.
```

```
Please use tf.compat.v1.summary.FileWriter instead.

Epoch 1/20
500/500 [==============================] - 134s 269ms/step - loss: 4.0954 -
rpn_class_loss: 0.0622 - rpn_bbox_loss: 2.3489 - mrcnn_class_loss: 0.5738 -
mrcnn_bbox_loss: 0.4957 - mrcnn_mask_loss: 0.6148 - val_loss: 2.0337 -
val_rpn_class_loss: 0.0147 - val_rpn_bbox_loss: 0.9678 - val_mrcnn_class_loss:
0.2840 - val_mrcnn_bbox_loss: 0.3038 - val_mrcnn_mask_loss: 0.4633
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-
packages/keras/callbacks.py:1265: The name tf.Summary is deprecated. Please use
tf.compat.v1.Summary instead.

Epoch 2/20
500/500 [==============================] - 103s 206ms/step - loss: 2.5473 -
rpn_class_loss: 0.0233 - rpn_bbox_loss: 1.2546 - mrcnn_class_loss: 0.4810 -
mrcnn_bbox_loss: 0.2867 - mrcnn_mask_loss: 0.5016 - val_loss: 2.4901 -
val_rpn_class_loss: 0.0126 - val_rpn_bbox_loss: 0.8211 - val_mrcnn_class_loss:
0.7781 - val_mrcnn_bbox_loss: 0.3889 - val_mrcnn_mask_loss: 0.4894
Epoch 3/20
500/500 [==============================] - 109s 219ms/step - loss: 1.9999 -
rpn_class_loss: 0.0173 - rpn_bbox_loss: 0.9498 - mrcnn_class_loss: 0.3921 -
mrcnn_bbox_loss: 0.2048 - mrcnn_mask_loss: 0.4359 - val_loss: 2.0480 -
val_rpn_class_loss: 0.0126 - val_rpn_bbox_loss: 0.9769 - val_mrcnn_class_loss:
0.2942 - val_mrcnn_bbox_loss: 0.1474 - val_mrcnn_mask_loss: 0.6169
Epoch 4/20
500/500 [==============================] - 94s 189ms/step - loss: 1.6074 -
rpn_class_loss: 0.0142 - rpn_bbox_loss: 0.6948 - mrcnn_class_loss: 0.3453 -
mrcnn_bbox_loss: 0.1524 - mrcnn_mask_loss: 0.4008 - val_loss: 1.9176 -
val_rpn_class_loss: 0.0104 - val_rpn_bbox_loss: 0.6169 - val_mrcnn_class_loss:
0.5143 - val_mrcnn_bbox_loss: 0.2485 - val_mrcnn_mask_loss: 0.5275
Epoch 5/20
500/500 [==============================] - 95s 190ms/step - loss: 1.4942 -
rpn_class_loss: 0.0135 - rpn_bbox_loss: 0.6207 - mrcnn_class_loss: 0.3200 -
mrcnn_bbox_loss: 0.1323 - mrcnn_mask_loss: 0.4077 - val_loss: 1.4026 -
val_rpn_class_loss: 0.0160 - val_rpn_bbox_loss: 0.6563 - val_mrcnn_class_loss:
0.3307 - val_mrcnn_bbox_loss: 0.1078 - val_mrcnn_mask_loss: 0.2917
Epoch 6/20
500/500 [==============================] - 95s 190ms/step - loss: 1.3252 -
rpn_class_loss: 0.0134 - rpn_bbox_loss: 0.5228 - mrcnn_class_loss: 0.2970 -
mrcnn_bbox_loss: 0.1048 - mrcnn_mask_loss: 0.3872 - val_loss: 1.3212 -
val_rpn_class_loss: 0.0236 - val_rpn_bbox_loss: 0.4646 - val_mrcnn_class_loss:
0.3977 - val_mrcnn_bbox_loss: 0.1247 - val_mrcnn_mask_loss: 0.3106
Epoch 7/20
500/500 [==============================] - 95s 190ms/step - loss: 1.2205 -
rpn_class_loss: 0.0127 - rpn_bbox_loss: 0.4830 - mrcnn_class_loss: 0.2688 -
mrcnn_bbox_loss: 0.1022 - mrcnn_mask_loss: 0.3539 - val_loss: 1.5851 -
val_rpn_class_loss: 0.0089 - val_rpn_bbox_loss: 0.5821 - val_mrcnn_class_loss:
0.3762 - val_mrcnn_bbox_loss: 0.2059 - val_mrcnn_mask_loss: 0.4120
```

```
Epoch 8/20
500/500 [==============================] - 95s 189ms/step - loss: 1.1591 -
rpn_class_loss: 0.0113 - rpn_bbox_loss: 0.4258 - mrcnn_class_loss: 0.2649 -
mrcnn_bbox_loss: 0.0965 - mrcnn_mask_loss: 0.3605 - val_loss: 1.3826 -
val_rpn_class_loss: 0.0061 - val_rpn_bbox_loss: 0.5371 - val_mrcnn_class_loss:
0.3687 - val_mrcnn_bbox_loss: 0.1073 - val_mrcnn_mask_loss: 0.3635
Epoch 9/20
500/500 [==============================] - 95s 190ms/step - loss: 1.0410 -
rpn_class_loss: 0.0104 - rpn_bbox_loss: 0.3910 - mrcnn_class_loss: 0.2135 -
mrcnn_bbox_loss: 0.0879 - mrcnn_mask_loss: 0.3381 - val_loss: 1.7972 -
val_rpn_class_loss: 0.0151 - val_rpn_bbox_loss: 0.8911 - val_mrcnn_class_loss:
0.4202 - val_mrcnn_bbox_loss: 0.1669 - val_mrcnn_mask_loss: 0.3039
Epoch 10/20
500/500 [==============================] - 95s 190ms/step - loss: 1.0067 -
rpn_class_loss: 0.0107 - rpn_bbox_loss: 0.3545 - mrcnn_class_loss: 0.2365 -
mrcnn_bbox_loss: 0.0815 - mrcnn_mask_loss: 0.3234 - val_loss: 1.2262 -
val_rpn_class_loss: 0.0056 - val_rpn_bbox_loss: 0.4851 - val_mrcnn_class_loss:
0.2992 - val_mrcnn_bbox_loss: 0.1335 - val_mrcnn_mask_loss: 0.3028
Epoch 11/20
500/500 [==============================] - 94s 189ms/step - loss: 0.8978 -
rpn_class_loss: 0.0104 - rpn_bbox_loss: 0.3133 - mrcnn_class_loss: 0.1941 -
mrcnn_bbox_loss: 0.0688 - mrcnn_mask_loss: 0.3112 - val_loss: 1.6351 -
val_rpn_class_loss: 0.0247 - val_rpn_bbox_loss: 0.4569 - val_mrcnn_class_loss:
0.6768 - val_mrcnn_bbox_loss: 0.0973 - val_mrcnn_mask_loss: 0.3794
Epoch 12/20
500/500 [==============================] - 95s 189ms/step - loss: 0.8852 -
rpn_class_loss: 0.0100 - rpn_bbox_loss: 0.3084 - mrcnn_class_loss: 0.1900 -
mrcnn_bbox_loss: 0.0719 - mrcnn_mask_loss: 0.3049 - val_loss: 1.5411 -
val_rpn_class_loss: 0.0180 - val_rpn_bbox_loss: 0.4866 - val_mrcnn_class_loss:
0.6242 - val_mrcnn_bbox_loss: 0.1031 - val_mrcnn_mask_loss: 0.3091
Epoch 13/20
500/500 [==============================] - 94s 188ms/step - loss: 0.8619 -
rpn_class_loss: 0.0084 - rpn_bbox_loss: 0.2894 - mrcnn_class_loss: 0.2089 -
mrcnn_bbox_loss: 0.0691 - mrcnn_mask_loss: 0.2860 - val_loss: 0.9672 -
val_rpn_class_loss: 0.0187 - val_rpn_bbox_loss: 0.5200 - val_mrcnn_class_loss:
0.0644 - val_mrcnn_bbox_loss: 0.1276 - val_mrcnn_mask_loss: 0.2364
Epoch 14/20
500/500 [==============================] - 95s 190ms/step - loss: 0.7908 -
rpn_class_loss: 0.0091 - rpn_bbox_loss: 0.2474 - mrcnn_class_loss: 0.1797 -
mrcnn_bbox_loss: 0.0582 - mrcnn_mask_loss: 0.2964 - val_loss: 1.4010 -
val_rpn_class_loss: 0.0122 - val_rpn_bbox_loss: 0.5037 - val_mrcnn_class_loss:
0.4676 - val_mrcnn_bbox_loss: 0.1205 - val_mrcnn_mask_loss: 0.2971
Epoch 15/20
500/500 [==============================] - 95s 190ms/step - loss: 0.7719 -
rpn_class_loss: 0.0090 - rpn_bbox_loss: 0.2511 - mrcnn_class_loss: 0.1717 -
mrcnn_bbox_loss: 0.0606 - mrcnn_mask_loss: 0.2796 - val_loss: 1.1897 -
val_rpn_class_loss: 0.0124 - val_rpn_bbox_loss: 0.6217 - val_mrcnn_class_loss:
0.1832 - val_mrcnn_bbox_loss: 0.1333 - val_mrcnn_mask_loss: 0.2392
```

```
Epoch 16/20
500/500 [==============================] - 95s 190ms/step - loss: 0.7055 -
rpn_class_loss: 0.0096 - rpn_bbox_loss: 0.2297 - mrcnn_class_loss: 0.1570 -
mrcnn_bbox_loss: 0.0515 - mrcnn_mask_loss: 0.2577 - val_loss: 1.2566 -
val_rpn_class_loss: 0.0078 - val_rpn_bbox_loss: 0.4970 - val_mrcnn_class_loss:
0.3387 - val_mrcnn_bbox_loss: 0.1028 - val_mrcnn_mask_loss: 0.3104
Epoch 17/20
500/500 [==============================] - 95s 190ms/step - loss: 0.6862 -
rpn_class_loss: 0.0092 - rpn_bbox_loss: 0.2096 - mrcnn_class_loss: 0.1396 -
mrcnn_bbox_loss: 0.0545 - mrcnn_mask_loss: 0.2733 - val_loss: 1.1768 -
val_rpn_class_loss: 0.0119 - val_rpn_bbox_loss: 0.2957 - val_mrcnn_class_loss:
0.5916 - val_mrcnn_bbox_loss: 0.0587 - val_mrcnn_mask_loss: 0.2190
Epoch 18/20
500/500 [==============================] - 95s 189ms/step - loss: 0.6804 -
rpn_class_loss: 0.0090 - rpn_bbox_loss: 0.2142 - mrcnn_class_loss: 0.1412 -
mrcnn_bbox_loss: 0.0519 - mrcnn_mask_loss: 0.2640 - val_loss: 1.0045 -
val_rpn_class_loss: 0.0044 - val_rpn_bbox_loss: 0.5210 - val_mrcnn_class_loss:
0.1510 - val_mrcnn_bbox_loss: 0.1010 - val_mrcnn_mask_loss: 0.2272
Epoch 19/20
500/500 [==============================] - 95s 189ms/step - loss: 0.6162 -
rpn_class_loss: 0.0084 - rpn_bbox_loss: 0.1855 - mrcnn_class_loss: 0.1212 -
mrcnn_bbox_loss: 0.0475 - mrcnn_mask_loss: 0.2535 - val_loss: 1.1202 -
val_rpn_class_loss: 0.0073 - val_rpn_bbox_loss: 0.3786 - val_mrcnn_class_loss:
0.4194 - val_mrcnn_bbox_loss: 0.0770 - val_mrcnn_mask_loss: 0.2379
Epoch 20/20
500/500 [==============================] - 95s 190ms/step - loss: 0.6365 -
rpn_class_loss: 0.0078 - rpn_bbox_loss: 0.1795 - mrcnn_class_loss: 0.1370 -
mrcnn_bbox_loss: 0.0480 - mrcnn_mask_loss: 0.2641 - val_loss: 1.0999 -
val_rpn_class_loss: 0.0116 - val_rpn_bbox_loss: 0.5544 - val_mrcnn_class_loss:
0.0828 - val_mrcnn_bbox_loss: 0.1083 - val_mrcnn_mask_loss: 0.3428
Training took 33.44 minutes
```

[0]:
```python
# Fine tune all layers
# Passing layers="all" trains all layers. You can also
# pass a regular expression to select which layers to
# train by name pattern.

# start_train = time.time()
# model.train(dataset_train, dataset_val,
#             learning_rate=config.LEARNING_RATE / 10,
#             epochs=8,
#             layers="all")
# end_train = time.time()
# minutes = round((end_train - start_train) / 60, 2)
# print(f'Training took {minutes} minutes')
```

[0]:
```
class InferenceConfig(Cervic_seven_classConfig):
    GPU_COUNT = 1
    IMAGES_PER_GPU = 1
    IMAGE_MIN_DIM = 512
    IMAGE_MAX_DIM = 512
    # DETECTION_MIN_CONFIDENCE = 0.85
    DETECTION_MIN_CONFIDENCE = 0.65

inference_config = InferenceConfig()
```

[0]:
```
# Set the ROOT_DIR variable to the root directory of the Mask_RCNN git repo
ROOT_DIR = '/content/drive/My Drive/'
assert os.path.exists(ROOT_DIR), 'ROOT_DIR does not exist. Did you forget to␣
 ↪read the instructions above? ;)'

# Import mrcnn libraries
sys.path.append(ROOT_DIR)
from mrcnn.config import Config
import mrcnn.utils as utils
from mrcnn import visualize
import mrcnn.model as modellib
```

[0]:
```
# Recreate the model in inference mode
model = modellib.MaskRCNN(mode="inference",
                          config=inference_config,
                          model_dir=MODEL_DIR )
```

[33]:
```
# Get path to saved weights

# Either set a specific path or find last trained weights
COCO_MODEL_PATH= '/content/drive/My Drive/cervic_logs/
 ↪mask_rcnn_cervic_seven_class_0020.h5'
model_path = os.path.join(ROOT_DIR, COCO_MODEL_PATH )
#model_path = model.find_last()

# Load trained weights (fill in path to trained weights here)
assert model_path != "", "Provide path to trained weights"
print("Loading weights from ", model_path)
model.load_weights(model_path, by_name=True)
```

```
Loading weights from  /content/drive/My
Drive/cervic_logs/mask_rcnn_cervic_seven_class_0020.h5
```

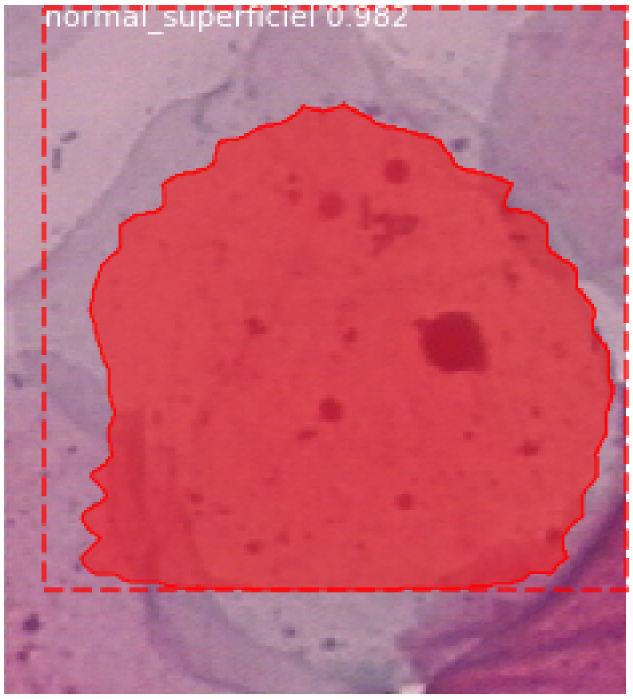
[0]:
```

  def class_find(cl_id):

      names=  {
```

```
                              '1': 'normal_intermediate',
                              '2': 'light_dysplastic',
                              '3': 'moderate_dysplastic',
                              '4': 'severe_dysplastic' ,
                              '5': 'normal_columnar' ,
                              '6': 'carcinoma_in_situ',
                              '7':  'normal_superficiel'
                      }
      return names.get(cl_id)
```

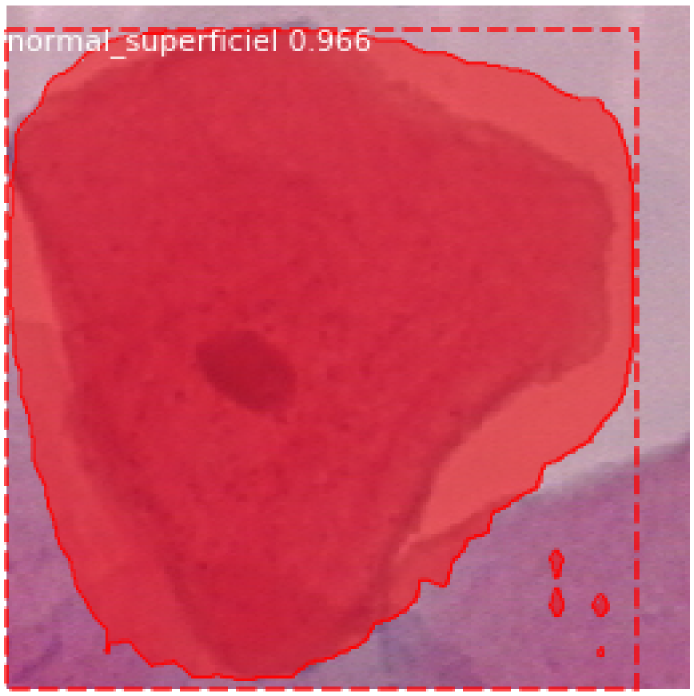
[56]:
```
import skimage
real_test_dir = '/content/drive/My Drive/cervic_test/normal_superficiel'

acc=0
image_paths = []
file_count=0
for filename in os.listdir(real_test_dir):
    if os.path.splitext(filename)[1] in ['.png', '.jpg', '.jpeg','.BMP']:
        image_paths.append(os.path.join(real_test_dir, filename))
        file_count=file_count+1

for image_path in image_paths:
    print('filename:'+image_path)
    img = skimage.io.imread(image_path)
    img_arr = np.array(img)
    results = model.detect([img_arr], verbose=1)
    r = results[0]
    print(r['class_ids'][0])
    class_name=class_find(str(r['class_ids'][0]))
    actual_class= real_test_dir.rsplit('/', 1)[1]
    if class_name==actual_class:
      acc=acc+1
    print('Predicted class :' +class_name + ' Actual class :'+actual_class)
    visualize.display_instances(img, r['rois'], r['masks'], r['class_ids'],
                                dataset_val.class_names, r['scores'],␣
 ↪figsize=(5,5))
print('Total no. of images in ',actual_class, '  is  ', file_count)
print('No. of images correctly classified is  ', acc)
accper=(acc/file_count) *100
print('Accuracy  of class:  ' , actual_class, ' is ', str(accper))
```

```
filename:/content/drive/My
Drive/cervic_test/normal_superficiel/157268504-157268544-001.BMP
Processing 1 images
image                    shape: (349, 315, 3)         min:   39.00000  max:
225.00000  uint8
molded_images            shape: (1, 512, 512, 3)      min: -123.70000  max:
```

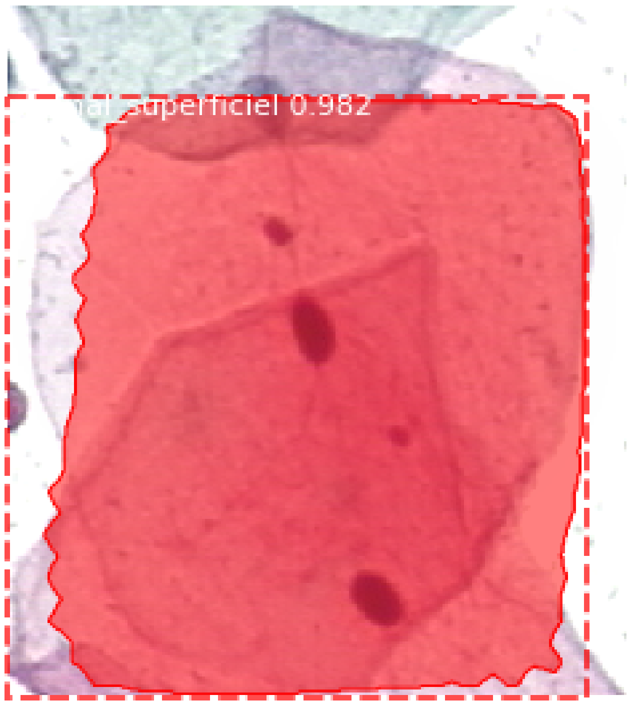
```
100.30000  float64
image_metas              shape: (1, 20)             min:    0.00000  max:
512.00000  float64
anchors                  shape: (1, 65472, 4)       min:   -0.17712  max:
1.05188  float32
7
Predicted class :normal_superficiel Actual class :normal_superficiel
```



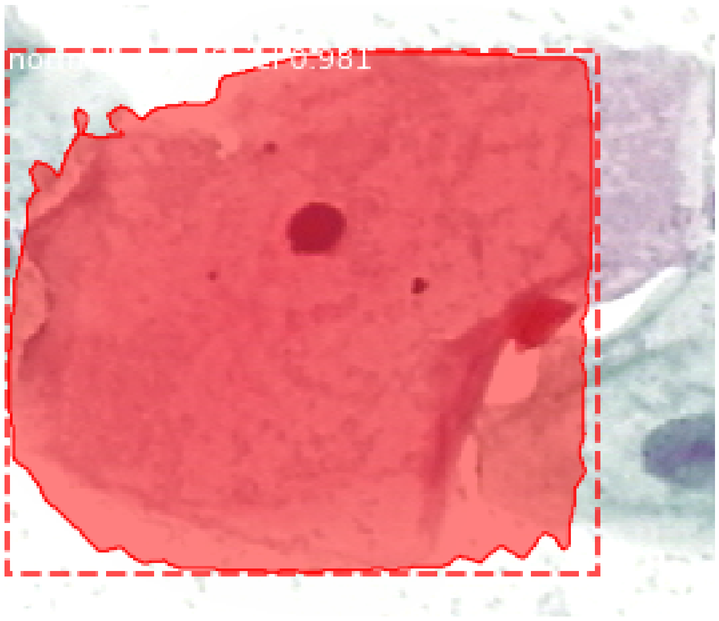
```
filename:/content/drive/My
Drive/cervic_test/normal_superficiel/157268587-157268617-001.BMP
Processing 1 images
image                    shape: (324, 323, 3)       min:   38.00000  max:
223.00000  uint8
molded_images            shape: (1, 512, 512, 3)    min: -123.70000  max:
98.30000  float64
image_metas              shape: (1, 20)             min:    0.00000  max:
512.00000  float64
anchors                  shape: (1, 65472, 4)       min:   -0.17712  max:
1.05188  float32
7
Predicted class :normal_superficiel Actual class :normal_superficiel
```



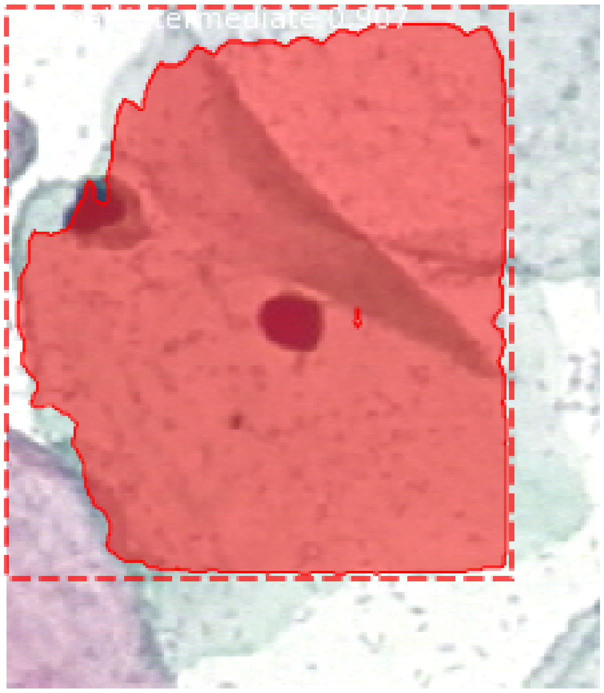
```
filename:/content/drive/My
Drive/cervic_test/normal_superficiel/158987033-158987057-001.BMP
Processing 1 images
image                    shape: (354, 318, 3)        min:   46.00000  max:
255.00000  uint8
molded_images            shape: (1, 512, 512, 3)     min: -123.70000  max:
151.10000  float64
image_metas              shape: (1, 20)              min:    0.00000  max:
512.00000  float64
anchors                  shape: (1, 65472, 4)        min:   -0.17712  max:
1.05188  float32
7
Predicted class :normal_superficiel Actual class :normal_superficiel
```



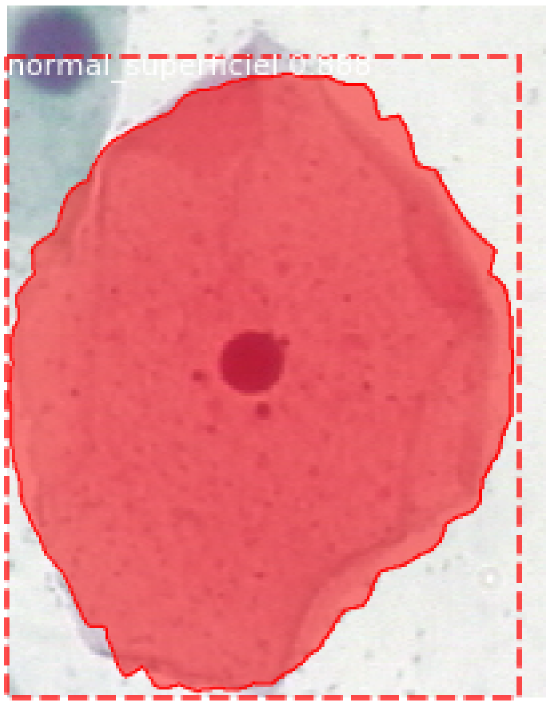
```
filename:/content/drive/My
Drive/cervic_test/normal_superficiel/158987453-158987462-001.BMP
Processing 1 images
image                    shape: (345, 402, 3)        min:   44.00000  max:
255.00000  uint8
molded_images            shape: (1, 512, 512, 3)     min: -123.70000  max:
151.10000  float64
image_metas              shape: (1, 20)              min:    0.00000  max:
512.00000  float64
anchors                  shape: (1, 65472, 4)        min:   -0.17712  max:
1.05188  float32
7
Predicted class :normal_superficiel Actual class :normal_superficiel
```



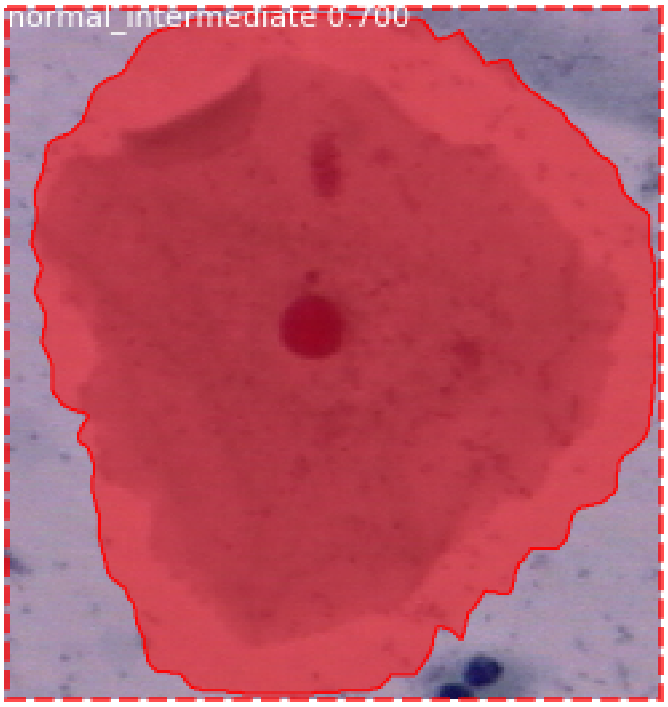
```
filename:/content/drive/My
Drive/cervic_test/normal_superficiel/158987493-158987505-001.BMP
Processing 1 images
image                    shape: (310, 269, 3)        min:   46.00000  max:
255.00000  uint8
molded_images            shape: (1, 512, 512, 3)     min: -123.70000  max:
151.10000  float64
image_metas              shape: (1, 20)              min:    0.00000  max:
512.00000  float64
anchors                  shape: (1, 65472, 4)        min:   -0.17712  max:
1.05188  float32
1
Predicted class :normal_intermediate Actual class :normal_superficiel
```



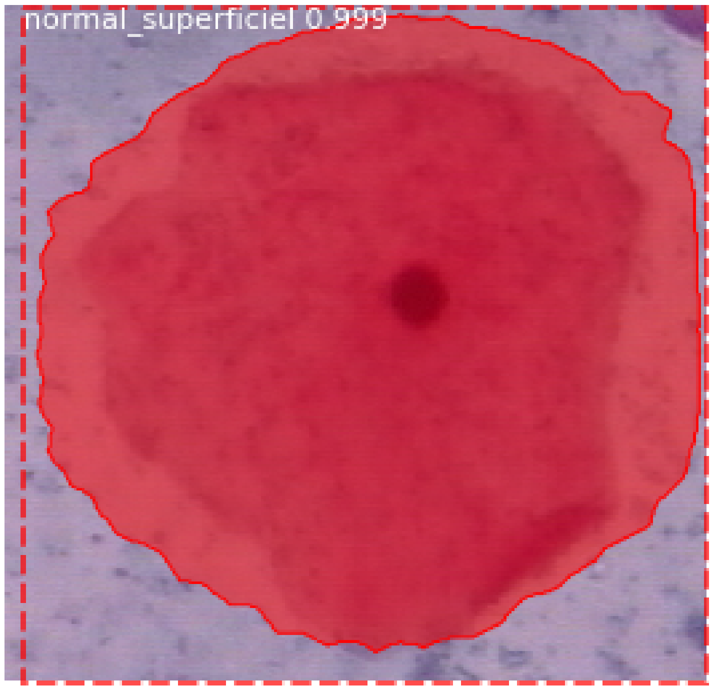
```
filename:/content/drive/My
Drive/cervic_test/normal_superficiel/158987493-158987499-001.BMP
Processing 1 images
image                    shape: (382, 298, 3)        min:   52.00000  max:
255.00000  uint8
molded_images            shape: (1, 512, 512, 3)     min: -123.70000  max:
150.10000  float64
image_metas              shape: (1, 20)              min:    0.00000  max:
512.00000  float64
anchors                  shape: (1, 65472, 4)        min:   -0.17712  max:
1.05188  float32
7
Predicted class :normal_superficiel Actual class :normal_superficiel
```



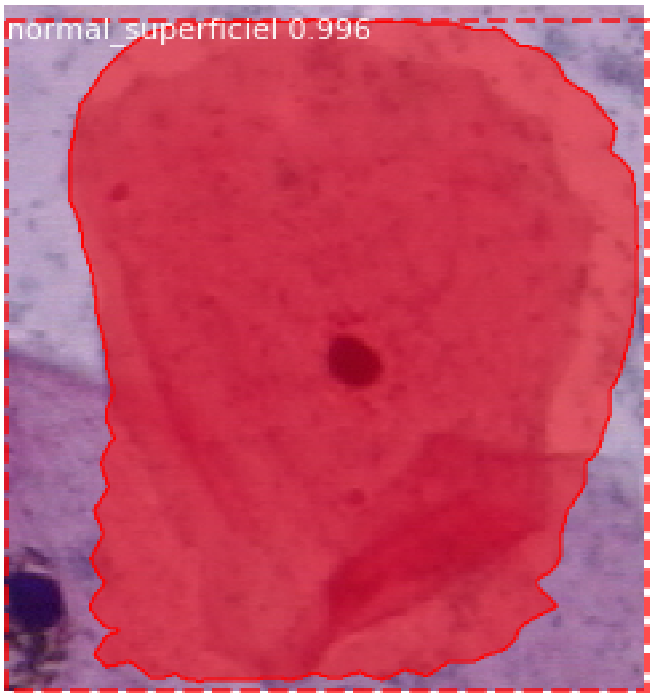
```
filename:/content/drive/My
Drive/cervic_test/normal_superficiel/209047342-209047400-001.BMP
Processing 1 images
image                    shape: (399, 378, 3)        min:    8.00000  max:
193.00000  uint8
molded_images            shape: (1, 512, 512, 3)     min: -123.70000  max:
87.10000  float64
image_metas              shape: (1, 20)              min:    0.00000  max:
512.00000  float64
anchors                  shape: (1, 65472, 4)        min:   -0.17712  max:
1.05188  float32
1
Predicted class :normal_intermediate Actual class :normal_superficiel
```



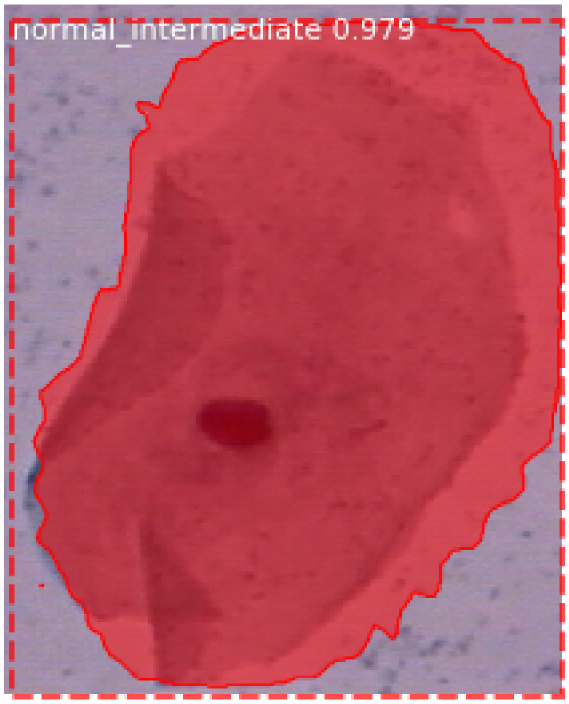
```
filename:/content/drive/My
Drive/cervic_test/normal_superficiel/209047342-209047443-001.BMP
Processing 1 images
image                    shape: (310, 322, 3)       min:    0.00000  max:
199.00000  uint8
molded_images            shape: (1, 512, 512, 3)    min: -123.70000  max:
92.10000  float64
image_metas              shape: (1, 20)             min:    0.00000  max:
512.00000  float64
anchors                  shape: (1, 65472, 4)       min:   -0.17712  max:
1.05188  float32
7
Predicted class :normal_superficiel Actual class :normal_superficiel
```



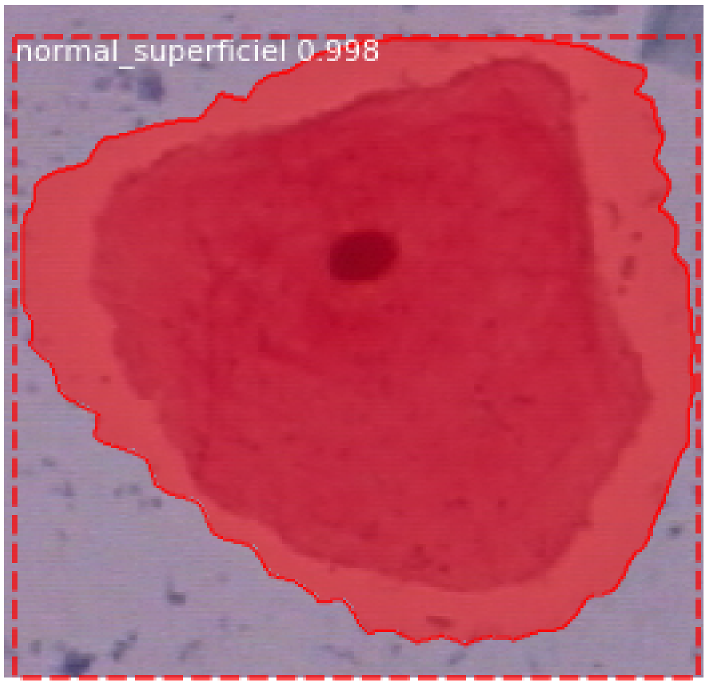
```
filename:/content/drive/My
Drive/cervic_test/normal_superficiel/209047342-209047478-001.BMP
Processing 1 images
image                    shape: (336, 314, 3)        min:    0.00000  max:
216.00000  uint8
molded_images            shape: (1, 512, 512, 3)     min: -123.70000  max:
111.10000  float64
image_metas              shape: (1, 20)              min:    0.00000  max:
512.00000  float64
anchors                  shape: (1, 65472, 4)        min:   -0.17712  max:
1.05188  float32
7
Predicted class :normal_superficiel Actual class :normal_superficiel
```



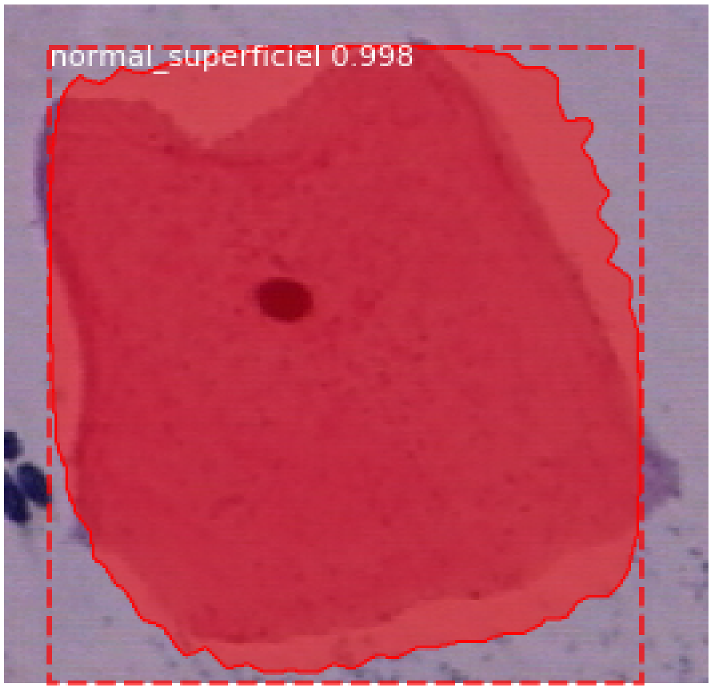
```
filename:/content/drive/My
Drive/cervic_test/normal_superficiel/209047526-209047717-001.BMP
Processing 1 images
image                    shape: (357, 289, 3)        min:    0.00000  max:
196.00000  uint8
molded_images            shape: (1, 512, 512, 3)     min: -123.70000  max:
87.10000  float64
image_metas              shape: (1, 20)              min:    0.00000  max:
512.00000  float64
anchors                  shape: (1, 65472, 4)        min:   -0.17712  max:
1.05188  float32
1
Predicted class :normal_intermediate Actual class :normal_superficiel
```



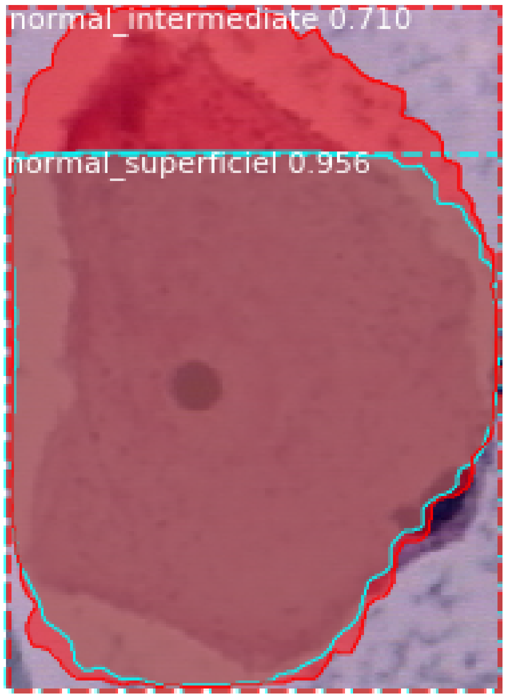
```
filename:/content/drive/My
Drive/cervic_test/normal_superficiel/209047881-209048017-001.BMP
Processing 1 images
image                    shape: (280, 291, 3)        min:    0.00000  max:
185.00000  uint8
molded_images            shape: (1, 512, 512, 3)     min: -123.70000  max:
80.10000  float64
image_metas              shape: (1, 20)              min:    0.00000  max:
512.00000  float64
anchors                  shape: (1, 65472, 4)        min:   -0.17712  max:
1.05188  float32
7
Predicted class :normal_superficiel Actual class :normal_superficiel
```



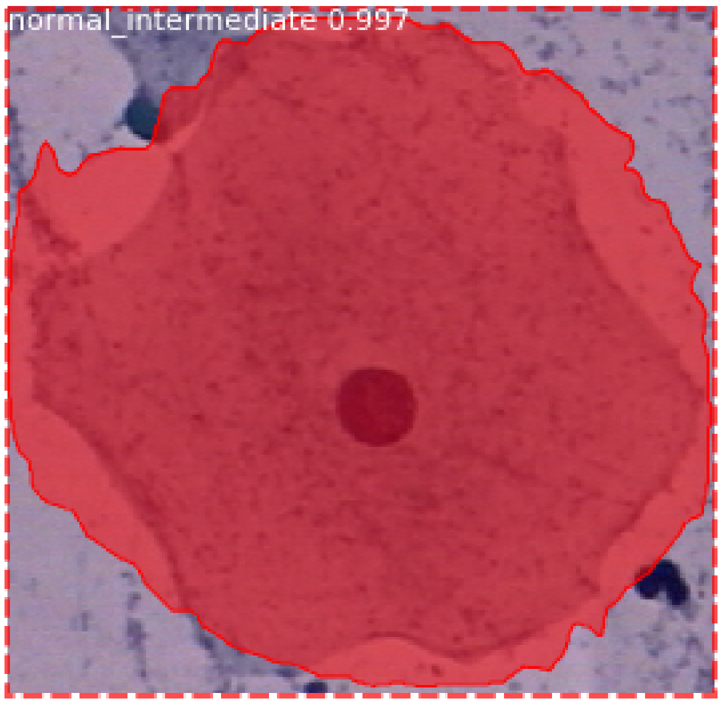
```
filename:/content/drive/My
Drive/cervic_test/normal_superficiel/209047526-209047798-001.BMP
Processing 1 images
image                    shape: (331, 345, 3)        min:    0.00000  max:
186.00000  uint8
molded_images            shape: (1, 512, 512, 3)     min: -123.70000  max:
79.10000  float64
image_metas              shape: (1, 20)              min:    0.00000  max:
512.00000  float64
anchors                  shape: (1, 65472, 4)        min:   -0.17712  max:
1.05188  float32
7
Predicted class :normal_superficiel Actual class :normal_superficiel
```



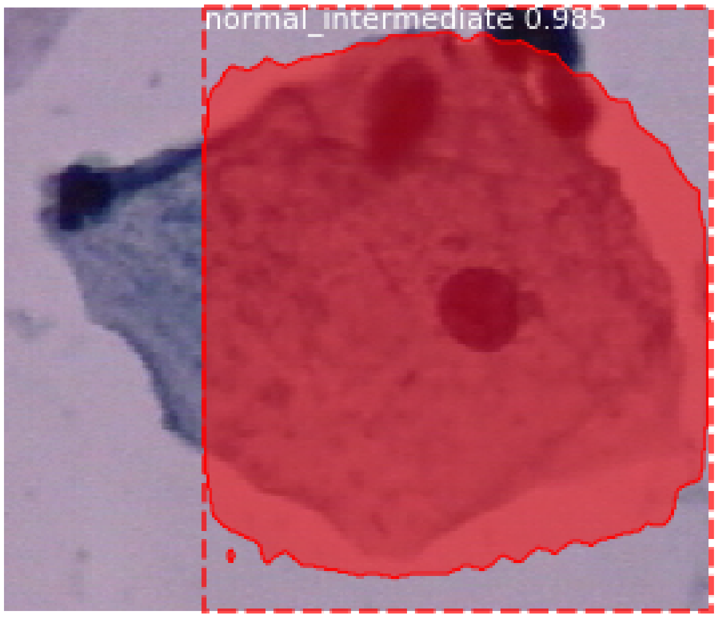
```
filename:/content/drive/My
Drive/cervic_test/normal_superficiel/209048086-209048278-001.BMP
Processing 1 images
image                    shape: (319, 231, 3)        min:    0.00000  max:
212.00000  uint8
molded_images            shape: (1, 512, 512, 3)     min: -123.70000  max:
106.10000  float64
image_metas              shape: (1, 20)              min:    0.00000  max:
512.00000  float64
anchors                  shape: (1, 65472, 4)        min:   -0.17712  max:
1.05188  float32
7
Predicted class :normal_superficiel Actual class :normal_superficiel
```



```
filename:/content/drive/My
Drive/cervic_test/normal_superficiel/209048086-209048137-001.BMP
Processing 1 images
image                    shape: (362, 374, 3)        min:    3.00000  max:
197.00000  uint8
molded_images            shape: (1, 512, 512, 3)     min: -123.70000  max:
88.10000  float64
image_metas              shape: (1, 20)              min:    0.00000  max:
512.00000  float64
anchors                  shape: (1, 65472, 4)        min:   -0.17712  max:
1.05188  float32
1
Predicted class :normal_intermediate Actual class :normal_superficiel
```



```
filename:/content/drive/My
Drive/cervic_test/normal_superficiel/209307421-209307597-001.BMP
Processing 1 images
image                    shape: (297, 347, 3)         min:    7.00000  max:
193.00000  uint8
molded_images            shape: (1, 512, 512, 3)      min: -123.70000  max:
85.10000  float64
image_metas              shape: (1, 20)               min:    0.00000  max:
512.00000  float64
anchors                  shape: (1, 65472, 4)         min:   -0.17712  max:
1.05188  float32
1
Predicted class :normal_intermediate Actual class :normal_superficiel
```



```
Total no. of images in  normal_superficiel   is   15
No. of images correctly classified is   10
Accuracy  of class:   normal_superficiel  is  66.66666666666666
```