Run an Instance Segmentation Model

For some applications it isn't adequate enough to localize an object with a simple bounding box. For instance, you might want to segment an object region once it is detected. This class of problems is called **instance segmentation**.

Materializing data for instance segmentation

Instance segmentation is an extension of object detection, where a binary mask (i.e. object vs. background) is associated with every bounding box. This allows for more fine-grained information about the extent of the object within the box. To train an instance segmentation model, a groundtruth mask must be supplied for every groundtruth bounding box. In additional to the proto fields listed in the section titled Using your own dataset, one must also supply `image/object/mask`, which can either be a repeated list of single-channel encoded PNG strings, or a single dense 3D binary tensor where masks corresponding to each object are stacked along the first dimension. Each is described in more detail below.

PNG Instance Segmentation Masks Instance segmentation masks can be supplied as serialized PNG images.

```
image/object/mask = ["\x89PNG\r\n\x1A\n\x00\x00\x00\rIHDR\...", ...]
```

These masks are whole-image masks, one for each object instance. The spatial dimensions of each mask must agree with the image. Each mask has only a single channel, and the pixel values are either 0 (background) or 1 (object mask). **PNG masks are the preferred parameterization since they offer considerable space savings compared to dense numerical masks.**

Dense Numerical Instance Segmentation Masks Masks can also be specified via a dense numerical tensor.

```
image/object/mask = [0.0, 0.0, 1.0, 1.0, 0.0, ...]
```

For an image with dimensions `H` x `W` and `num_boxes` groundtruth boxes, the mask corresponds to a [`num_boxes`, `H`, `W`] float32 tensor, flattened into a single vector of shape `num_boxes` * `H` * `W`. In TensorFlow, examples are read in row-major format, so the elements are organized as:

```
... mask 0 row 0 ... mask 0 row 1 ... // ... mask 0 row H-1 ... mask 1 row 0 ...
```

where each row has W contiguous binary values.

To see an example tf-records with mask labels, see the examples under the Preparing Inputs section.

Pre-existing config files

We provide four instance segmentation config files that you can use to train your own models:

1. mask_rcnn_inception_resnet_v2_atrous_coco
2. mask_rcnn_resnet101_atrous_coco
3. mask_rcnn_resnet50_atrous_coco
4. mask_rcnn_inception_v2_coco

For more details see the detection model zoo.

Updating a Faster R-CNN config file

Currently, the only supported instance segmentation model is Mask R-CNN, which requires Faster R-CNN as the backbone object detector.

Once you have a baseline Faster R-CNN pipeline configuration, you can make the following modifications in order to convert it into a Mask R-CNN model.

1. Within `train_input_reader` and `eval_input_reader`, set `load_instance_masks` to `True`. If using PNG masks, set `mask_type` to `PNG_MASKS`, otherwise you can leave it as the default 'NUMERICAL_MASKS'.
2. Within the `faster_rcnn` config, use a `MaskRCNNBoxPredictor` as the `second_stage_box_predictor`.
3. Within the `MaskRCNNBoxPredictor` message, set `predict_instance_masks` to `True`. You must also define `conv_hyperparams`.
4. Within the `faster_rcnn` message, set `number_of_stages` to `3`.
5. Add instance segmentation metrics to the set of metrics: `'coco_mask_metrics'`.
6. Update the `input_path`s to point at your data.

Please refer to the section on Running the pets dataset for additional details.

> Note: The mask prediction branch consists of a sequence of convolution layers. You can set the number of convolution layers and their depth as follows:
>
> 1. Within the `MaskRCNNBoxPredictor` message, set the `mask_prediction_conv_depth` to your value of interest. The default value is 256. If you set it to `0` (recommended), the depth is computed automatically based on the number of classes in the dataset.
> 2. Within the `MaskRCNNBoxPredictor` message, set the `mask_prediction_num_conv_layers` to your value of interest. The default value is 2.