GPU usage at JEODPP platform

September 26, 2018

Table of contents

Instance segmentation

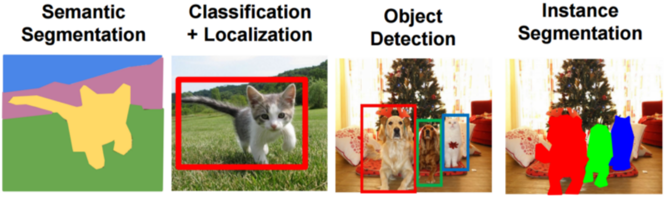


Figure: Source: [1]

Mask R-CNN

Two parts:

- backbone
- head

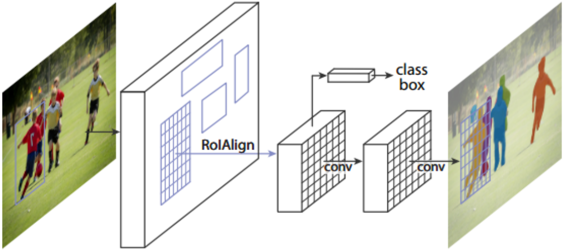


Figure: Source: [2]

Usage

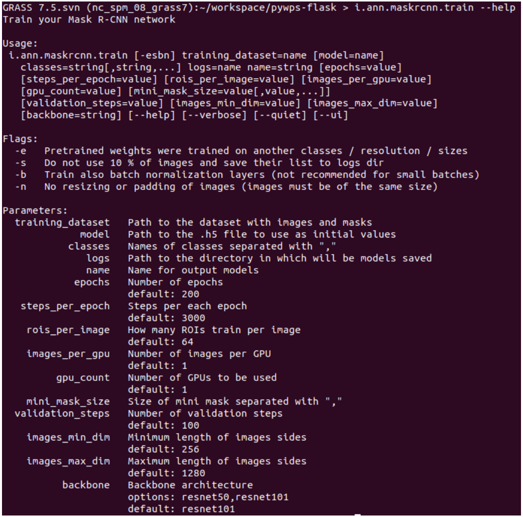
Software:

GRASS GIS

Workflow:

- i.ann.maskrcnn.train
- i.ann.maskrcnn.detect

i.ann.maskrcnn.train



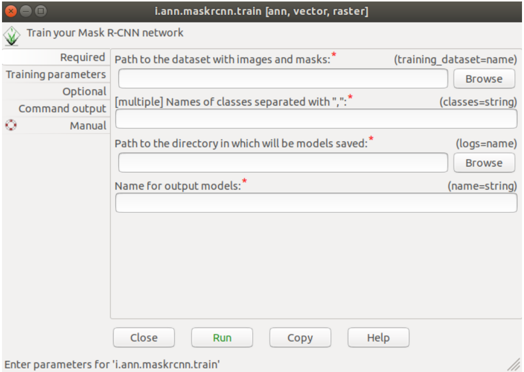
```
GRASS 7.5.svn (nc_spm_08_grass7):~/workspace/pywps-flask > i.ann.maskrcnn.train --help
Train your Mask R-CNN network

Usage:
 i.ann.maskrcnn.train [-esbn] training_dataset=name [model=name]
   classes=string[,string,...] logs=name name=string [epochs=value]
   [steps_per_epoch=value] [rois_per_image=value] [images_per_gpu=value]
   [gpu_count=value] [mini_mask_size=value[,value,...]]
   [validation_steps=value] [images_min_dim=value] [images_max_dim=value]
   [backbone=string] [--help] [--verbose] [--quiet] [--ui]

Flags:
  -e   Pretrained weights were trained on another classes / resolution / sizes
  -s   Do not use 10 % of images and save their list to logs dir
  -b   Train also batch normalization layers (not recommended for small batches)
  -n   No resizing or padding of images (images must be of the same size)

Parameters:
  training_dataset   Path to the dataset with images and masks
             model   Path to the .h5 file to use as initial values
           classes   Names of classes separated with ","
              logs   Path to the directory in which will be models saved
              name   Name for output models
            epochs   Number of epochs
                     default: 200
   steps_per_epoch   Steps per each epoch
                     default: 3000
    rois_per_image   How many ROIs train per image
                     default: 64
    images_per_gpu   Number of images per GPU
                     default: 1
         gpu_count   Number of GPUs to be used
                     default: 1
    mini_mask_size   Size of mini mask separated with ","
  validation_steps   Number of validation steps
                     default: 100
    images_min_dim   Minimum length of images sides
                     default: 256
    images_max_dim   Maximum length of images sides
                     default: 1280
          backbone   Backbone architecture
                     options: resnet50,resnet101
                     default: resnet101
```

i.ann.maskrcnn.train



i.ann.maskrcnn.train [ann, vector, raster]

Train your Mask R-CNN network

Path to the dataset with images and masks:* (training_dataset=name)

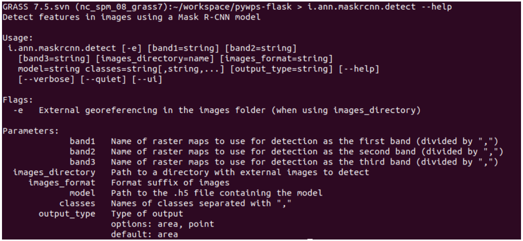
[multiple] Names of classes separated with ",":* (classes=string)

Path to the directory in which will be models saved:* (logs=name)

Name for output models:* (name=string)

Enter parameters for 'i.ann.maskrcnn.train'

i.ann.maskrcnn.detect



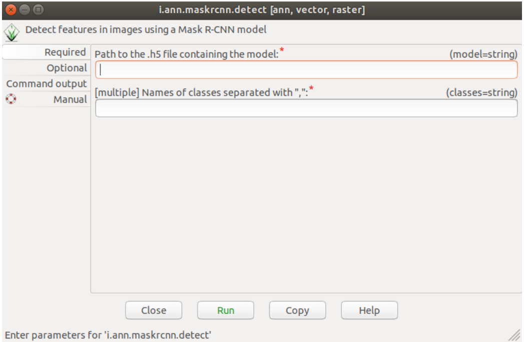
```
GRASS 7.5.svn (nc_spm_08_grass7):~/workspace/pywps-flask > i.ann.maskrcnn.detect --help
Detect features in images using a Mask R-CNN model

Usage:
 i.ann.maskrcnn.detect [-e] [band1=string] [band2=string]
   [band3=string] [images_directory=name] [images_format=string]
   model=string classes=string[,string,...] [output_type=string] [--help]
   [--verbose] [--quiet] [--ui]

Flags:
  -e   External georeferencing in the images folder (when using images_directory)

Parameters:
             band1   Name of raster maps to use for detection as the first band (divided by ",")
             band2   Name of raster maps to use for detection as the second band (divided by ",")
             band3   Name of raster maps to use for detection as the third band (divided by ",")
  images_directory   Path to a directory with external images to detect
     images_format   Format suffix of images
             model   Path to the .h5 file containing the model
           classes   Names of classes separated with ","
       output_type   Type of output
                     options: area, point
                     default: area
```

i.ann.maskrcnn.detect



i.ann.maskrcnn.detect [ann, vector, raster]

Detect features in images using a Mask R-CNN model

Path to the .h5 file containing the model:* (model=string)

[multiple] Names of classes separated with ",":* (classes=string)

Enter parameters for 'i.ann.maskrcnn.detect'

Outputs

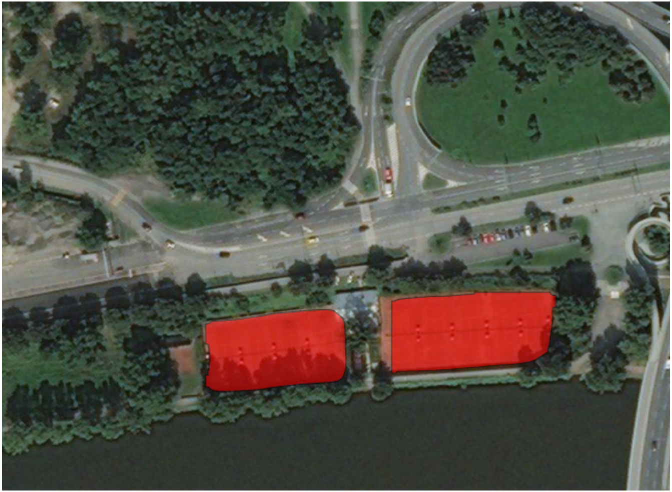


Figure: loss function 0.96, 54000 training images

Outputs

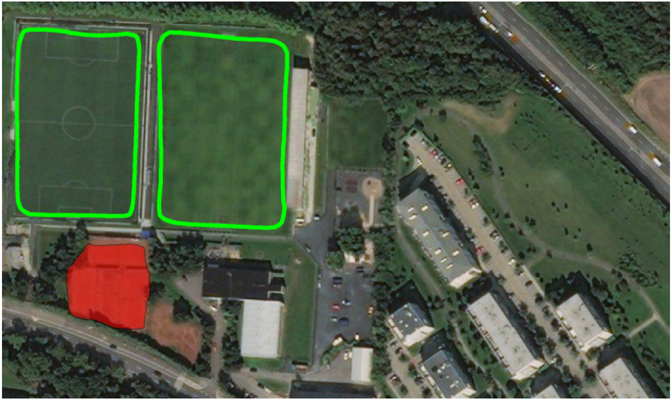


Figure: loss function 0.96, 54000 training images

Outputs

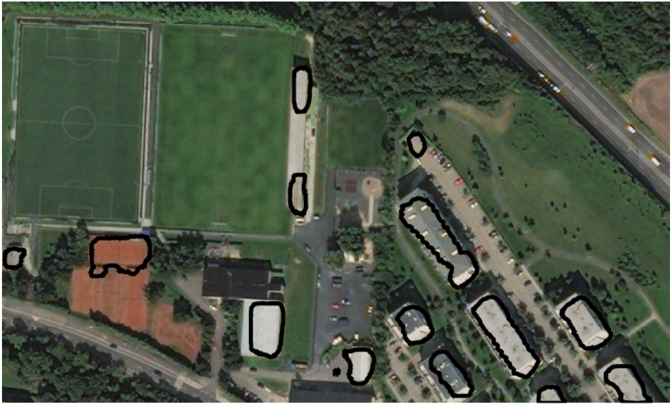


Figure: epoch 150, loss function 0.63, 2400 training images

Tools availability

- source code
  - https://github.com/ctu-geoforall-lab/i.ann.maskrcnn
  - https://svn.osgeo.org/grass/grass-addons/grass7/imagery/i.ann.maskrcnn/
- installation using command

```
g.extension extension=i.ann.maskrcnn
```

Statistics

Processing units	Seconds per image	GPU usage [%]	GPU memory usage [MB]	CPU relative load	Main memory usage [GB]
20 CPUs	9.937	0	0	0.18	147.7
1 GPU	3.945	81	11.31	0.06	151.1
2 GPUs	3.711	69	11.31	0.06	115.8
3 GPUs	3.456	58	11.31	0.07	173.7
4 GPUs	3.010	46	11.31	0.08	219.9

Statistics

Processing units	Seconds per image	GPU usage [%]	GPU memory usage [MB]	CPU relative load	Main memory usage [GB]
20 CPUs	9.937	0	0	0.18	147.7
1 GPU	3.945	81	11.31	0.06	151.1
2 GPUs	3.711	69	11.31	0.06	115.8
3 GPUs	3.456	58	11.31	0.07	173.7
4 GPUs	3.010	46	11.31	0.08	219.9

Processing units	Images per GPU	Seconds per image
1 GPU	4	4.250
4 GPUs	1	3.010

Statistics

Processing units	Seconds per image	GPU usage [%]	GPU memory usage [MB]	CPU relative load	Main memory usage [GB]
20 CPUs	9.937	0	0	0.18	147.7
1 GPU	3.945	81	11.31	0.06	151.1
2 GPUs	3.711	69	11.31	0.06	115.8
3 GPUs	3.456	58	11.31	0.07	173.7
4 GPUs	3.010	46	11.31	0.08	219.9

Processing units	Images per GPU	Seconds per image
1 GPU	4	4.250
4 GPUs	1	3.010

Processing units	Number of simultaneous processes	Seconds per image
1 GPU	1	3.945
1 GPU	2	3.999
2 GPUs	1	3.711
2 GPUs	2	3.762

Conclusion

- 1 GPU is in average 2.5 times faster than 20 CPUs

Conclusion

- 1 GPU is in average 2.5 times faster than 20 CPUs
- 4 GPUs are in average 1.5 times faster than 1 GPU

Conclusion

- 1 GPU is in average 2.5 times faster than 20 CPUs
- 4 GPUs are in average 1.5 times faster than 1 GPU
- GPUs with 1 image loaded per GPU are in average 1.5 faster than 1 GPU with 4 images per GPU

Conclusion

- 1 GPU is in average 2.5 times faster than 20 CPUs
- 4 GPUs are in average 1.5 times faster than 1 GPU
- GPUs with 1 image loaded per GPU are in average 1.5 faster than 1 GPU with 4 images per GPU
- GPU with restricted access to the memory of other GPUs reaches almost the same speed as with access

Conclusion

- 1 GPU is in average 2.5 times faster than 20 CPUs
- 4 GPUs are in average 1.5 times faster than 1 GPU
- GPUs with 1 image loaded per GPU are in average 1.5 faster than 1 GPU with 4 images per GPU
- GPU with restricted access to the memory of other GPUs reaches almost the same speed as with access
- The optimal solution is to allow multiuser usage with access to 1 GPU per docker

Theoretical framework
Instance segmentation
Mask R-CNN
Tools
Usage
i.ann.maskrcnn.train
i.ann.maskrcnn.detect
Outputs
Tools availability
Statistics
Conclusion
Sources

Sources

[1] http://cs231n.stanford.edu/
[2] HE, Kaiming et al. Mask R-CNN. In: International Conference on Computer Vision (ICCV). 2017.

Thank you for your attention.