Computer-Aided Spatial Planning and Design - Mask R-CNN

Ondřej Pešek

Abstract

The usage of artificial neural networks in the geomatics field is really wide. One of the most requested applications is the classification of satellite images. Among many different approaches, I have chosen Mask R-CNN (region-based convolutional neural networks) proposed by Facebook AI Research as our framework, allowing user to both detect objects and generate a segmentation mask for each instance.

INTRODUCTION

ARTIFICIAL neural network. A magical formula that can be heard more and more often in almost every part of computer science. Computing systems with many applications in finance, data mining, language recognition, pattern recognition and many more. With the last named are connected attempts to use artificial neural networks in geomatics field for an automated classification of satellite images.

Classification is a separation of regions of interest. Artificial neural networks allowed two approaches for classification so far (following the terminology from [10]):

- **Object detection**: Detecting objects via bounding boxes, for example [4]
- **Semantic segmentation**: Per-pixel classification of whole image without assigning pixels to different instances, for example those used in remote sensing [5]

Mask region-based convolutional neural networks (Mask R-CNN) proposed by Facebook AI Research provides a combination of both approaches - object detection via bounding boxes and following semantical segmentation of bounded area. As we are segmenting the detected instance, this process is called instance segmentation and could be useful in satellite images classification.

The goal of this paper is to give an insight into Mask R-CNN and artificial neural networks generally and discuss and design its implementation in geomatics field.

I. PREREQUISITES

SPEAKING generally, Mask R-CNN are just Faster R-CNN [9] with one additional layer for the mask (and few smaller changes). Considering that, I should firstly talk about Faster R-CNN and other related terms before explaining Mask R-CNN specifications in chapter II.

A. R-CNN

R-CNN is an acronym for region-based convolutional neural networks, an algorithm proposed in [7] and combining region proposals with convolutional neural networks to detect objects in image via bounding boxes.

The first step of detection is to propose bounding boxes using the selective search [1] - an algorithm consisting of two steps:

- **All scales capturing**: Generating many bounding boxes (region proposals) at different scales based on their texture, color or intesity similarities. An hierarchical algorithm is used.
- **Group regions together**: Grouping is influenced by colors, shades and other natural and non-natural conditions. To reduce these influences, a diverse set of strategies is held.

After bounding boxes proposition we have to compute region features. This requires warping the region into 227 x 227 square RGB image, which can be forward propagated through an architecture composed of five convolutional layers and two fully connected layers (modified architecture described in [2]).

The final step is scoring of those extracted features (deciding which class the feature is and whether it is any class at all). This is done for each feature separately using a Support Vector Machine (SVM) trained for that class.

In general, the bounding box has a large overlap with the background. This is the last problem with which the algorithm has to deal. To improve the localization (make bounding boxes tighter), a bounding box regression proposed in [6] is used with one modification - R-CNN applies regression on features computed by the CNN instead of features computed on the inferred deformable part model (DPM) locations.

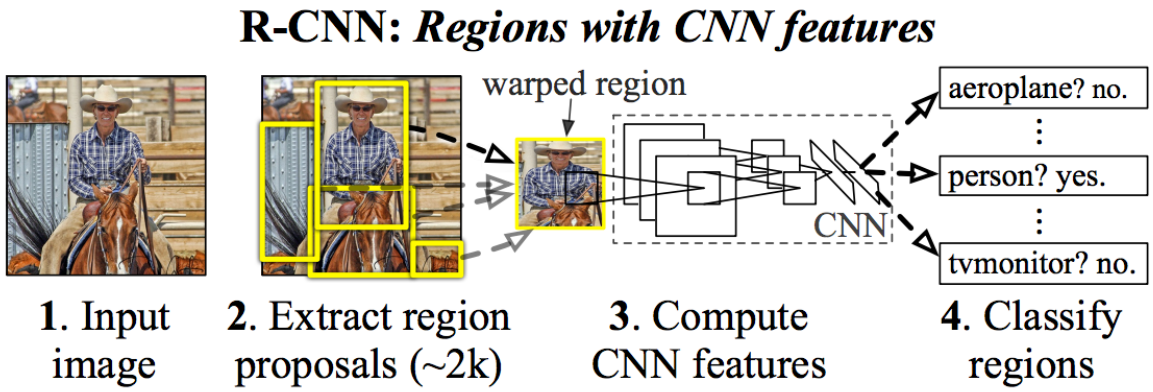


Figure 1. R-CNN architecture, source: [7]

Although R-CNN outperformed similar architectures including OverFeat (according to [7]), there were still some shortcomings. The biggest one was the slowness caused mainly by the forward propagation through CNN (each region of every image must be passed separately), by its triplicated training (a network for generating image features, a network for the class decision and the bounding box regression model) and by the generating of bounding box proposals. Then, in year 2015, Fast R-CNN came to solve first two of these issues. Lead by the author of original R-CNN, Ross Girshick.

B. Fast R-CNN

Because the main reason for a new architecture was to speed-up R-CNN, Ross Girshick named his new architecture proposed in [8] simply Fast R-CNN. Its main purposes were to avoid first two issues mentioned in chapter I-A and improve its accuracy.

The first issue, the separate forward propagations, was solved by propagating the entire image and the full set of its object proposals through a set of convolutional and max pooling layers. For each object proposal is then from the feature map extracted a fixed-length feature vector by a region of interest (RoI) pooling layer. The RoI pooling layer can be seen as a one level spatial pyramid pooling layer, a max pooling based downsampling algorithm proposed in [3], and its purpose is to decompose separately each valid region of interest into a fixed size (7×7) grid, where each cell represents a feature map. The RoIPool decomposition is made by quantization a RoI to the rounded discrete granularity of the feature map which is then (again with quantization) divided into 7×7 bins. Final values of each bin are aggregated by max pooling.

Output of RoI pooling layers is an input for a sequence of fully connected layers. Then the last, branched step where two different outputs are obtained depending on the last layer - class probabilities from a softmax function and a bounding box defined by 4 values from a regressor.

There can be seen also a solution of the second problem named in I-A. Instead of three separate models, all steps are joint into only one model by appending classification (using a softmax layer instead of a separate SVM) and bounding box regression as parallel layers to the end of the model.

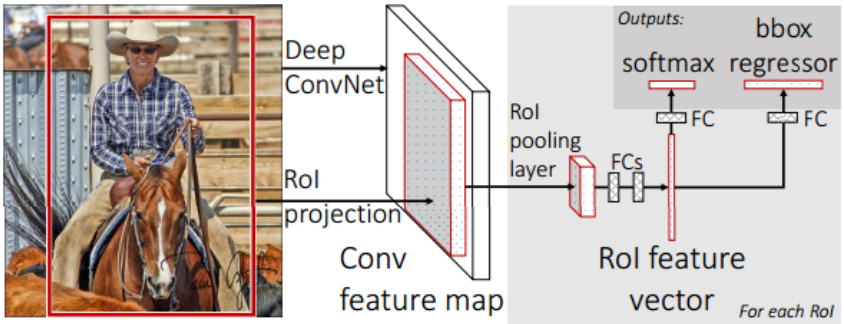


Figure 2. Fast R-CNN architecture, source: [8]

There can be raised a question whether is the SVM replacement with the softmax layer as accurate as the original approach. According to tests performed in [8], the softmax layer is even slightly outperforming SVM.

C. Faster R-CNN

The third speed issue mentioned in chapter I-A - the region proposer based on the selective search - was solved in year 2016 in an architecture imaginatively named Faster R-CNN, proposed in [9].

Microsoft Research team found that feature map computed in the first part of Fast R-CNN can be used to generate region proposals instead of using slower selective search algorithm. This is done by including Region Proposal Network (RPN) consisting of a few convolutional layers (a fully convolutional network) after the convolutional feature maps of Fast R-CNN. The RPN layer backbone are two fully connected layers, one for a bounding box regression and one for a bounding box classification. The complete architecture alternates between fine-tuning for the region proposal task and for the object detection. This approach with shared convolutional features quickly converges.

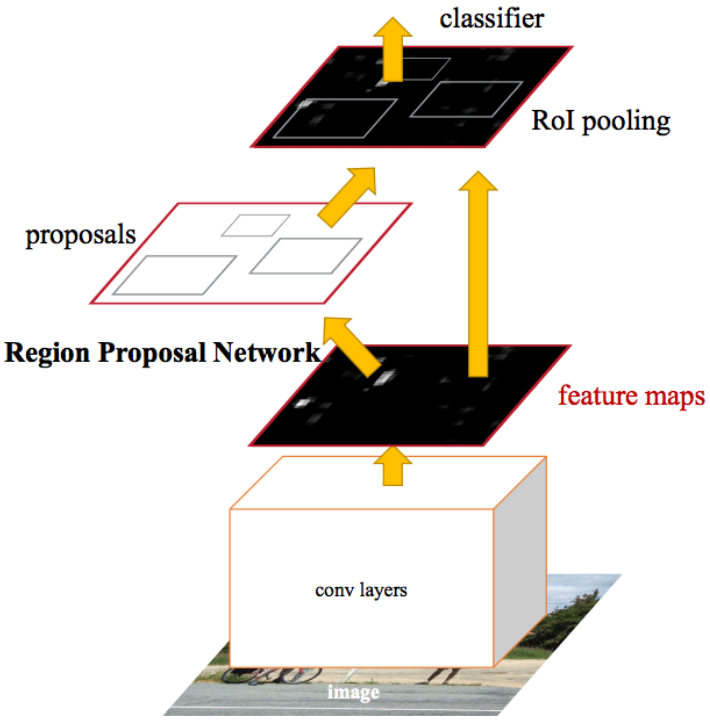


Figure 3. Faster R-CNN architecture, source: [9]

RPN approach is different than the ones used in other architectures. Instead of pyramids of images or filters, RPN uses anchor boxes - a set of rectangular bounding boxes proposals and scores created by sliding a spatial window over the entire convolutional feature map. Those anchors are defined by a scale and aspect ratio, so they are of different shapes. Powerful attributes of anchors are that they are translation-invariant and multi-scale, which concludes also into a reduction of the model size.

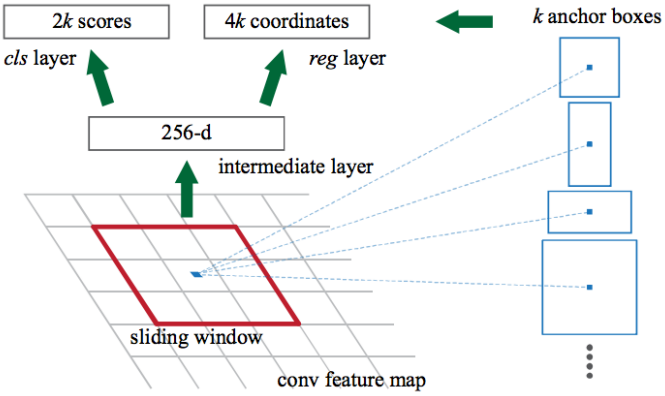


Figure 4. Region Proposal Network, source: [9]

II. Mask R-CNN

SO we had R-CNN, Fast R-CNN and Faster R-CNN. What could be the next step? The first answer that comes to your mind is wrong - instead of expected *The Fastest R-CNN*, the Mask R-CNN were proposed in year 2017 by Facebook AI Research (FAIR) in [10].

FAIR stood in a front of another question. According to [9], Faster R-CNN outperformed most of their concurrents in the object detection. But it was still unusable for some users looking for a semantic segmentation neural network. They had a powerful network for object detection, so the question was: Is there any way to use the current network for the semantic segmentation?

Obviously, the answer was - yes. But instead of the semantic segmentation, they proposed more advanced approach, the instance segmentation, which was described in the introduction of this paper and in [10]. An approach that provides much richer usage in geomatics field.

To implement the instance segmentation with sufficient accuracy, there was a need for some changes in the architecture of Faster R-CNN. Following the original terminology from [10], in the following part, I will distinguish between the backbone architecture for feature extraction and the head architecture for classification, bounding box regression and mask prediction. Those will be described in the following parts of this chapter along with an extra focus on some other major changes.

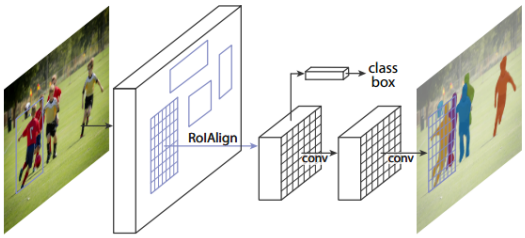


Figure 5. Mask R-CNN architecture, source: [10]

A. Backbone architecture

For the backbone architecture, more architectures can be used, but in the original paper [10], ResNet (residual network) was used. ResNet is a deep neural network oriented residual framework with layers reformulated as learning residual functions with reference to the layer inputs proposed in [11]. ResNet provides frameworks of various depth - ResNet-20, ResNet-32, ResNet-50, ResNet-101 and ResNet-152 where the number suffix represents a number of layers. In [11], even network with more than 1000 layers was proposed, but it is not recommended to use such an extreme solution as its testing result was surprisingly worse than the that of 110 layer network (most likely because of overfitting).

Moreover, the ResNet based backbone was extended by FPN (Feature Pyramid Network), a top-down architecture with lateral connections developed for building high-level semantic feature maps at different scales proposed in [12]. FPN uses the fact that deep convolutional network computes a feature hierarchy layer by layer. By subsampling these layers, we can obtain a different spatial resolution feature hierarchy with multi-scale, pyramidal shape. To achieve strong semantics at all scales, FPN uses bottom-up, top-down pathways and lateral connections to create predictions independently on each level. The feature hierarchy is created by propagating a single scale image through the convolutional network (bottom-up pathway). Because each stage of the network outputs a map of the same size, one level of the FPN is created from output of the last layer of each stage except of the first stage due to its memory claims. The top-down pathway is done by upsampling feature maps from higher levels by a factor of 2 and using nearest neighbour upsampling, and then enhancing them via lateral connections with features from the bottom-up pathway as can be seen in figure 6.

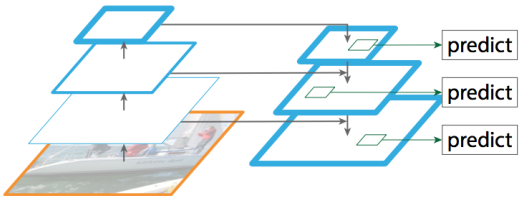


Figure 6. A top-down architecture generating the feature pyramid, source: [12]

B. Head architecture

In the head architecture, there was the most important change to allow the instance segmentation - including the mask branch in parallel with branches for classification and bounding box regression.

The mask branch is a pixel-to-pixel FCN predicting a mask individually for each RoI, where mask is a binary matrix of ones (object location) and zeros (elsewhere).

In figure 7, we can see two implementations of the head architecture. The left one is based on ResNet-C4 (4 stage ResNet) and extends it with the compute intensive fifth stage, the right one is based on FPN which already includes the fifth stage. In these implementations, the last convolution is 1×1 and the other ones are 3×3 and are the ones with preserved shapes; deconvolutions are 2×2 with stride 2 and increase shapes.

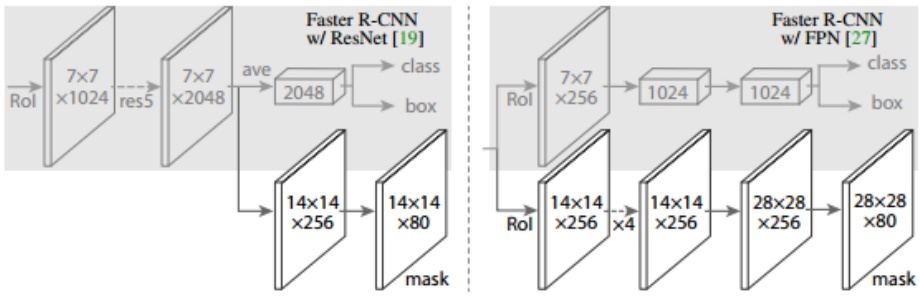


Figure 7. Head architecture, source: [10]
Left: Extended Faster R-CNN with ResNet. *Right:* Extended Faster R-CNN with FPN

C. Loss function

Because there are three individual tasks (bounding box regression, classification, mask), the loss function L is defined as a summation of individual loss functions:

$$L = L_{cls} + L_{box} + L_{mask}$$

L_{cls} and L_{box} are the same as in [8]:

$$L_{cls} = -log(p_u)$$

where:

p_u is a probability of predicting the right class u,
and

$$L_{box} = \sum_{i \in \{x,y,w,h\}} smooth_{L_1}(t_i^u - v_i)$$

$$smooth_{L_1}(x) = \begin{cases} 0.5x^2 & \text{if } |x| < 1 \\ |x| - 0.5 & \text{otherwise} \end{cases}$$

where:

v is a tuple of ground-truth bounding box targets for class u (v_x for a x coordinate, v_y for an y coordinate, v_w for a width, and v_h for a height of the box),

t^u is a tuple of predicted bounding box targets for class u (t_x^u, t_y^u, t_w^u, t_h^u).

A mask of resolution $m \times m$ is generated for each of the K classes, so an output of the mask branch for each RoI is Km^2 dimensional. Mask loss L_{mask} was defined in [10] as the average binary cross-entropy loss of a per-pixel sigmoid function applied to the output of positive RoIs. This approach means that each mask is generated without competing with the others relying on the prediction of class label to select the right one. This is different from a common approach using per-pixel softmax function and a multinomial cross-entropy loss and according to experiments provided by [10], the decoupled prediction gives better results in instance segmentation.

D. RoIAlign

In figure 5, we can see a RoIAlign layer instead of Fast R-CNN RoIPooling from figure 2.

This change was needed because of spatial inaccuracy of RoIPool due to the fact that it was not intended to be used for a pixel-to-pixel alignment. This inaccuracy is caused by quantizations and roundings and RoIAlign avoids it simply by skipping the rounding. To compute the value of the input, a bilinear interpolation at regular sampled locations (usually 4) is used.

According to [10], RoiAlign improved Mask R-CNN mask accuracy by relative 10 % to 50 %.

III. Geomatics field

IN remote sensing we are dealing with huge amount of satellite photos. Though there are many algorithms for an automated classification, there is one promising child which is just growing up and hogging the limelight. This child is called artificial neural networks.

Artificial neural networks for image processing are very strong in an object detection. The object detection in aerial photos could be very useful for example for detecting sick trees in a forest image or for a vehicle detection (airplane, ship, car, etc.). In the world of geographical information systems (GIS), it means elements that can be represented as points. But what about searching for forests? Searching for lakes? Rivers? Oil spills? Do we really want to represent them in GIS as points in their center of mass?

It seems that better representation is in polygons (or lines). But epresenting a detection bounding box as a polygon square does not seem like the right way. So there is the instance segmentation - generating a mask of instances that can be transformed into vector polygons or lines.

A. Implementation

Figure 8. GRASS GIS logo, source: https://grass.osgeo.org/download/logos/

For our implementation, I have decided for GRASS (Geographic Resources Analysis Support System) GIS. GRASS GIS is a multi-platform GIS used for management, analysis, modeling and visualization of spatial data and images. Since GRASS GIS is an open-source software (GNU GPL license), implementing it into this software can provide it to many users. GRASS GIS also contains a lot of modules that can be used to fasten the vectorization and final data cleaning and is also computably fast, that is why GRASS GIS was chosen instead of other open-source GIS like QGIS.

Our implementation was written in Python. There are few reasons for this:

- Although some GRASS GIS modules are written in C and C++, most of them is written in Python.
- Python allows us to use libraries such as TensorFlow[1] and Keras[2].

Our code is based on a Python3 implemetation of Mask R-CNN by Matterport, Inc[3]., written by Waleed Abdulla. I used files `model.py` including the model using ResNet101 as its backbone and `parallel_model.py` enabling the multi-GPU training from the Matterport official repository[4]. This code was published under open MIT license permitting us to use it and modify the code, and Abdulla himself also granted us the permission. `parallel_model.py` was used in its original version, `model.py` was modified, and the rest of code was written by us.

[1] https://www.tensorflow.org/
[2] https://keras.io/
[3] https://matterport.com/
[4] https://github.com/matterport/Mask_RCNN

One is allowed to choose his preferred option - train the network from the scratch or use a pre-trained network. Because training from the scratch is time consuming and the latter is in most cases sufficient, I included pre-trained weights trained on MS COCO[5] provided by Matterport. For our usage it is recommended to use the pre-trained weights for RGB images, otherwise it is recommended to start the training from the scratch.

With the preferred option is connected also another feature - a possibility to train your network in three steps. The first step trains only *heads*, the second one fine-tunes ResNet stage 4 and higher and the last step all layers. This approach prefer training of new weights at the expense of the pre-trained ones in the beginning of the process and seems to allow faster training for images not greatly different from the ones used for pre-training.

Training must be done on 3-band images with masks, where mask is provided for each instance separately. It means that if you want to train classes `soccer`, `tennis` and `golf` and you have image `tainan-01.jpg` containing two instances of soccer fields and one of a tennis field, you have to provide three masks `tainan-01-soccer-0.png`, `tainan-01-soccer-1.png` and `tainan-01-tennis-0.png`. These image-mask sets should be contained in separate folders (a logical one for this example should be `tainan-01`) inside of a training folder. For the best precision, all training images should be in the same resolution as images on which I would like to use the network.

Code may be seen at our Github repository[6].

B. Results

The training dataset was created by using OpenStreetMap[7] and Bing Maps[8]. I downloaded ortophoto tiles and clipped them using query selecting `soccer` (but I prefer to call it *football* in the text, sorry) and `tennis` fields from layer `sports` from OpenStreetMaps. This clip was then transformed into a binary mask separately for each instance.

Our training was stopped after 235 epochs, each consisting of 1500 steps. As I used weights pre-trained on MS COCO dataset provided by Matterport, Inc., I divided our training into 3 steps: 40 epochs of heads training, 60 epochs of fine-tuning with ResNet stage 4+ and 135 epochs of fine-tuning all layers, while the learning rate (0.001) was divided by 10 in the second step and by 100 in the last step. The training was stopped when the total loss oscilated for a long time (several epochs) around 0.58.

Results of testing on a random sample of images not seen by the network (instead of USA I used Prague where I am more competent for considering the correctness) can be seen in the following table. There are percents for correctly recognized fields, for considering another field as a searched field and for not recognizing the searched field. Statistics were made on images with the same resolution as training images and on images with a halved revolution.

Type of field	**Correctly recognised [%]**		**Labeled another field [%]**		**Not labeled [%]**	
	Same resolution as training	**Halved resolution**	**Same**	**Halved**	**Same**	**Halved**
Tennis	99	97	0	0	1	3
Football	80	60	15	20	5	20

Figures illustrating the results visually can be seen in the appendix.

Although I consider the results for tennis fields satisfying, the poor results in football fields are quite disappointing. This can be caused by the fact that some fields are not included in OSM and they are considered as non-target during the training. OSM areas does not also always fit exactly to the field from Bing tile. These facts affect both tennis and football fields, but another one does not - while tennis fields are mostly very professional, football fields are not. And football fields made on a meadow just by erecting two goals are the cause for considering empty fields as football fields.

CONCLUSION

I reviewed Mask R-CNN and the evolution concluding in this framework and I implemented it into an application (which I find as the best possible review). Although I find it very powerful and it allows user to both detect and segment the instance with satisfying results after a training in a tolerable time, my results were sometimes unsatisfying.

However, I believe that using better training dataset can raise the accuracy in really stunning way. Another improvement could be using another architecture for the mask head branch, where I am currently using very basic one. Also when we look at chapter I, we can see the huge progress made within 3 years and I am not afraid to say: Future promises more than ever!

[5]http://cocodataset.org/
[6]https://github.com/ctu-geoforall-lab-projects/dp-pesek-2018
[7]https://www.openstreetmap.org/
[8]https://www.bing.com/maps

REFERENCES

[1] Uijlings J. R. R., van de Sande K. E. A., Gevers T. Selective Search for Object Recognition. *International Journal of Computer Vision 104*, pp. 154–171, April 2013.

[2] Krizhevsky A., Sutskever I., Hinton G. E. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, pp. 1097–1105, 2012.

[3] He K., Zhang X., Ren S., Sun J. *Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition*. ECCV 2014. https://arxiv.org/abs/1406.4729.

[4] Huang J, Rathod V, Sun C, Zhu M, Korattikara A, Fathi A, Fischer I, Wojna Z, Song Y, Guadarrama S, Murphy K. *Speed/accuracy trade-offs for modern convolutional object detectors*. CVPR 2017. https://arxiv.org/abs/1703.06870.

[5] Kemker R., Salvaggio C., Kanan Ch. *Algorithms for Semantic Segmentation of Multispectral Remote Sensing Imagery using Deep Learning*. CoRR 2017. https://arxiv.org/abs/1703.06452.

[6] Felzenszwalb P. F., Girshick R., McAllester D., Ramanan D. *Object detection with discriminatively trained part based models*. IEEE Vol. 32 Issue 9, 2009.

[7] Girshick R., Donahue J., Darrell T., Malik J. *Rich feature hierarchies for accurate object detection and semantic segmentation*. Computer Vision and Pattern Recognition. 2014. https://arxiv.org/abs/1311.2524.

[8] Girshick R. *Fast R-CNN*. Computer Vision and Pattern Recognition. 2015. https://arxiv.org/abs/1504.08083.

[9] Ren S., He K., Girshick R., Sun J. *Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks*. Advances in Neural Information Processing Systems. 2015. https://arxiv.org/abs/1506.01497.

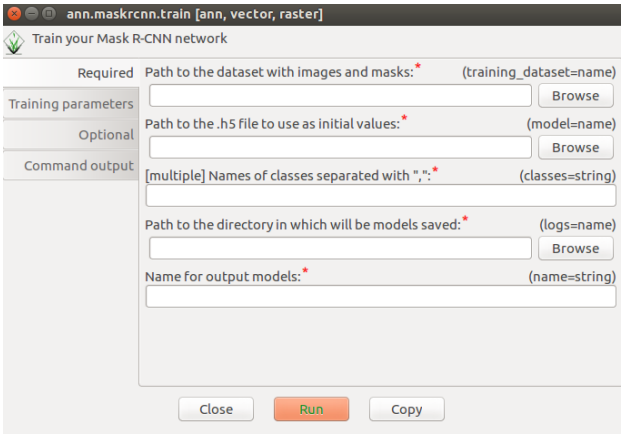
[10] He K., Gkioxari G., Dollár P., Girshick R. *Mask R-CNN*. CoRR. March 2017. https://arxiv.org/abs/1703.06870.

[11] He K., Zhang X., Ren S., Sun J. *Deep Residual Learning for Image Recognition*. CoRR. December 2015. https://arxiv.org/abs/1512.03385.

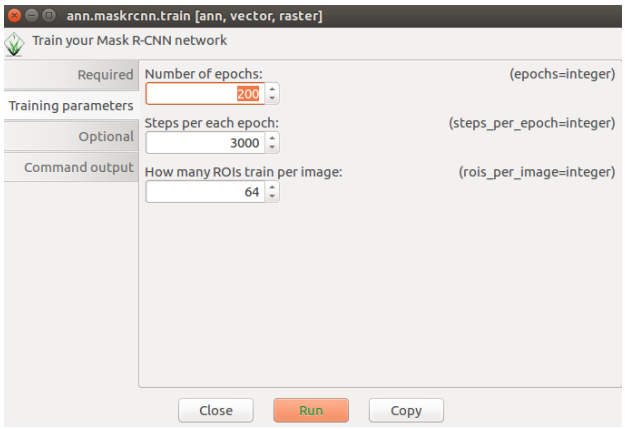
[12] Lin T., Dollár P., Girshick R., He K., Hariharan B. Belongie S. *Feature Pyramid Networks for Object Detection*. CoRR. December 2016. https://arxiv.org/abs/1612.03144.

[13] Parthasarathy D. *A Brief History of CNNs in Image Segmentation: From R-CNN to Mask R-CNN*. Athelas. https://blog.athelas.com/a-brief-history-of-cnns-in-image-segmentation-from-r-cnn-to-mask-r-cnn-34ea83205de4.

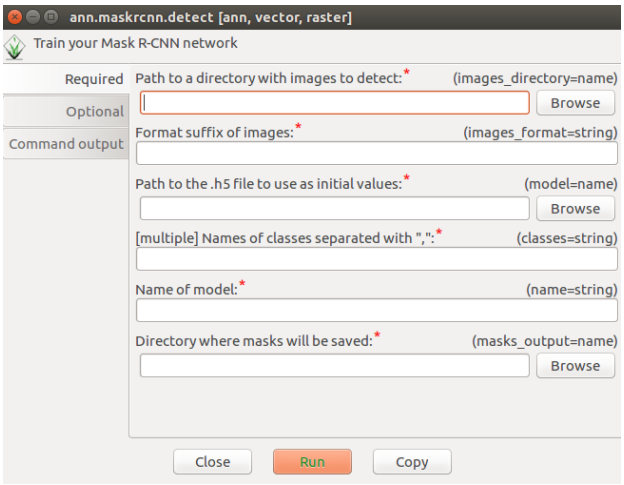
APPENDIX A
IMAGE ATTACHMENT



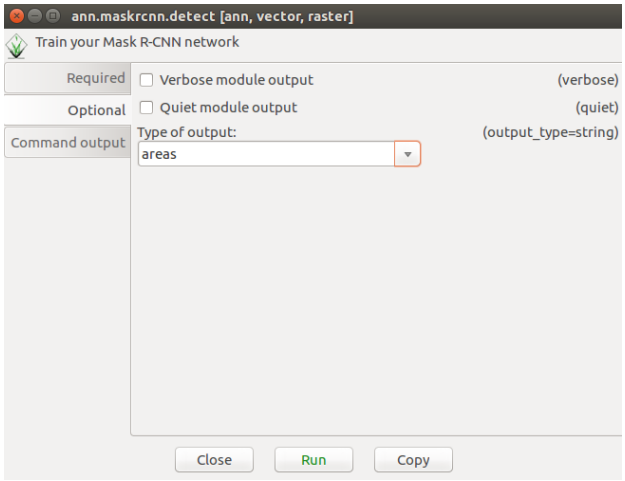
(a) GUI of my module ann.maskrcnn.train



(b) GUI of my module ann.maskrcnn.train



(c) GUI of my module ann.maskrcnn.detect

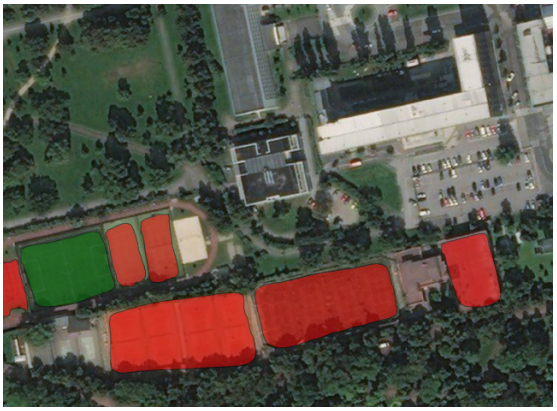


(d) GUI of my module ann.maskrcnn.detect

(e) Original image

(f) Masked image. The left mask could fit better

(g) Original image



(h) GRASS output

(i) Original image

(j) GRASS output

(k) Georeferenced tiles and polygons

(l) Georeferenced polygons (GRASS output without background)