Building segmentation on satellite images

Sébastien Ohleyer
ENS Paris-Saclay
sebastien.ohleyer@ens-paris-saclay.fr

Abstract

Segmentation in remote sensing is a challenging task, especially concerning the classifier capacity to learn on a specific area of the earth and generalize to other regions. Common aerial image datasets propose to split each image in a training part and a test part. In this report, we use another type of dataset, proposing different geographic areas for the training set and for the test set.

We perform segmentation using a recent network architecture: the Mask R-CNN. We compare our performances to baselines results obtained with two more classical networks: a Fully convolutional network and its extension using a Multi-layer perceptron. Our approach outperforms baselines on particular regions with low training time but not on others.

1. Introduction

The development of new acquisition techniques for remote sensing over the last few years raise new challenges, especially in pixel classification on satellite images. More generally, image segmentation faces several problem and one of the most chalenging issue is to be able to train on a dataset and generalize well on another test set. This problem is particularly blatant in satellite images, where images in the test set could be subject to different illumination or even correspond to different areas from images in the training set.

Recently, the Inria provided an appropriate dataset to work on satellite images with training and test images on separate geographic areas [7]. They also come up with some segmentation results based on a Fully Convolutional Network (FCN) and its extension using a Multi-layer perceptron (MLP). These two proposals constitute baseline results for our work.

Our approach consists in two parts. First, we try to reproduce baseline results of [7]. Then, we aspire to study a new Convolutional Neural Network (CNN) architecture, that has proved its efficiency in instance segmentation tasks: the Mask R-CNN (Mask Region-based CNN) [3].

1.1. The Inria Aerial Image Dataset

The dataset is split in two sets and gathers satellite images of US and Austria cities or regions. 36 aerial images of Austin (TX, USA), Chicago (IL, USA), Kitsap County (WA, USA), West Tyrol (Austria) and Vienna (Austria) compose the training set. 36 other images of Bellingham (WA, USA), San Francisco (CA, USA), Bloomington (IN, USA), Innsbruck (Austria) and East Tyrol (Austria) compose the test set. All of these images are open-access, and with corresponding official cadastral records available. They cover varied urban landscapes and illumination. Both subsets contain regions from American and European landscapes with varying density of urban settlements. The validation set is created by removing the first five tiles of each area from the training set (e.g. Austin$\{1-5\}$).

Each landscape are separated between two semantic classes: the *building* class and *not building* class, extracted from the cadaster. Thus, the training set is composed by 180 images and their corresponding masks and the test set by 180 images without masks. For test images, predictions can be submitted at `https://project.inria.fr/aerialimagelabeling/`.

1.2. Evaluation measures

To quantify performances, two evaluation measures are considered: the **accuracy**, defined by the percentage of correctly classifed pixels, and the **intersection over union (IoU)** of the building class. This latter measure is defined as the number of pixels labeled as building in both the prediction and the reference, divided by the number of pixels labeled as building in the prediction or the reference. The IoU measure is a standard measure in standard semantic segmentation [5] as it is particularly fitted for this type of unbalanced datasets.

2. Baseline results

2.1. Fully convolutional network (FCN)

As introduced below, the dataset providers also come up with some segmentation results in [7]. They use a FCN for

		Austin	Chicago	Kitsap Co.	West Tyrol	Vienna	Overall
FCN in [7]							
	IoU	47.66	53.62	33.70	46.86	60.60	53.82
	Acc.	92.22	88.59	98.58	95.83	88.72	92.79
FCN using [1]							
	IoU	52.19	59.62	40.71	51.98	63.40	57.80
	Acc.	91.71	88.42	96.51	97.02	88.15	92.36
MLP in [7]							
	IoU	61.20	61.30	51.50	57.95	72.13	64.67
	Acc.	94.20	90.43	98.92	96.66	91.87	94.42
MLP using [1]							
	IoU	63.61	67.63	54.67	66.11	73.84	68.16
	Acc.	93.70	90.76	97.27	97.86	91.34	94.19
Mask R-CNN							
	IoU	65.63	48.07	54.38	70.84	64.40	59.53
	Acc.	94.09	85.56	97.32	98.14	87.40	92.49

Table 1. Numerical evaluation on validation set (%).

segmentation (see the upper part of Figure 2), presented in details in [6], and train it for 120,000 iterations on randomly sampled patches of the dataset.

For reproducibility purpose and to be able to provide more detailed statistics on the segmentation task, we decide to use an existing Caffe implementation of this FCN (available at [1]). This is not the classic version of Caffe, as it includes a layers coded in C++ to handle the concatenation of an image and its corresponding mask as an input. It also furnishes pre-trained weights for the FCN. Table 1 shows results presented in [7] and obtained using [1].

FCN performs prediction for 180 images in approximately 132 min on a single CPU. As we can see on Table 1, results obtained using implementation [1] are better than results in [7], probably because weights furnished in [1] are different from the ones used in [7].

2.2. Multi-layer perceptron (MLP)

The MLP architecture [6, 7], shown on Figure 2, allows the network to extract and upsample features from the previous FCN. Thus, local and fine features are combined with more large scaled and imprecise features which increases precision of classification. The MLP is trained for 250,000 more iterations. Similarly to the FCN network, we used implementation and pre-trained weights furnished by [1]. We also present both results of [7] and obtained with [1] on Table 2.

Because of its more complex architecture, MLP performs prediction for 180 images in approximately 289 min on a single CPU which is nearly twice longer than the FCN. Also here we observed better results using Caffe implementation [1] than performances in [7].

Hence, we will use these latter results as baselines for our approach consists in using Mask R-CNN [3].

3. Mask R-CNN

The Mask R-CNN network is composed by two networks : a Region Proposal Network (RPN) and a FCN. The RPN takes the whole image as input and output the image with bounding box proposals. According to the RPN, each bounding is supposed to be an object. With this proposal, the FCN performed a segmentation as in Section 2.1. This network is more complex than the two previous one and we choose an existing open-source implementation [2]. It is designed to train on COCO dataset [4], which is clearly different from ours. We manage to reuse without modification the part of [2] corresponding to the network, however we had to change the pipeline to preprocess images.

3.1. Choices of implementation

Initialization. Concerning the weights initialization, we use pre-trained weights on COCO, excluding some of them as the COCO dataset has 81 classes and ours has only 2 classes. This option is handle by implementation [2].

Input image size. Mask R-CNN has many hidden layers, hence the training on CPU can be very long. We decided to use AWS GPU p2.xlarge. This is a 12GB GPU and it can typically handle 2 images of 1024×1024px. As Inria images are 5000×5000px, we split them into 25 patches of size 1024×1024px. It yields to a dataset of 4500 images which is very large. Due to the memory limit of the GPU, the batch size is set to 2 and by testing the network, a step size last 4.5 secs. If we want to pass the entire dataset, an epoch would last more than 2.5 hours. That is inconceivable considering our limitation on GPU use.

Hence, we create several subsets of the dataset containing 180 images of size 1024×1024px - one localized patch on every image. Each epoch lasts approximately 7 mins which is more acceptable.

Training. We train each subset for 40 epochs splitted as follows:

1. 10 epochs for RPN training (freezing other weights), learning rate set to 0.001.
2. 10 epochs for FCN training (freezing other weights), learning rate set to 0.001.
3. 20 epochs for all layers, learning rate set to 0.0001.

This strategy is inspired by the training process on COCO dataset in [2]. We provide the complete configuration in Appendix B.2.

3.2. Model selection

Contrained by the allowed time on the GPU, we train on only 4 subsets and compare different losses: the complete loss, the RPN box loss, the RPN class loss, the Mask R-CNN box loss, the Mask R-CNN class loss, the Mask R-CNN mask loss and the corresponding ones on validation set. One training clearly attains lower losses than others and we choose it to compare with baselines.

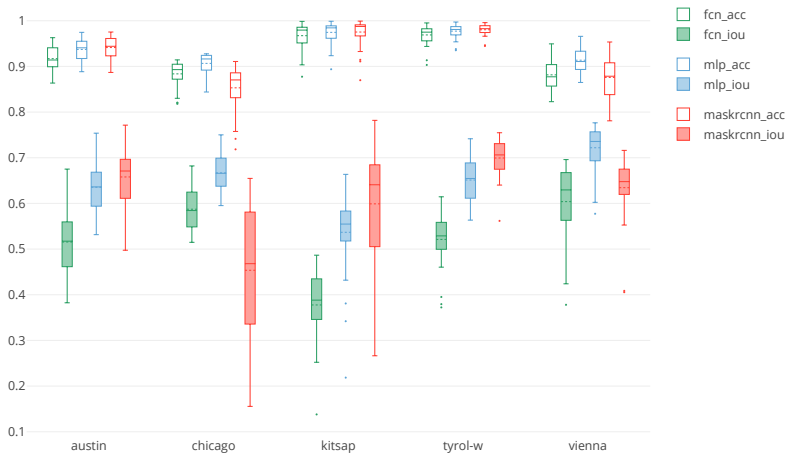

Figure 1. Box plots for accuracy and IoU for each method (FCN, MLP and Mask R-CNN) on the 36 images of each city. The dashed line represent the mean and solid line the median.

3.3. Results

While training was only on subsets, we compare performance on the 5000×5000px original images of the validation set. As Table 1 shows, Mask R-CNN achieves better perfomances than MLP on Austin and West-Tyrol, similar on Kitsap and lower on Chicago and Vienna. On average, our approach outperforms FCN but not MLP. Figure 1 presents box plots of accuracy and IoU for each method on each city. According to this figure, the variances of evaluation measures are higher for Mask R-CNN than for FCN and MLP. The training of Mask R-CNN is surely not good enough which is not surprising because we trained Mask R-CNN on only 1/25 of the training set while FCN and MLP where trained on the entire training set.

Figure 4 presents qualitative results on West Tyrol and Chicago. On the West Tyrol image, Mask R-CNN outperforms MLP and FCN while on the Chicago image, the MLP outperforms other methods.

We can observe that MLP gets smoother predictions than FCN which match better with building outlines. The Mask R-CNN also has the property to give smooth predictions correspond well to the ground truth. West Tyrol is a low density urban settlement area and the Mask R-CNN performs well on these satellite images. However, as Figures 1 and Table 1 stress our model failed to predict well on very dense cities (e.g. Chicago). On the last Mask R-CNN prediction of Chicago (Figure 4), several close buildings are completely omitted. Looking deeply in the architecture, we observe (Figure 5) that buildings are detected by the RPN but with low probability. Hence, they are deleted at the next step (Figure 6). To avoid that, we try to tune hyperparameters like `DETECTION_MIN_CONFIDENCE` or `RPN_NMS_THRESHOLD` but we did not have enough time to come up with interesting conclusions.

4. Concluding remarks

Mask R-CNN is an interesting and promising network architecture for semantic segmentation. Looking further, it even allows to perform instance segmentation on satellite images (we did not deal with this aspect in this project). However, it requires more training and better hyperparameters tuning than what we were capable of.

To assess generalization performances on the test set, a proposition (under the name "Fantomas") was submitted on the dataset webpage, but the results are not instantaneous and we are waiting for them.

References

[1] Caffe Remote Sensing for Python. `https://github.com/emaggiori/CaffeRemoteSensing`.

[2] Mask R-CNN implementation for Python. `https://github.com/matterport/Mask_RCNN`.

[3] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask r-cnn. *arXiv preprint arXiv:1703.06870*, 2017.

[4] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[5] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3431–3440, 2015.

[6] E. Maggiori, Y. Tarabalka, G. Charpiat, and P. Alliez. High-resolution semantic labeling with convolutional neural networks. *arXiv preprint arXiv:1611.01962*, 2016.

[7] E. Maggiori, Y. Tarabalka, G. Charpiat, and P. Alliez. Can semantic labeling methods generalize to any city? the inria aerial image labeling benchmark. In *IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*. IEEE, 2017.

Appendix

A. Fully convolutional network (FCN) and extension with Multi-layer perceptron (MLP).

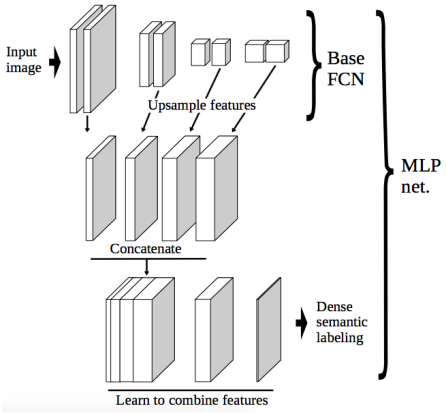

Figure 2. FCN (upper part of the figure) MLP (complete figure) network for semantic labeling (taken from [7]).

B. Mask R-CNN

B.1. Network architecture

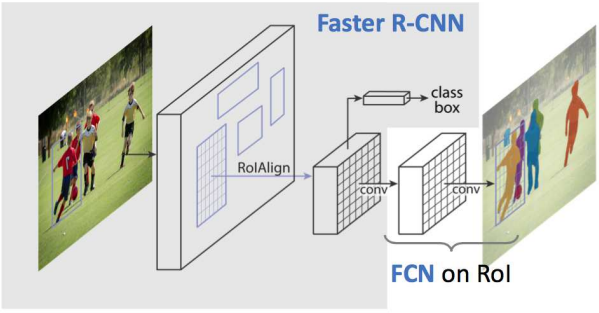

Figure 3. Mask R-CNN network architecture (taken from [3]).

B.2. Training configuration

BACKBONE_STRIDES	[4, 8, 16, 32, 64]
BATCH_SIZE	2
BBOX_STD_DEV	[0.1 0.1 0.2 0.2]
DETECTION_MAX_INSTANCES	100
DETECTION_MIN_CONFIDENCE	0.7
DETECTION_NMS_THRESHOLD	0.3
GPU_COUNT	1
IMAGES_PER_GPU	2
IMAGE_PADDING	True
IMAGE_SHAPE	[1024 1024 3]
LEARNING_MOMENTUM	0.9
LEARNING_RATE	0.001
MASK_POOL_SIZE	14
MASK_SHAPE	[28, 28]
MAX_GT_INSTANCES	100
MEAN_PIXEL	[123.7 116.8 103.9]
MINI_MASK_SHAPE	(128, 128)
NUM_CLASSES	2
POOL_SIZE	7
POST_NMS_ROIS_INFERENCE	1000
POST_NMS_ROIS_TRAINING	2000
ROI_POSITIVE_RATIO	0.33
RPN_ANCHOR_RATIOS	[0.5, 1, 2]
RPN_ANCHOR_SCALES	(32, 64, 128, 256, 512)
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	90
TRAIN_ROIS_PER_IMAGE	200
USE_MINI_MASK	True
USE_RPN_ROIS	True
VALIDATION_STEPS	20
WEIGHT_DECAY	0.0001

These configuration variables exactly correspond to [2]. Some hyper-parameters has been changed to match our training set like IMAGE_SHAPE, MINI_MASK_SHAPE, STEPS_PER_EPOCH and VALIDATION_STEPS.

C. Qualitative results on training test

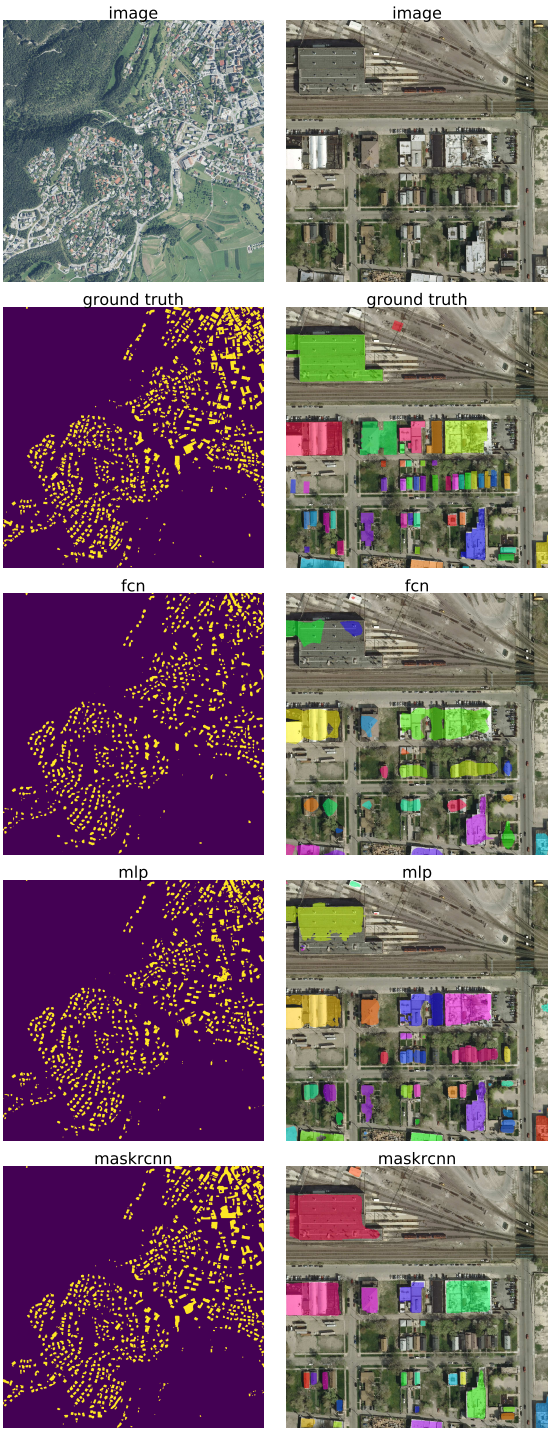

Figure 4. (Left, from top to bottom) Ground truth 5000×5000px image of West Tyrol, segmentations using FCN, MLP and Mask-RCNN. (Right, from top to bottom) Ground truth 1024×1024px patch of Chicago, stacked with segmentations using FCN, MLP and Mask-RCNN (colors are randomly assigned).

D. RPN outputs: Regions of Interest

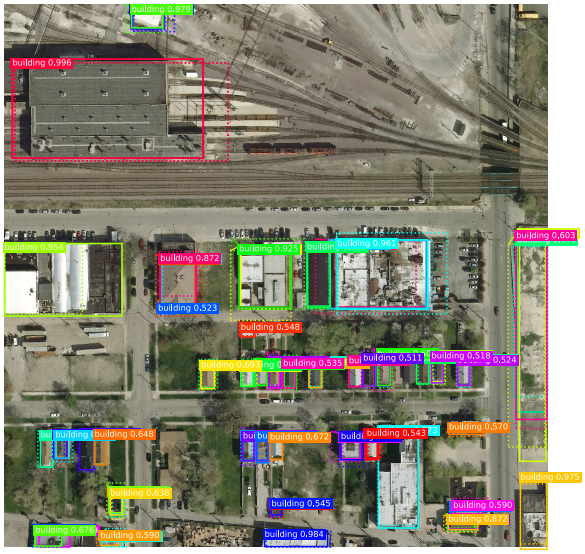

Figure 5. Regions of Interest on a 1024×1024px Chicago patch output by the RPN. A refinement is performed on each region to match building outlines (dashed lines boxes before refinement and solid lines boxes after refinement). Numbers are the probability of the detection in the box to effectively correspond to a building (colors are randomly assigned).

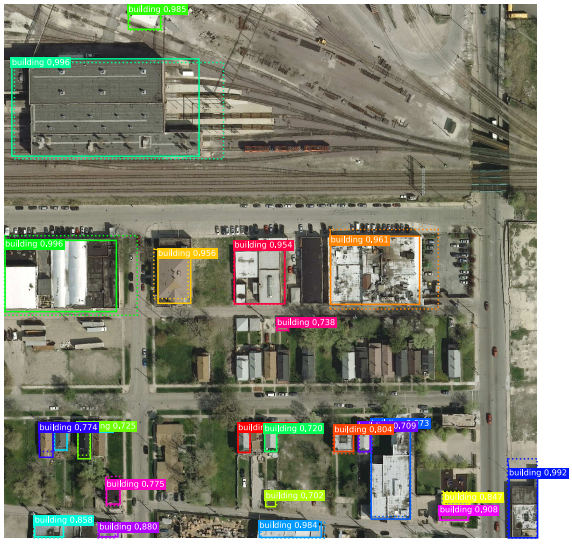

Figure 6. Regions of Interest on a 1024×1024px Chicago patch output by the RPN after low confidence detections filtering and non-max suppression. Numbers are the probability of the detection in the box to effectively correspond to a building (colors are randomly assigned).