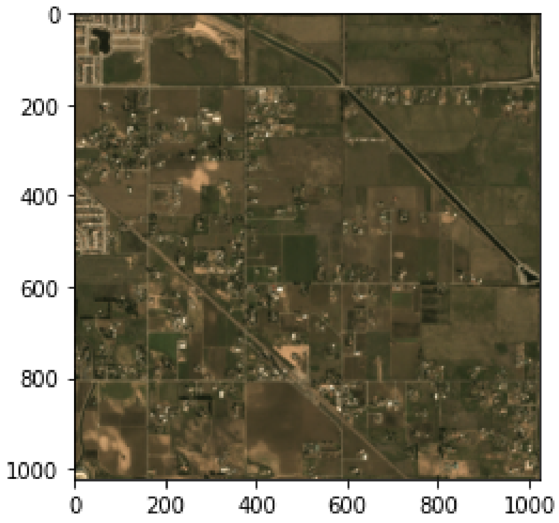
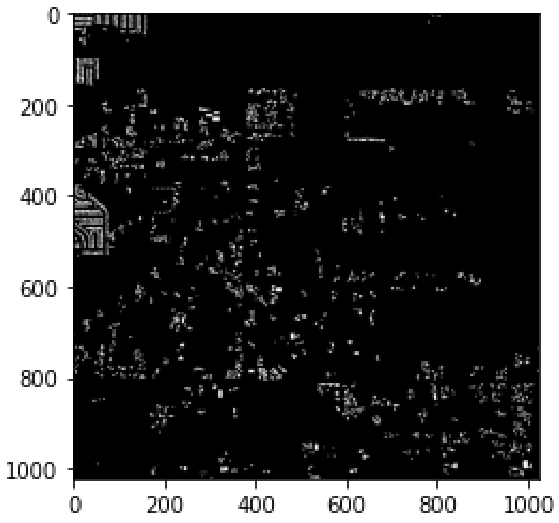
Ecole Polytechnique fédérale de Lausanne

Image Processing for Earth Observation

Detecting infrastructure development

Group 17
Francesco Ballestrazzi
Thomas Cros





December, 18th, 2020

Contents

1 Introduction

1.1 Project definition

This project is based on data from the SpaceNet 7 challenge, called Multi-Temporal Urban Development. The goal is to solve (using solely satellite imagery) two different tasks: building detection and change detection.

The former consist in identifying where buildings are in a satellite image. The latter consist in spotting differences, in terms of buildings, between two images of the same area at two different months.

1.2 Motivation

The challenge is motivated by the difficulty to perform image processing with low resolution images (4m), at low frequency, including seasonal , atmospheric, and lighting effects. Yet the ability to localize and track the change in building is crucial for many applications such as disaster response or humanitarian efforts.

1.3 Literature review

This challenge is a great opportunity for us to apply our knowledge on the well-known topic of building detection using satellite images.

Building were initially detected based on their geometrical properties, mostly edges [6]. In most recent years, the development of convolutional neural networks (CNN) disrupts how people performed image detection. A well-know example is the success of AlexNet in the 2012 ImageNet Large Scale Visual Recognition Challenge [3]. Since then, more and more complex architectures were developed to improved performance and computational efficiency. In this context, the dataset provided by SpaceNet 7 is good opportunity to tackle these methods. As described below, the dataset offers pre-labeled images than can be used to performed supervised learning.

In the section 3, we describe the CNN architecture that we used to perform semantic segmentation. This architecture is called U-net, a CNN originally developed in 2015 for biomedical image segmentation [2]. Since this architecture proved to be very good for both semantic segmentation and change detection[5][4] we decided to use it for both our tasks.

1.4 Data description

The SpaceNet 7 dataset (https://spacenet.ai/sn7-challenge/) contains more than 100 locations images (only 60 used, the training set of 707 images), captured by Planet's satellites every month during 2 years.

Samples are 1023x1023 RGBA images with a 4m resolution. Format is GEOTiff.

Building labels information is stored in GEOJson files that contain all the polygons that delineate buildings in the images.

The Spherical Mercator projection coordinate system EPSG:3857 is used for both the images and the labels.

1.5 Frameworks and libraries

The programming language used in the project is Python and the code have been executed on Google Colaboratory.

For data pre-processing we used Opencv, Solaris and Numpy, while the neural network is implemented in Keras.

1.6 Flowchart

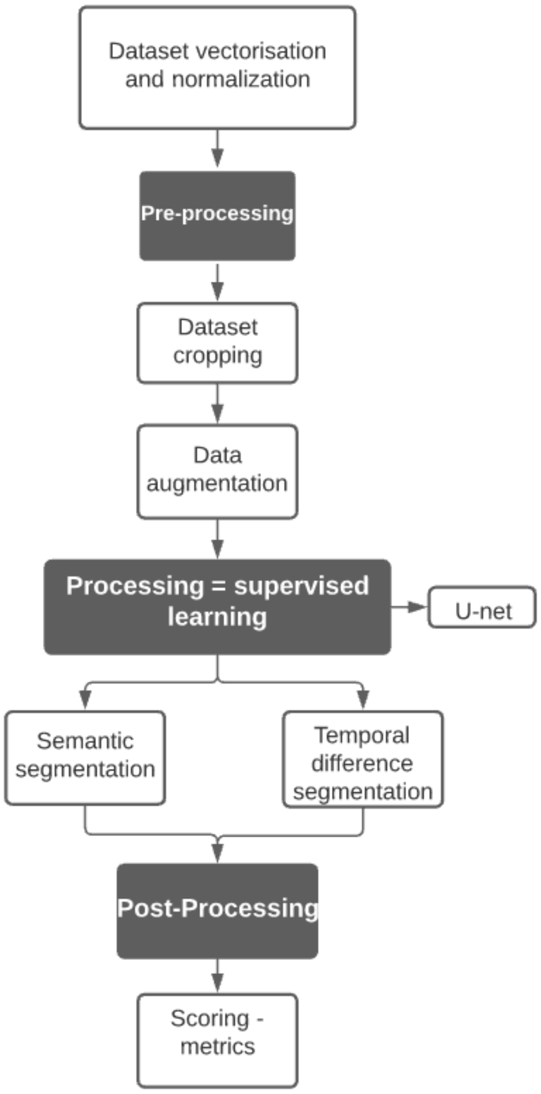


Figure 2: Flowchart of the project

2 Pre-processing

In this section we describe the pre-processing needed to work with the data collected from the SpaceNet challenge.

2.1 Initial data analysis and transformations

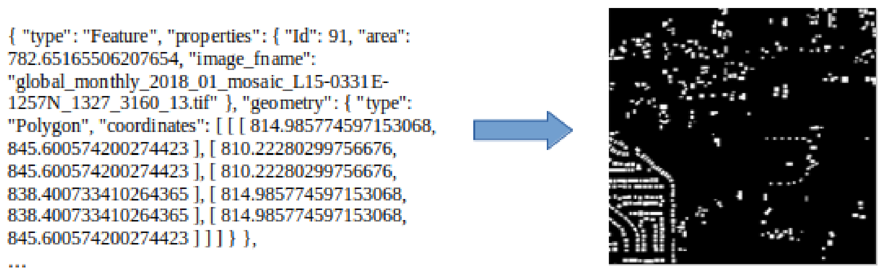
The data from the SpaceNet challenge is available as folders each containing images and labels for each geographical location.

The images are .tiff images and the labels are given in GEOJson format.

In order to simplify the next steps of the project, both images and labels have been converted to Numpy arrays.

In particular, images have been transformed to tensors of shape (1023,1023,3) where the the first two dimensions are the size of the image in pixels and the 3rd dimension represent the number of channels. All these vectors have also been normalized by dividing every pixel value by 255.

Labels have been converted from GEOJson to tensors of shape (1023,1023,1) with binary values 0,1. Every pixel with value 1 indicate the presence of a building at those coordinates. The conversion have been done using the library Solaris, that provide a Python API and that take care to project the resulting mask/vector in the right coordinates.



```
{ "type": "Feature", "properties": { "Id": 91, "area":
782.65165506207654, "image_fname":
"global_monthly_2018_01_mosaic_L15-0331E-
1257N_1327_3160_13.tif" }, "geometry": { "type":
"Polygon", "coordinates": [ [ [ 814.985774597153068,
845.600574200274423 ], [ 810.22280299756676,
845.600574200274423 ], [ 810.22280299756676,
838.400733410264365 ], [ 814.985774597153068,
838.400733410264365 ], [ 814.985774597153068,
845.600574200274423 ] ] ] } },
...
```

Figure 3: Example of the transformation form GEOJson to binary mask

Another label mask is created from these to be used in the second part of the project, the building change detection. To do so, for each geographical location, the pixel-wise differences between two sequential label masks (more recent image - less recent image) have been computed. Since the temporal changes among two sequential months are not sufficient to provide reasonable examples to the network, we combined images with temporal difference greater than one month, and up to two years.

2.2 Data setup

We divided the full dataset (60 geographic locations) into three separate sets: train (30 locations) and test (20 locs). Each location contains around 24 images.

2.3 Data cropping

To overcome technical issues we faced using Google Colab, the images have been cropped (from 1023x1023 to 200x200 pixels) to reduce the computational power required to process them.

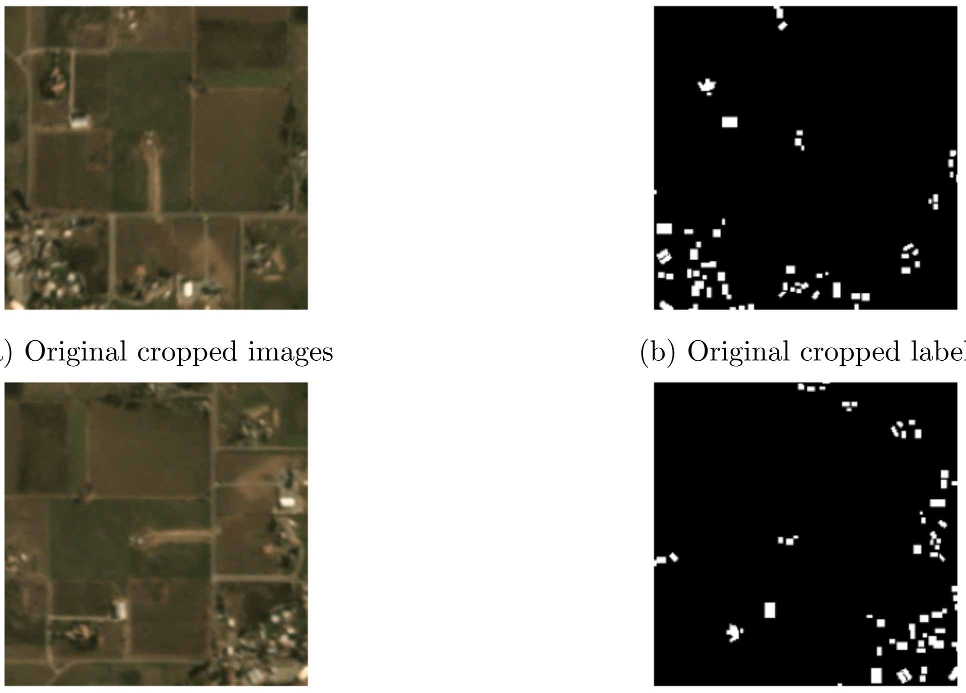
Two different cropping methods are used for semantic segmentation and temporal difference segmentation.

A unique random crop on each of the 707 images was used to study semantic segmentation, in particular the data-augmentation's effect on the algorithm's performance.

For temporal difference segmentation, each image is partitioned in 200x200 pixels patches. Each patch is saved for model training only if the proportion of buildings in it is bigger than a threshold (manually determined). This second method allows to avoid uninformative parts of the image. However, due to the memory limitations of the platform, we used only a small subset of the locations for this second task.

2.4 Data augmentation

The dataset size is increased using data augmentation techniques such as flipping (up-down and left-right), rotating or modifying the contrast ($image = mean + (image - mean) \cdot factor$). These operations were performed separately, such that each cropped's images is transformed by one of the four operation and added as a new image, which double the original dataset. These operations are performed to generate new data while increasing the robustness and reducing overfitting.
Figure 4 shows the output of a rotating operations performed on a cropped images.


(a) Original cropped images

(b) Original cropped label

(c) New rotated image

(d) New rotated label

Figure 4: Example of data augmentation used in the project, rotating by 90° in this case

2.5 Edge augmentation

We thought about highlighting building's edges to simplify their segmentation using contour reinforcement. Contour reinforcement is performed by applying by a convolutional transformations with a kernel equal to the sum of the Laplacian filter and the identity matrix such that:

$$k = \begin{pmatrix} -1 & -1 & -1 \\ -1 & 9 & -1 \\ -1 & -1 & -1 \end{pmatrix}$$

To test the results, the operation was performed on one of the channel of one of the first images, see figure 5. The result obtained is quite disappointing as he does not properly highlight edges around buildings. We assume that the images low resolution was the reason of this bad results. Consequently, contour reinforcement was discarded in our project.

(a) Original images (b) New image using contour reinforcement filter

Figure 5: Application of contour reinforcement filter

3 Process

3.1 Semantic segmentation

The semantic segmentation task consist in assigning a label/class to each pixel of an image. For the first part of our project we have only two labels, value one where there is a building in the image and value zero in any other case.

To solve this task we used a deep neural network architecture typically used in these situations, called U-net. We don't describe all the details of the network (that can be found in the attached jupyters), but the structure of our architecture is similar to the one in the figure below.

The peculiar skip connections (in grey in the figure) are a crucial factor to make UNET appropriate to segmentation, allowing to maintain the spatial structure of the image from the input to the final segmentation mask.

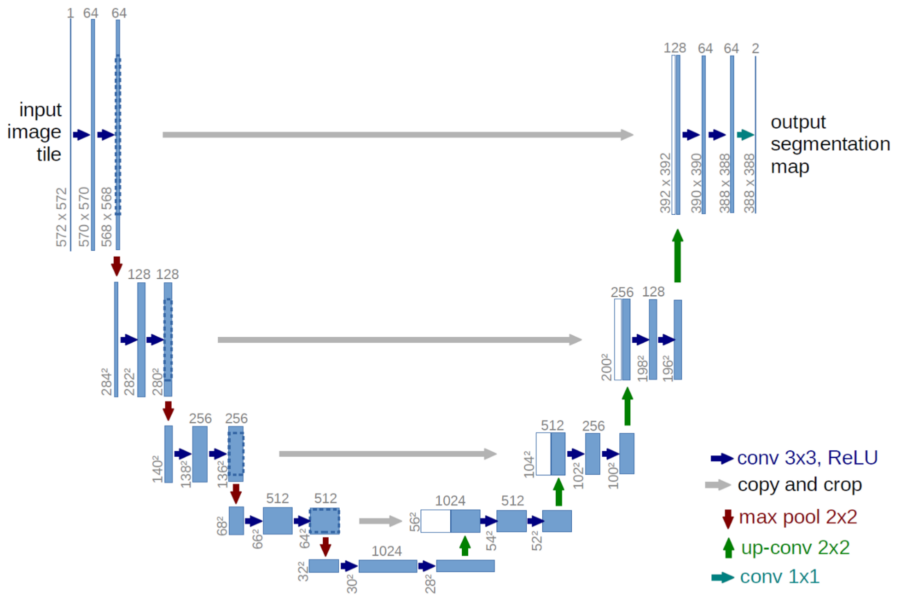


Figure 6: Example of U-net architecture

3.2 Temporal difference segmentation

For the second part of the project the goal is to spot changes between images of the same location taken in different months.

The labels for segmentation are two also in this case, but class 1 correspond to a building temporal change from one month to another, i.e the construction of a new building. For this project, demolitions of buildings are not considered as changes.

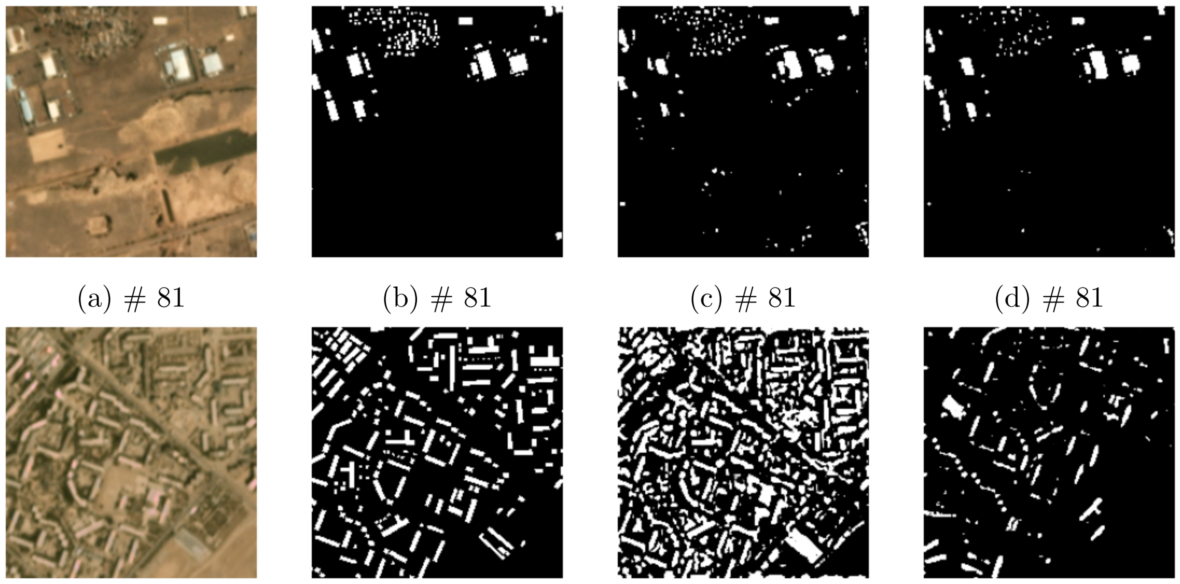
The network used to solve this task is the same U-net as before, except for the input layer. Since we need to pass two images to the network, those are stacked on the channel axis, giving a final image of shape (1023,1023,6).

4 Results and discussions

4.1 Results

4.1.1 Semantic segmentation

The results is shown for two types of dataset. The first one, called Regular, is based on 707 cropped images, while the second one is composed of 1414 images where the 707 extra images were made using data augmentation. To compare the performance of the two datasets, a prediction was performed using 437 new images. The difference with the true label was then calculated and mean to compute the mean error. As shown in table 1, the perfomance of the second dataset is much better to predict the label. The mean error was divided by 4.5 from 41% to 9%. This improvement can also be visualized using figure 7. For images 81 for example (first row), the predicted label using augmented dataset (d) looks more like the true label (b) than the predicted label using regular dataset (c).



(a) # 81 (b) # 81 (c) # 81 (d) # 81

(e) # 15 (f) # 15 (g) # 15 (h) # 15

Figure 7: Two examples of semantic segmentation results obtained. (a) and (e) are the orginal images, (b) and (f) and the true label, respectively, (c) and (g) and the predicted label using regular dataset, respectively, and (d) and (h) are the predicted label using augmented dataset

Training dataset	Nb of images	Mean error [%]
Regular	707	41
Augmented	1414	9

Table 1: Semantic segmentation mean error for the different training dataset

4.1.2 Temporal difference segmentation

Since the dataset was already big enough with the new type of patch selection introduced for the second part of the project, no data augmentation has been done. The model has been trained on a set of 1400 thresholded patches. Training has been (manually) stopped as soon as the validation error started increasing. The results on the test set are shown below.

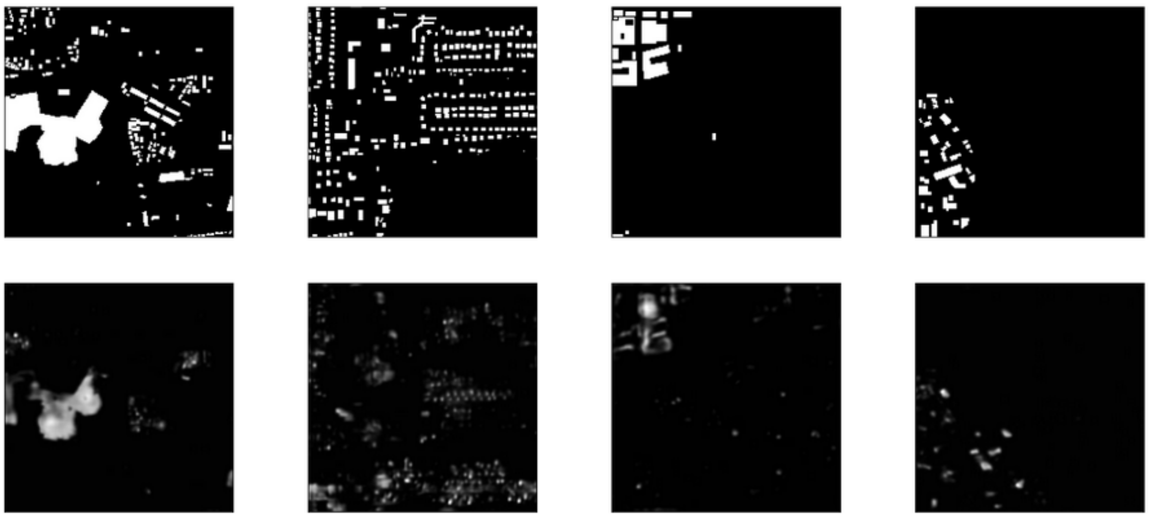


Figure 8: Results on unseen data: the first row shows the ground truth, the second the predictions of our network.

Since it was hard to load and train on more data, the network is not really good at generalizing. However despite its simplicity it's quite surprising how it is able to spot main areas of change.

4.2 Discussions

The models developed in this project represent a first step towards semantic segmentation. We saw that data augmentation improved the performance of the prediction and is a good way to expand the dataset with less risk of over-fitting.

Many limitations of our approach arise from technical limits. In particular, the memory in Colab turned out to be not enough to load the 1023x1023 images and train the model at same time. Cropping the images have been a quick way to solve this, but with some information and context loss. Maybe this could have been solved by loading and processing images one by one in memory through a "generator". Since more data would also mean better generalization, this is probably the most important thing to solve to achieve better results. Moreover the use of Colab limited the possibility to tune hyperparameters correctly, thus many of those just have values that are typically used in semantic segmentation, but not specific to our problem.

It was also difficult to determine which pre-processing step would help the model to predict. As shown, contour reinforcement output was not as relevant as we wanted. Also, others techniques could have been used such as : pre-training the NN (using transfer learning), combining data augmentation or use others methods such as up-sampling the original images.

Since the results of the challenge have been published [1] we had the opportunity to confront the winning techniques with the ones we used. One possible improvement would be to work more on post-processing. As mentioned earlier, the threshold used to compute binary prediction output was set manually. We could have determine this parameter using an extension in the neural network for example. However, 4 teams out of 5 used a U-net, even if combined with other models, and thus this appears to be a good choice of architecture to solve our task as well.

Another weak point of our analysis are the metrics we used to evaluate the different versions of our model. Only the Binary Cross-entropy loss, between the ground truth and the output of the model, has been optimized. A possible improvement in this direction would be to consider more meaningful metrics for segmentation, like Intersection Over Union (IOU).

To conclude, we are well aware that this is far from being a complete and robust model, however, given the limitations imposed by the data and technical aspects, the results are promising and encouraging.

References

[1] Adam Van Etten. "The SpaceNet 7 Multi-Temporal Urban Development Challenge: Announcing the Winners". In: (2020).

[2] Olaf Ronneberger; Philipp Fischer; and Thomas Brox. "U-Net: Convolutional Networks for BiomedicalImage Segmentation". In: (2015).

[3] Alex Krizhevsky; Ilya Sutskever; Geoffrey E. Hinton. "ImageNet Classification with Deep Convolutional Neural Networks". In: (2012).

[4] Raveerat Jaturapitpornchai et al. "Newly Built Construction Detection in SAR Images Using Deep Learning". In: *Remote Sensing* 11 (June 2019), p. 1444. DOI: `10.3390/rs11121444`.

[5] Lazhar Khelifi and Max Mignotte. *Deep Learning for Change Detection in Remote Sensing Images: Comprehensive Review and Meta-Analysis.* 2020. arXiv: `2006.05612 [cs.CV]`.

[6] Parvaneh Saeedi; Harold Zwick. "Automatic Building Detection in Aerial and Satellite Images". In: (2008).