# Project Summary: instance segmentation of individual buildings by category from an aerial/satellite image.

## Overview

In this project the objective here is to automate building extraction and identifying their type. The idea was to apply Mask-R-CNN method to achieve instance segmentation but the final approach implemented in this project is doing : **Semantic segmentation of the images with U-net like DL model as a backbone, a Fully Convolutional Network that outputs a pixel-wise segmentation of the image**. The rest of the report will explain a little bit this choice (besides the deadline pressure and the limitation of my hardware computational capacity).

## Limitations:

- Because of the time constraint and the limited capabilities of my computer, also how challenging the data is, I chose to proceed with a solution that performs a semantic segmentation that detects buildings from an image. Before sticking to this solution I tried to build an instance segmentation algorithm that uses mask r-cnn method; I had some difficulties while training the model which led me to drop it as an option (using google colab probably might solve the issue but I didn't have time to fully investigate this option)

- Most of the previous works in the field of satellite imagery did not specifically train their models to differentiate between different types of buildings [2][3]; they rather focus differentiating between different types of infrastructures and landscapes [4][6]. After analyzing the dataset, I decided to give up on the task of identifying the building type ("buildings" / "Houses" / "Sheds/garages").

## Dataset preparation and processing:

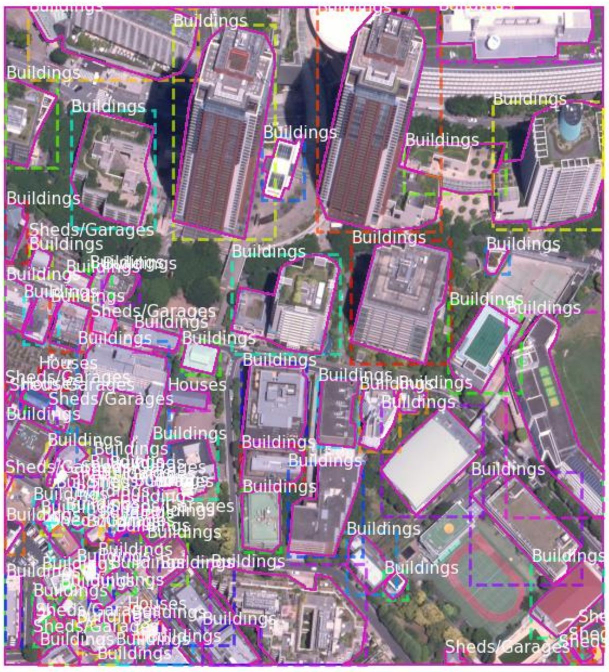
- Annotations :

2D bounding polygons – Polygon annotation for city layout plan and urban management. They look like cuboid annotations since buildings in images appear to be 3d (as if they were rendered image of a virtual map layout). There is no issues with the annotations other than some slight overlapping between adjacent instances.

- Images:

Satellite image are large in size, to make it easier to work with I reduce its size to 900x900pxls and divided each image into 9 crops of size 300x300pxls. The slight distortion happening in the process didn't affect the quality of the results.

When looking at objects in the images, they look obviously too small, for instance 'houses' and 'sheds'. These objects usually big and most methods (ML or non ML based methods) are tuned to that condition. The number of these instances in every image is rather too low compared the instances labeled as 'buildings'. It was clear that identifying the class of these instances will be difficult. That was one the reason that led me to ignore the classification task and focus on the segmentation task. The photo below show a sample of the ground truth.



The total image of the map seems a little over exposed which caused some scattered shadows. Fortunately the image quality still good enough to work with. But that's not the case for the position of the satellite sensor will change the type of the images. In our case it's an 'OFF Nadir position (check further info part below). An 'off nadir' position will have building hiding smaller objects.

The previous points explain a little bit why this data is a challenging and the limitations stated above.

## AI pipeline:

Every object is represented by few pixels, few features, which means classical computer vision methods may not work. The context around satellite imagery analysis will always change; this why deep learning has proven great solution for such use cases. They turn toward CNN or an evolution, such as Capsule Networks.[1]. The network won't be able to process all of it at once (GPU/CPU will run out of memory during training). I had to divide it into smaller tiles before feeding it to the NN.

A training data sample is an RGB image with a pixel-wise binary masks and bounding boxes for each instance in the image. The masks are derived from the metadata. Each mask is generated as an array of shape (width, height, number of instances). An additional bounding boxes information is generated too since initially it was needed for Mask R-CNN method [5].

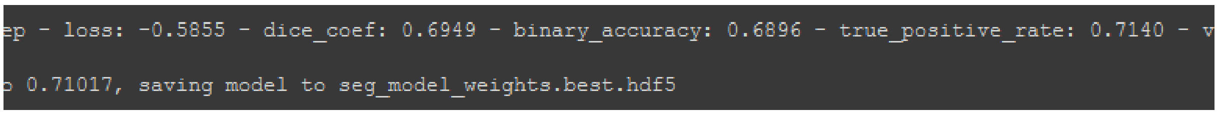
While going through different architecture most of the research studies end up in selecting **U-Net** architecture. It is understood that U-Net architecture is designed for semantic segmentation. U-Nets have an ability to learn in environments of low to medium quantities of training data. The final model structure is altered slightly.

In order to find the best performing configuration, different parameters and training setups were tested and tuned, such as comparing the Adam and SGD optimizer, testing different learning rates and changing weights of the loss function. Finding the optimal learning rate can be problematic. For that I used a learning rate schedule. A learning rate schedule is a way to decrease the learning rate as the model converges automatically, by scheduling a ‘ step decay’.

To optimally train a U-Net, one needs to considerably augment the dataset. Thus every time an image is fed to the network; it is randomly rotated and mirrored before proceeding to train on that image.

## Evaluation

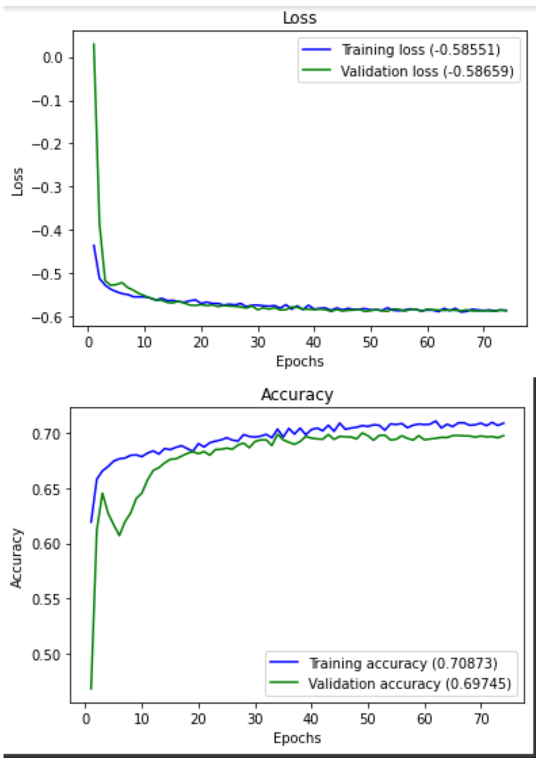
When using the Adam optimizer and Dice Loss, the result is presented as the F1 score, along with the precision and recall. (The final accuracy is up to 72% last time trained)



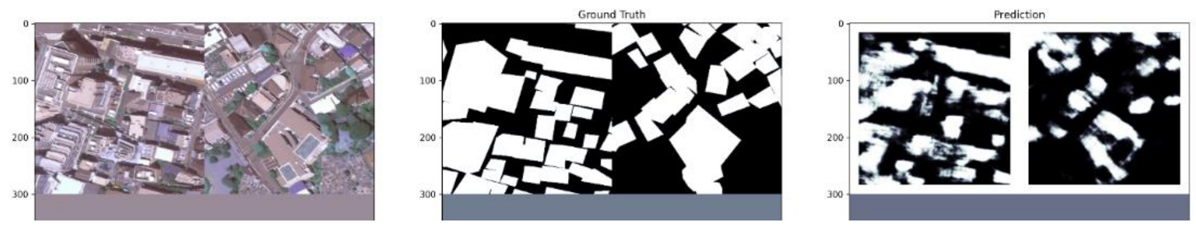
```
ep - loss: -0.5855 - dice_coef: 0.6949 - binary_accuracy: 0.6896 - true_positive_rate: 0.7140 - v

o 0.71017, saving model to seg_model_weights.best.hdf5
```

The networks were trained until the training loss flattened or when signs of over-fitting occurred. During training both the training and validation losses were monitored.   The weights with the lowest validation loss were saved and used for evaluation.



During testing phase, the model prediction is inferred on one image at the time, where the predicted binary mask array is converted into polygons. The geometries of all polygons in one image are stored in a separate file. The following figure is an example of the model prediction before being processed and converted into vectors.



**Conclusion:**

To sum up, I have built a pipeline for training and evaluating neural networks; refined the neural architecture and configuration to best solve for a given problem. I managed to use reasonable computational resources. Building neural networks is an iterative process where one needs to start somewhere and slowly increase a metric / evaluation score by modifying the neural network architecture and hyperparameters (configuration). The work described above, is mostly done in function of the data and time at hand; it goes without saying that the amount of training data is quite scarce for having a good accuracy.

**For further improvements!**

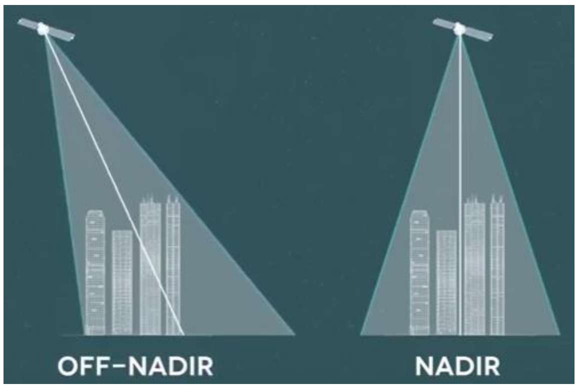
- Further tuning on learning rate and batch size ; further training may also help.
- I think it would be interesting to test different combination of data transformation (similar to the one in the official paper of U-net)
- a U-Net can be adapted to recent research, since its architecture is quite similar to the PSPNet or the One Hundred Layers Tiramisu, which are recent improvements such as for when dealing with more data. To have a good neural network, one may not only have to train it. One may not only have to adjust the learning rate and hyperparameters. One may want to also adjust the whole architecture of the neural network, automatically.

**Further info:**

- The geo-reference issue: the position of the satellite when taking an image of the earth is important to consider geo-referenced images. These images were recorded in an elliptical frame and then the coordinate were projected into flat plan. From this process we can define the ground resolution; each pixel record a certain distance of the surface, which varies depending on the satellite. This information is usually available in the metadata (even further info can be available such as sun angle, atmosphere conditions, the satellite position …) may help improving the results

- The satellite records an image of a specific area at a certain time. The position of sun is important because the image may be overexposed or it has huge shadows.

The position of the satellite sensor will change the type of the images. In our case it’s an OFF nadir position.



- About the use of pre-trained models : According to Andrew Ng : a neural network architecture, despite it might be good and well-fitted at one task, needs to be tuned again for fitting another task… one must not be hesitant on changing the architecture of a neural network: trying new things along the process is good.

**Reference:**

[1] Sara Sabour and Nicholas Frosst and Geoffrey E Hinton, Dynamic Routing Between Capsules,2017 arXiv

[2] Adam Van Etten, Daniel Hogan ; The SpaceNet Multi-Temporal Urban Development Challenge; Feb 2021 arXiv

[3] spacenet.ai - SpaceNet challenges

[4] Volodymyr Mnih and Geoffrey E. Hinton ; Learning to Detect Roads in High-Resolution Aerial Images; 2010 University of Toronto

[5] Waleed Abdulla. Mask r-cnn for object detection and instance segmen-tation on keras and tensorflow.https://github.com/matterport/Mask_RCNN, 2017.

[6] Adrian Boguszewski, Dominik Batorski et al. , LandCover.ai: Dataset for Automatic Mapping of Buildings, Woodlands and Water from Aerial Imagery ; May 2020; arXiv