Segmentation of High Spatial Resolution Satellite Images

Artificial Intelligence field- ENSIAS

Realized by :
Dahmani Zineb
Arbib Nada

Academic year 2021/2022

Abstract

The present report is a synthesis of the work carried out within the framework of the project of Multimedia data processing , carried out within the ENSIAS, a big school of engineers specialized in Technologies of the Information and the Communication. Its missions are the training of state engineers and research for the technological and economic development of Morocco.

The mission we have been given is part of dealing with satellite images that are special in order to do a segmentation and classification of this type of data that have become commercially available and have been increasingly used in various aspects of environmental monitoring and management.

Contents

Introduction

Semantic Segmentation is a fundamental task in computer vision and remote sensing imagery. Many applications, such as urban planning, change detection, and environmental monitoring, require the accurate segmentation; hence, most segmentation tasks are performed by humans. Currently, with the growth of Deep Convolutional Neural Network (DCNN), there are many works aiming to find the best network architecture fitting for this task.

During the few years, various algorithms have been developed to extract features from the high resolution satellite imagery. For classification of these extracted features several complex algorithms have been developed. But these algorithms do not possess critical refining stages of processing the data at the preliminary phase. There are various satellite sensors that have been launched such as LISS3, IKONOS, QUICKBIRD, WORLD VIEW etc. Prior to classification and extraction of semantic data, imagery of the high resolution must be refined. The whole refinement process involves several steps of interaction with the data. These steps are preprocessing algorithms. Preprocessing steps involves Geometric correction, radiometric correction, Noise removal, Image enhancement etc. Due to these preprocessing algorithms the accuracy of the data is increased.Various applications of these preprocessing of the data is in meteorology, hydrology, soil science, forest, physical planning .

1 High resolution satellite images as a special data

The resolution of an image refers to the potential detail provided by the imagery. In remote sensing we refer to three types of resolution: spatial, spectral and temporal.

1.1 Spatial resolution

Spatial Resolution refers to the size of the smallest feature that can be detected by a satellite sensor or displayed in a satellite image. It is usually presented as a single value representing the length of one side of a square. For example, a spatial resolution of 250m means that one pixel represents an area 250 by 250 meters on the ground.
The finest resolution as of now is 30cm provided by very high-resolution commercial satellites.

— Low resolution: over 60m/pixel .
— Medium resolution: 10 – 30m/pixel .
— High to very high resolution: 30cm – 5m/pixel .

To illustrate the difference between various resolutions of satellite image data, look at these two satellite images :

Figure 1 – 10m medium-resolution VS 50cm high-resolution

1.2 Spectral resolution

When talking about spectral data, you need to understand both the electromagnetic spectrum and image bands. Spectral remote sensing data are collected by powerful camera-like instruments known as imaging spectrometers. Imaging spectrometers collect reflected light energy in "bands."

A band represents a segment of the electromagnetic spectrum.

Often when you work with a multispectral dataset, the band information is reported as the center wavelength value. This value represents the center point value of the wavelengths represented in that band. Thus in a band spanning 800-850 nm, the center would be 825 nm.

The spectral resolution of a dataset that has more than one band, refers to the spectral width of each band in the dataset.

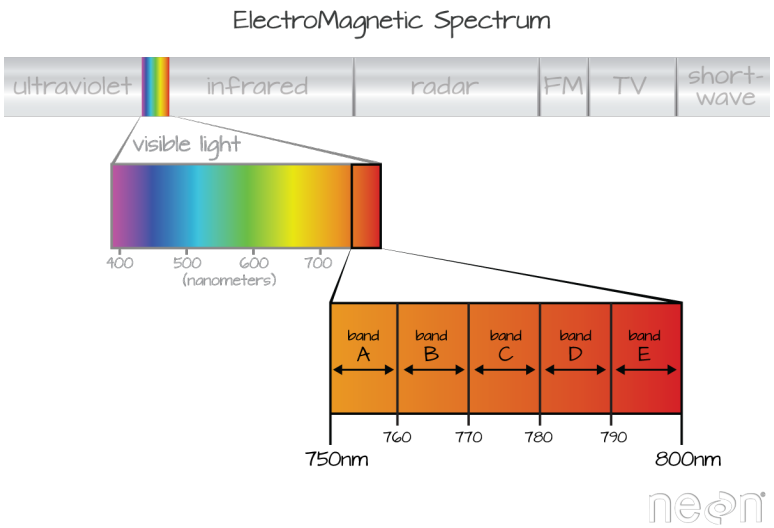


Figure 2 – Imaging spectrometers collect reflected light information within defined bands

1.3 Metadata

Another aspect that characterise satellite images and make them special is their metadata. Metadata means "data for the data". And if satellite images are our data, then we refer to the data that describe these images. What time were they taken? What was the sensor's geometry at that time? Where exactly, in the Earth, does the image refer to? These kinds of questions are answered by the metadata.

1.4 Example of a study

We will explore an image from Planet lab using Rasterio .

Figure 3 – Planet Scope image provided by Planet lab

Dataset attributes :

After opening our image in reading mode , we will first look for the properties of our raster data stored in the example . Dataset objects have bands and this example has a band count of 4 (Blue , Green , Red , Near infared) .

Dataset georeferencing :

A raster dataset is different from an ordinary image; its elements (or "pixels") are mapped to regions on the earth's surface. Every pixels of a dataset is contained within a spatial bounding box :

BoundingBox(left=544491.0, bottom=4178370.0, right=569514.0, top=4191009.0)

Our example covers the world from 544491 meters to 569514 meters, left to right, and 4178370 meters to 4191009 meters bottom to top.

We can have more details and information about our image :

['driver': 'GTiff', 'dtype': 'uint16', 'nodata': 0.0, 'width': 8341, 'height': 4213, 'count': 4, 'crs': CRS.fromepsg(32610), 'transform': Affine(3.0, 0.0, 544491.0, 0.0, -3.0, 4191009.0), 'blockxsize': 256, 'blockysize': 256, 'tiled': True, 'compress': 'lzw', 'interleave': 'pixel']

The coordinate reference system of a dataset is accessed from its crs attribute.That indicates :

- **WGS84 Bounds:** -126.0000, 0.0000, -120.0000, 84.0000 .
- **Projected Bounds:** 166021.4431, 0.0000, 833978.5569, 9329005.1825 .
- **Scope:** Large and medium scale topographic mapping and engineering survey .
- **Last Revised:** June 2, 1995 .
- **Area:** World - N hemisphere - 126°W to 120°W - by country .

Displaying Composite Imagery :

We load the bands into numpy arrays and we scale their values for display purposes .
Plot the RGB stack to see a true-color representation :

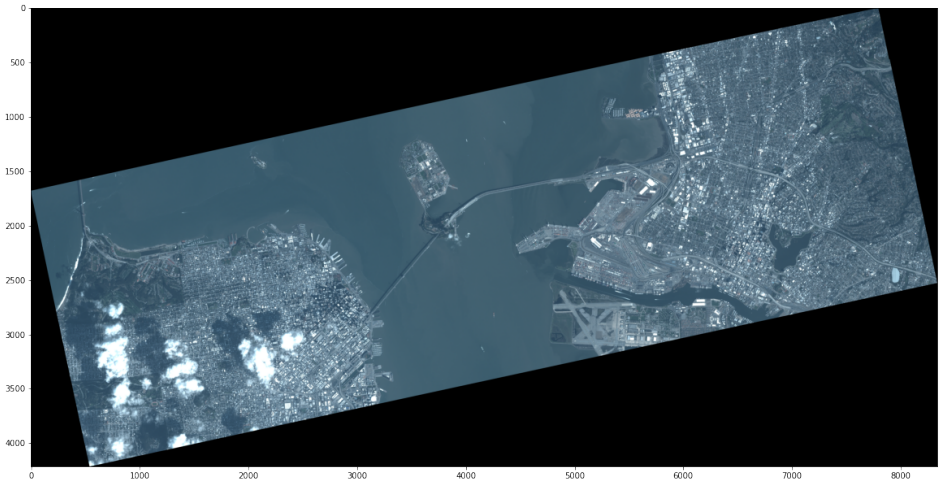


Figure 4 – True-color representation

2 K Means

2.1 Principle of the algorithm

Kmeans algorithm is an iterative algorithm that tries to partition the dataset into **K** pre-defined distinct non-overlapping subgroups (clusters) where each data point belongs to only one group. It tries to make the intra-cluster data points as similar as possible while also keeping the clusters as different (far) as possible. It assigns data points to a cluster such that the sum of the squared distance between the data points and the cluster's centroid (arithmetic mean of all the data points that belong to that cluster) is at the minimum. The less variation we have within clusters, the more homogeneous (similar) the data points are within the same cluster.

The way kmeans algorithm works is as follows:

- Specify number of clusters K .
- Initialize centroids by first shuffling the dataset and then randomly selecting K data points for the centroids without replacement .
- Keep iterating until there is no change to the centroids. i.e assignment of data points to clusters isn't changing.
- Compute the sum of the squared distance between data points and all centroids. Assign each data point to the closest cluster (centroid) .
- Compute the centroids for the clusters by taking the average of the all data points that belong to each cluster.

To get a better results , The k-means clustering algorithm runs several times in a row, changing the initial position of the centroids each time.

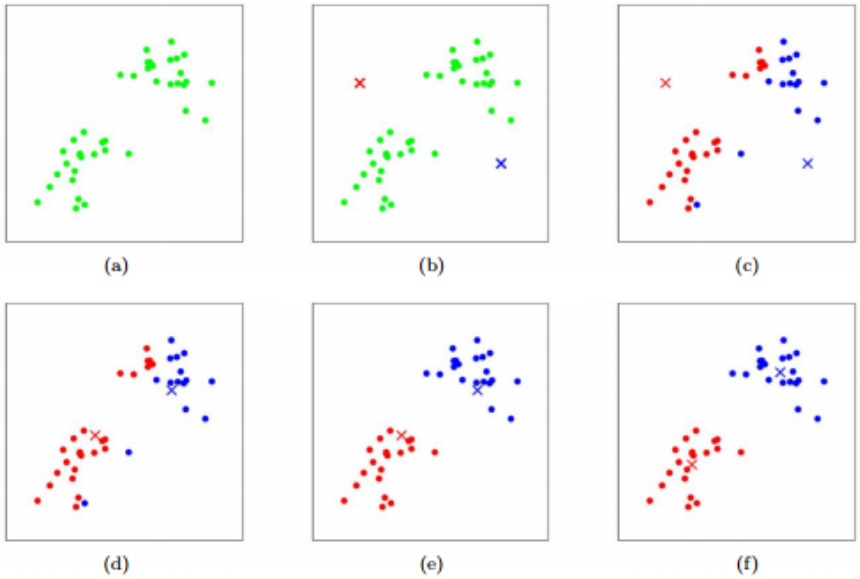


Figure 5 – Explanation of the working method of k means

2.2 Algorithm diagram

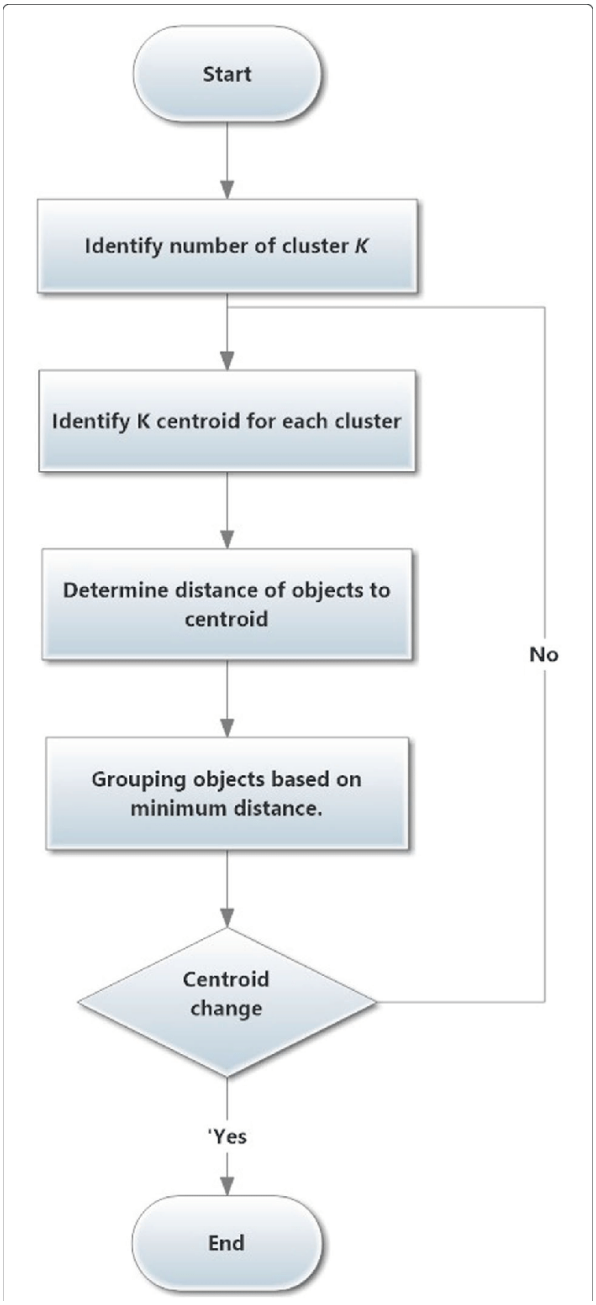


Figure 6 – K Means diagram

To get a better results , The k-means clustering algorithm runs several times in a row, changing the initial position of the centroids each time.

2.3 Application and results

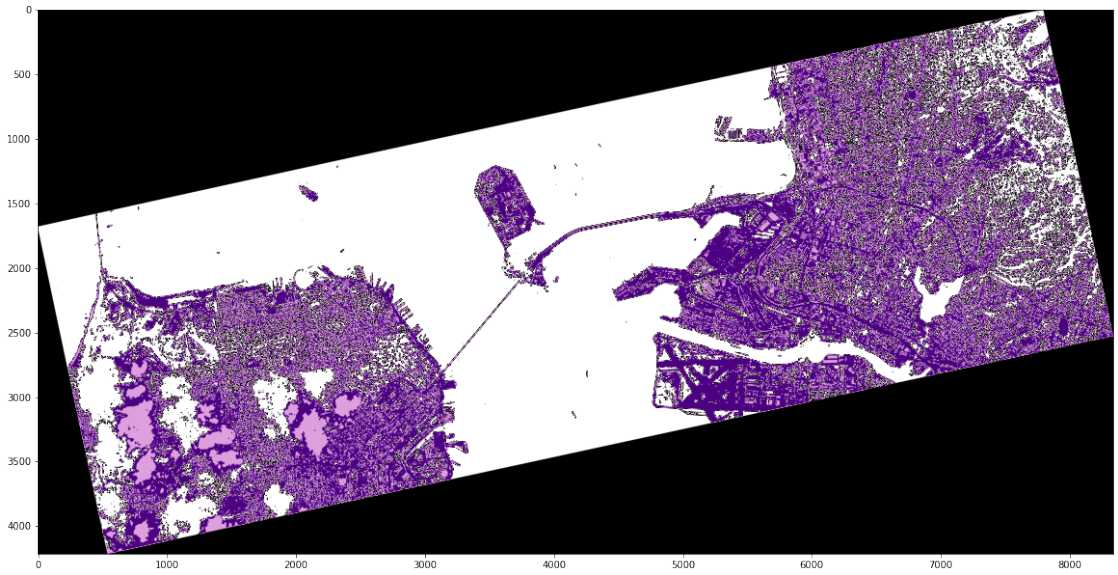


Figure 7 – Results for k means k = 4

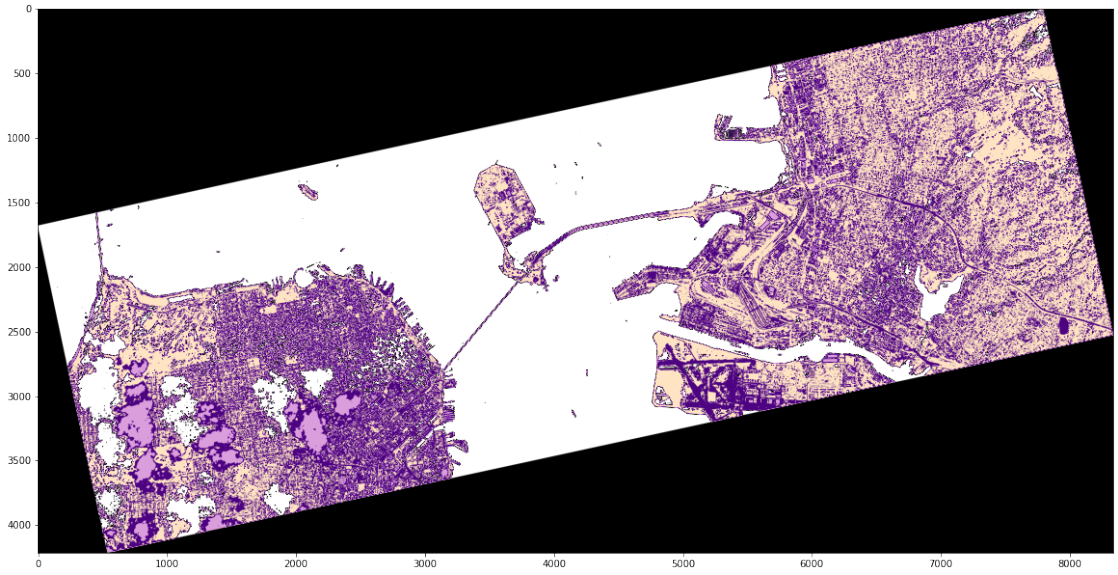
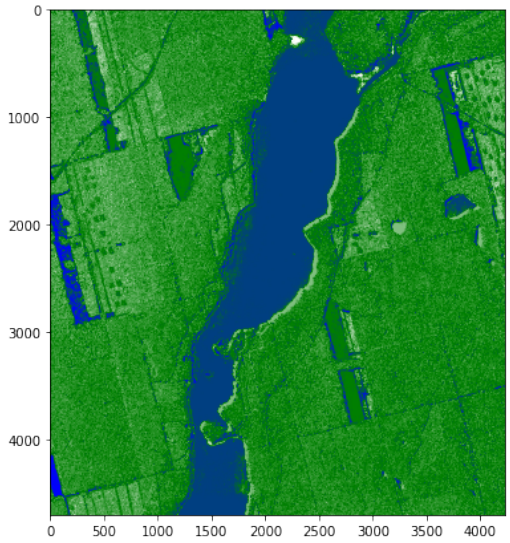
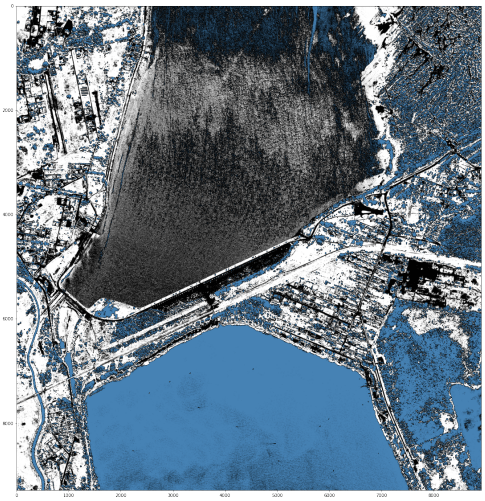
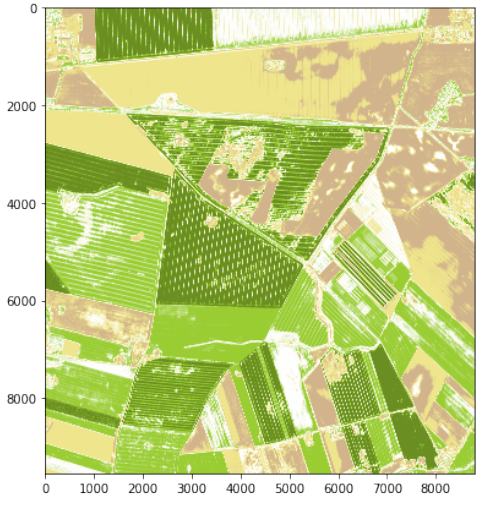


Figure 8 – Results for k means k = 5

Clearly , applying k means for K = 5 provides better results . However, there are still points where the segmentation has failed.For example , the shadow of the clouds has been put in the same cluster with the water (which is normal since this segmentation using K Means is essentially based on a clustering of colors).

Let's see more examples :

	
Original	k = 3
	
Original	k = 3
	
Original	k = 5

2.4 Pros and Cons

Pros

— Simple and easy to understand : We are assigning the points to the clusters which are closest to them .
— Improving the segmentation quality by increasing the number of clusters.

Cons

— K must be determined .
— Segmentation can fail with weather conditions (reflections of the sun (example 3) - clouds (study example)) .
— Is based on a clustering of colors .
— It looks at all the samples at every iteration, so the time taken is too high

3 U-Net

Before we dive into the UNET model, it's very important to understand the different operations that are typically used in a Convolutional Network. Please make a note of the terminologies used.

3.1 Principle of convolutional network

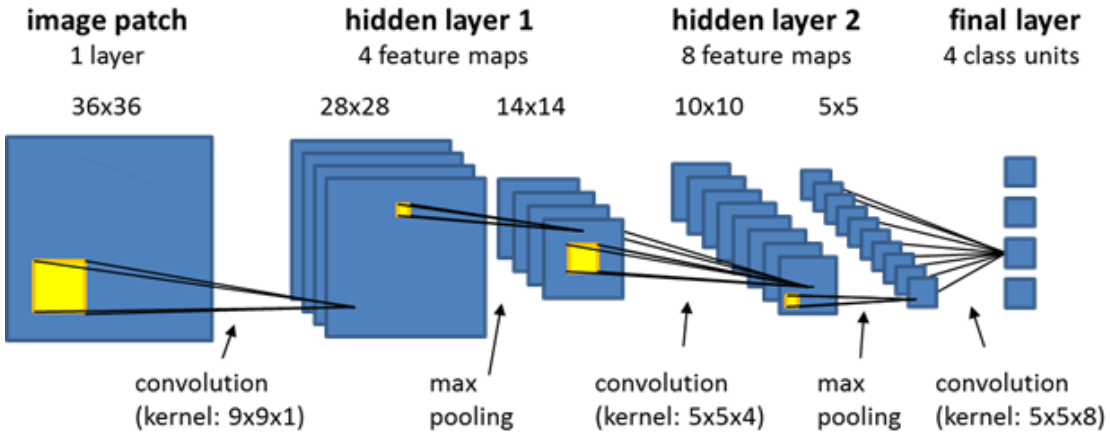


Figure 9 – CNN diagram

3.1.1 Convolution operation

There are two inputs to a convolutional operation :
— A 3D volume (input image) of size (n_{in} x n_{in} x channels) .
— A set of 'k' filters (also called as kernels or feature extractors) each one of size (f x f x channels), where f is typically 3 or 5 .

The output of a convolutional operation is also a 3D volume (also called as output image or feature map) of size (n_{out} x n_{out} x k).

The relationship between nin and nout is as follows:

$$n_{out} = [\frac{n_{in} + 2p - k}{s}] + 1 \tag{1}$$

where :
— n_{out} : number of output features .
— n_{in} : number of input features .
— **k** : convolution kernel size .
— **p** : convolution padding size .
— **s** : : convolution stride size .

3.1.2 ReLU Activation Layer

The convolution maps are passed through a nonlinear activation layer, such as Rectified Linear Unit (ReLu), which replaces negative numbers of the filtered images with zeros.

3.1.3 Pooling Layer

The pooling layers gradually reduce the size of the image, keeping only the most important information. For example, for each group of 4 pixels, the pixel having the maximum value is retained (this is called max pooling), or only the average is retained (average pooling).

Pooling layers help control overfitting by reducing the number of calculations and parameters in the network.

After several iterations of convolution and pooling layers (in some deep convolutional neural network architectures this may happen thousands of times), at the end of the network there is a traditional multi layer perceptron or "fully connected" neural network.

3.1.4 Fully Connected Layer

In many CNN architectures, there are multiple fully connected layers, with activation and pooling layers in between them. Fully connected layers receive an input vector containing the flattened pixels of the image, which have been filtered, corrected and reduced by convolution and pooling layers. The softmax function is applied at the end to the outputs of the fully connected layers, giving the probability of a class the image belongs to – for example, is it a road, a building or an airplane.

3.2 Principle of U-Net algorithm

The UNET was developed by Olaf Ronneberger et al. for Bio Medical Image Segmentation. The architecture contains two paths. First path is the contraction path (also called as the encoder) which is used to capture the context in the image. The encoder is just a traditional stack of convolutional and max pooling layers. The second path is the symmetric expanding path (also called as the decoder) which is used to enable precise localization using transposed convolutions. Thus it is an end-to-end fully convolutional network (FCN), i.e. it only contains Convolutional layers and does not contain any Dense layer because of which it can accept image of any size.

In the original paper, the UNET is described as follows:

3.3 U-Net diagram

Here's the U-Net diagram and its explanation :

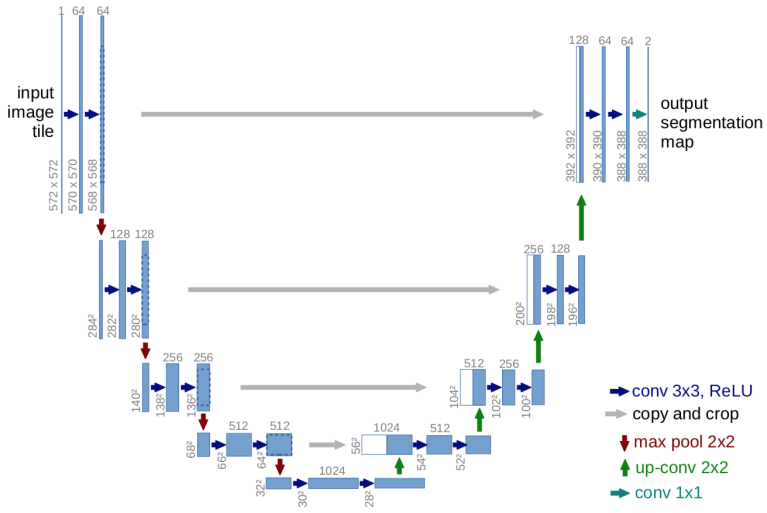


Figure 10 – U-Net diagram

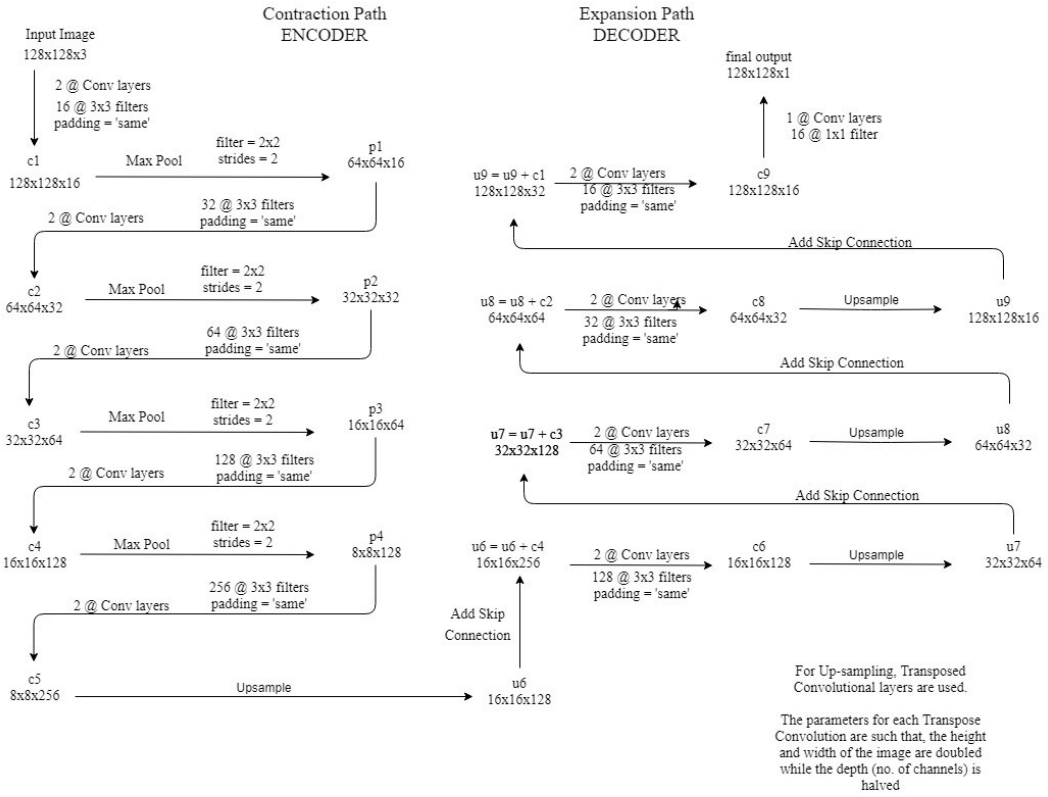


Figure 11 – Explanation of the U-Net diagram

3.4 Application and results

3.4.1 Description of the dataset

The LandCover.ai (Land Cover from Aerial Imagery) dataset is a dataset for automatic mapping of buildings, woodlands, water and roads from aerial images.

Dataset features

— Land cover from Poland, Central Europe 1 .
— Three spectral bands - RGB .
— 33 orthophotos with 25 cm per pixel resolution (9000x9500 px) .
— 8 orthophotos with 50 cm per pixel resolution (4200x4700 px) .
— Total area of 216.27 km2 .

Dataset format

— Rasters are three-channel GeoTiffs with EPSG:2180 spatial reference system .
— Masks are single-channel GeoTiffs with EPSG:2180 spatial reference system .

Dataset classes

— Classes: building (1), woodland (2), water(3), road(4) .
— Areas: 1.85 km2 of buildings, 72.02 km2 of woodlands, 13.15 km2 of water, 3.5 km2 of roads .

3.4.2 Data preprocessing

We have a dataset of a hundreds of large images that we break down into tens of thousands of of smaller images and then need to read them directly from the desk, train a model and segment larger images.
Starting with larger images , we need to find a way we can organize them so we can use U-Net model and ImageDataGenerator from Keras. To read these images directly from the disk train a model, save the model and apply to large images patch by patch.

Data preparation workflow:

1. Read large images and corresponding masks, divide them into smaller patches of size 256x256. And write the patches as images to the local drive.

2. Save only images and masks where masks have some decent amount of labels other than 0. Using blank images with label=0 is a waste of time and may bias the model towards unlabeled pixels.

3. Divide the sorted dataset from above into train and validation datasets.

4. manually move some folders and rename them appropriately to use ImageDataGenerator from keras.

- Quick understanding of the dataset

We read an image using OpenCV so we notice that the is of size (9582,9170,3). So have 3 channels per spectral bands.In the image below we View each channel.

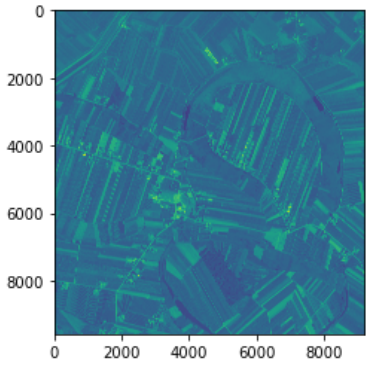


Figure 12 – Chanel 0

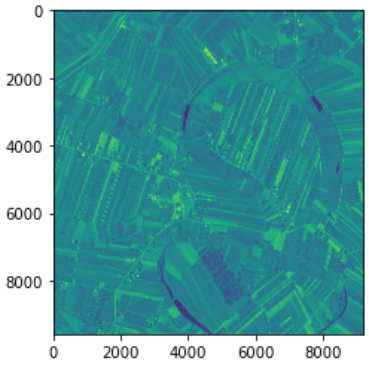


Figure 13 – Chanel 1

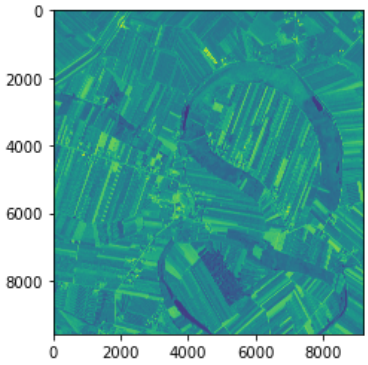


Figure 14 – Chanel 1

- Cropping images :

- Read images from repsective 'images' subdirectory.
- As all images are of different size we have 2 options, either resize or crop but, some images are too large and some small. Resizing will change the size of real objects.therefore, we will crop them to a nearest size divisible by 256 and then divide all images into patches of 256x256x3.

In the picture below we visualize some images with their masks.

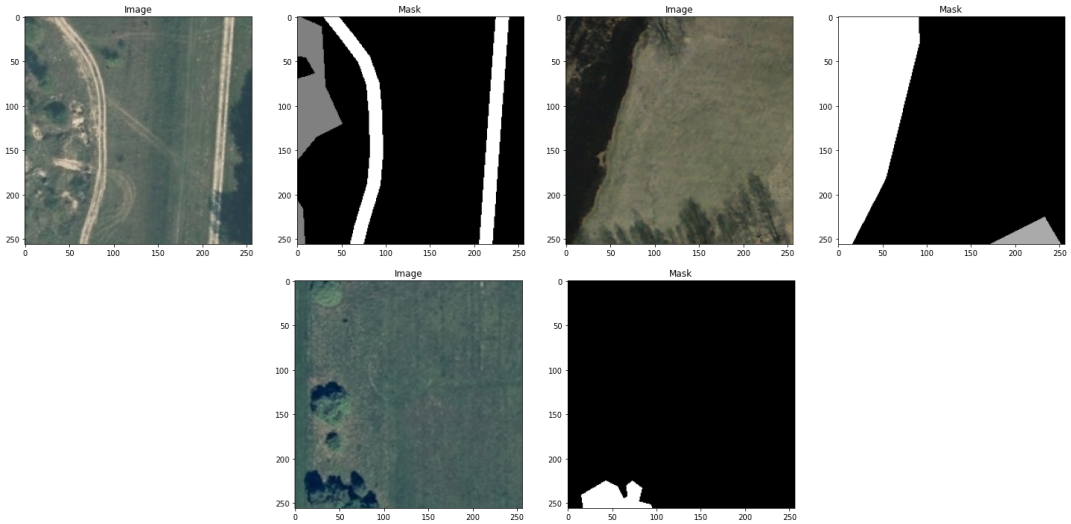


Figure 15 – Images with Masks

- Copy images and masks with real information to a new folder. Information is real if mask has decent amount of labels other than 0.

- Split the data into training, validation and testing using split-folders.

- Manually move folders around to bring them to the following structure.

```
"""

Data/
    train_images/
                train/
                    img1, img2, img3, ......

    train_masks/
                train/
                    msk1, msk, msk3, ......

    val_images/
                val/
                    img1, img2, img3, ......
    val_masks/
                val/
                    msk1, msk, msk3, ......


"""
```

Figure 16 – Files structure

3.4.3 Results

We choose to use of pretrained encoder in the U-net - available as part of segmentation models library.
- Encoder "resnet34" with pretrained weights of ImageNet, that speed up our training.
- Preprocess the data using get_processing of of segmentation_models package.
- Generate data using the ImageDataGenerator of keras appliyin horizontal flip , vertical flip and scale, preprocess and convert masks to categorical.

Training the model

The model converged very well and the intersection union of all classes combined was getting 67 to about 79% after training for 25 epochs which is not bad.

```
Mean IoU = 0.67488253  Mean IoU = 0.7908055
```

Figure 17 – mean IoU

Figure 18 – visualizing Loss and IoU

Testing generator using validation data

Here same examples of what we get after training :

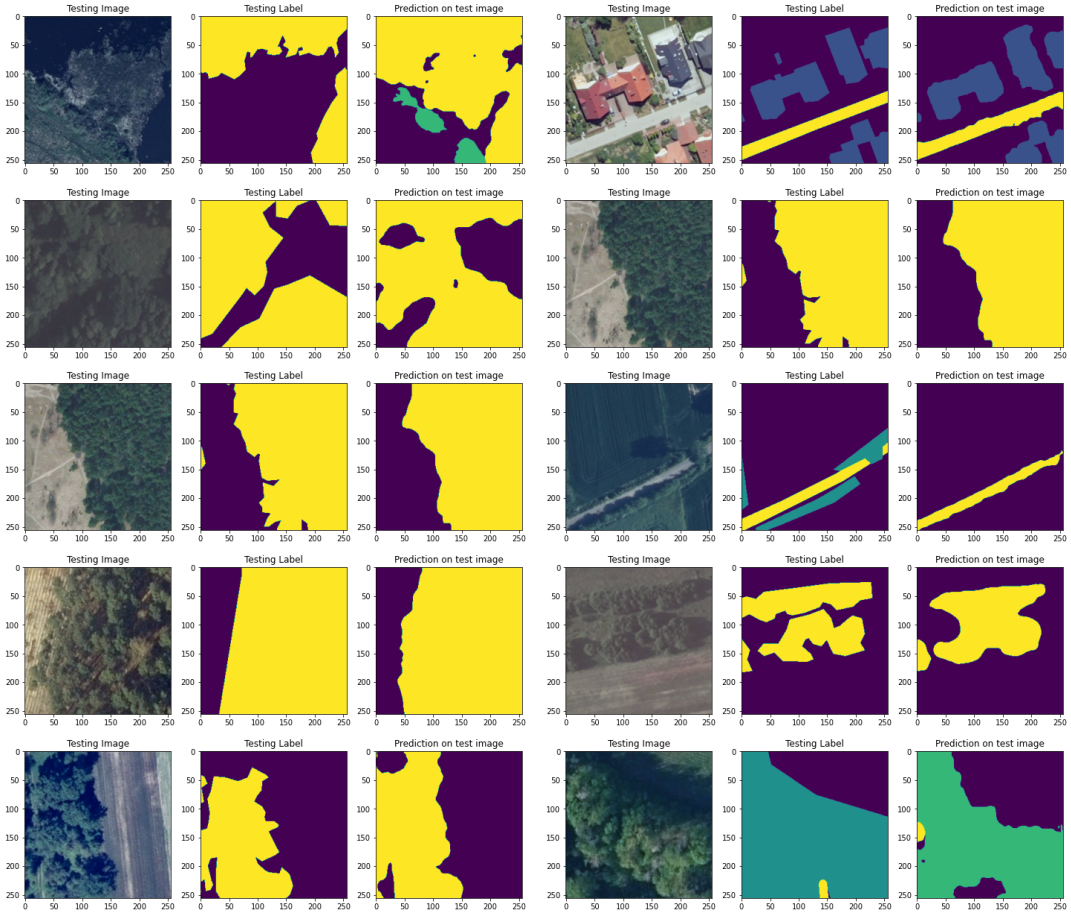


Figure 19 – visualizing images, masks and corresponding predictions

3.5 Pros and Cons

Pros

— In this architecture, the network is input image size agnostic since it does not contain fully connected layers.This also leads to smaller model weight size.
— Can be easily scaled to have multiple classes.
— Relatively easy to understand why the architecture works, if we have basic understanding of how convolutions work.
— Architecture works well with small training set, thanks due robustness provided with data augmentation.

Cons

— The Architecture initially was designed to overcome the ad hoc choice of 'rolling window' / patches hyper-parameters. To have decent results the size of U-NET that you will be using must be comparable with the size of features and its surroundings.
— Because of many layers takes significant amount of time to train.
— Relatively high GPU memory footprint for larger images.

Conclusion

This project was very enriching for us because it allowed us to discover and make a first step towards the field of computer vision, its actors, its constraints but also to participate concretely in its stakes through the missions which were given to us.

Through this project we were able to meet the requested tasks (Segmentation of high resolution images). This was accomplished by realizing an approach focused on two algorithms K Means and U Net.

Higher-resolution imagery more accurately delineated cover classes, identified smaller patches, retained patch shape, and detected narrower, linear patches.But we have faced problems dealing with this type of images ,since this type of images is really big , the trainning time of our models was way too big .