UNIVERSIDAD NACIONAL DE COLOMBIA

Automatic Generation of GIS Vector Layers from Orthomosaics using Deep Learning

John Robert Ballesteros Parra

Universidad Nacional de Colombia – Sede Medellín Facultad de Minas Programa de Doctorado en Ingeniería de Sistemas e Informática
Medellin, Colombia
2022

Automatic Generation of GIS Vector Layers from Orthomosaics using Deep Learning

John Robert Ballesteros Parra

A dissertation submitted in partial fulfillment of the requirements for the degree of:
PhD. Doctor en Ingeniería - Ingeniería de Sistemas e Informática

Advisor:
German Sanchez Torres, Ph.D.
Universidad del Magdalena

Co-Advisor:
John Willian Branch Bedoya, Ph.D.
Universidad Nacional de Colombia

Universidad Nacional de Colombia – Sede Medellín Facultad de Minas Programa de Doctorado en Ingeniería de Sistemas e Informática
Medellin, Colombia
2022

Abstract

Keywords: GIS, Vectorization, GAN, Semantic Segmentation, Orthomosaics, Deep Learning, Image Translation, Image Caption.

This thesis presents a three methods pipeline for extraction of point, line, and polygon vector objects from orthomosaics using a deep generative model as an alternative to the default semantic segmentation approach. The first method consists of two workflows, the vector ground truth is acquired by manual digitalization of certain objects or from Open Street Maps. Raster layers input are spectral and geometrically augmented, both inputs are then tessellated and paired into image-masks that pass through an imbalance checking step. Balanced dataset is then random split into a final dataset. Conditional and unpaired generative models are compared and pix2pix is chosen by its better results on image to mask translation. Results of the chosen model on different datasets and configurations are reported on the mIoU metric. A batch size of 10 and datasets of 1000 image-masks pairs of 512x512 pixels, with overlapping augmentation showed the best quantitative results. Height of objects from the DSM, and VARI index contribute to decrease variance of discriminator and generator losses. Producing synthetic data is the horsepower of generative models, so a double image to mask translation is used to improve resultant masks in terms of continuity and uniform width. Double image to mask translation model is trained with a dataset of equal size masks of 1 meter called primitive masks, that are obtained by a buffer distance parameter. This cleaning procedure showed to improve resultant masks, that are then converted to vector and measured by quantity, length, or area against vector ground truth, using a proposed metric for map creation called “The average geometry similarity (AGS)”.

Generación Automatica de Capas Vectoriales SIG de Ortomosaicos usando Aprendizaje Profundo

Resumen

Palabras clave: SIG, Vectorización, Redes Antagónicas, Segmentación Semántica, Ortomosaicos, Aprendizaje Profundo, Traducción de Imagen.

Esta tesis presenta una metodología basada en tres métodos para la extracción de puntos, lineas, y polígonos de objetos vectoriales presentes en ortomosaicos usando un modelo generativo basado en aprendizaje profundo como una alternativa al enfoque de segmentación semántica usado por defecto. El primer método consiste en dos líneas de trabajo, las capas vector de entrenamiento son adquiridas bien sea por digitalización manual de los objetos de interés o directamente desde Open Street Maps (OSM). Las capas raster de entrada son aumentadas spectral y geométricamente, teseladas y emparejadas en pares imagen-mascara que se chequean ante el imbalance. El conjunto de datos balanceado es luego partido al azar para obtener el conjunto final. Los modelos generativos, condicionales y no emparejados son comparados y el mejor es escogido para realizar las traducciones entre imagen y mascara. Los resultados de la comparación y los obtenidos por el mejor modelo sobre diferentes conjuntos de datos, y su configuración son reportados usando la metrica mIoU. Un lote de tamaño diez para un conjunto de 1000 image-mascaras de 512x512 pixeles, con augmentación por solapamiento mostró los mejores resultados cuantitativos. La altura de los objetos obtenida del DSM, y el índice VARI contribuyen a disminuir la varianza del discriminador y del generador. La producción de datos sintéticos es el caballo de batalla de los modelos generativos, así que una doble traducción de imagen a mascara (DCIT) es empleada para mejorar las mascaras resultantes en términos de su continuidad y uniformidad. Un modelo para realizar DCIT es entrenado con un conjunto de datos de igual tamaño de mascara de 1 metro llamado mascaras primitivas, que son obtenidas usando una distancia buffer como parametro. Este procedimiento de limpieza mostró que mejora las mascaras resultantes, que son luego convertidas a vector y medidas en cantidad, distancia, o area vs la realidad vectorial, usando una métrica propuesta para la creación de mapas llamada “Similaridad geomética promedia (AGS)”.

Acknowledgements

This thesis would not have been possible without the love, patience, and support of my beloved family who have accompanied me during these years.

I would like to thank my advisor and co-advisor, Professors German Sanchez Torres and John Willian Branch Bedoya, for guiding me in this process. Their advice, support, enthusiasm, friendship, and patience have been of great help in the research pathway. Their recommendations, discussions, and interest have shaped this work.

Special thanks go to the whole Universidad Nacional de Colombia, it has been my "Alma Mater"

To my mother who always dreamed to give me higher education, but also to my father who was a suffice support in the time he could stay by my side.

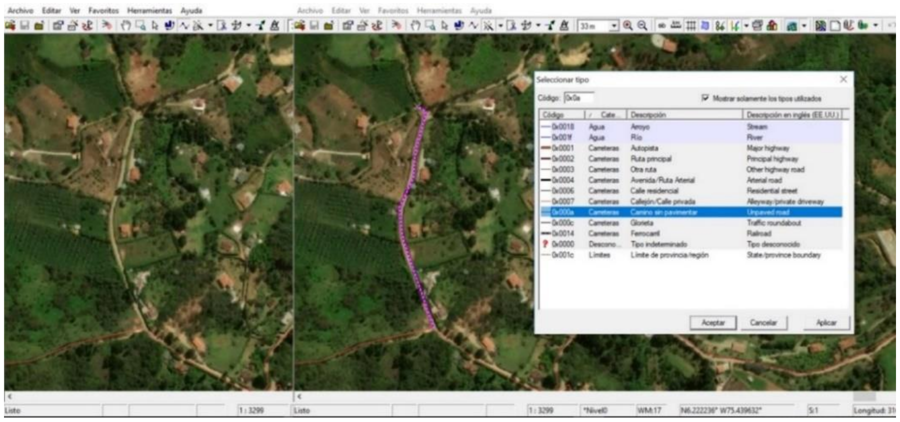
Contents

Table of Contents

Chapter 1

1 Introduction

From ancient times, people have tried to represent physical objects of the landscape, they include natural and man-made of any scale, complexity, and character. The remote sensing and GIS mapping communities call them geographical objects or features, vehicles, roads, and building are examples of geographical objects. Even today, maps are obtained by a manual restitution of objects from drone, aerial, or satellite orthomosaics (Van Etten et al., 2019). The process includes recognition, visual classification, digitalization, and attribution of objects that are organized in different layers. This manual method is cumbersome and prone to errors; besides it takes large efforts in terms of time, and expert people. Furthermore, landscape changes rapidly due to earth natural processes and human interaction, which demands constant updating and curation of maps. Figure 1.1 illustrates afore mentioned steps applied to create a road layer from a satellite orthomosaic. An orthomosaic is an ortho-rectified and geo-referenced image of the earth's surface. It is created by stitching partially overlapped images in a software using a method called Structure from Motion (SfM) (Kameyama & Sugiura, 2021). The rapid development, and low cost of satellite imagery, but specially, the ease of acquisition and high resolution of drone aerial imagery (Murtiyoso et al., 2020), have made orthomosaics a useful tool in engineering, architecture, geology, and many other fields (Avola & Pannone, 2021; Ballesteros et al., 2021; Bhatnagar et al., 2020; Zhang et al., 2015).



Código: 0x0a

[✓] Mostrar solamente los tipos utilizados

Código	Cate...	Descripción	Descripción en inglés (EE.UU.)
0x0018	Agua	Arroyo	Stream
0x001f	Agua	Río	River
0x0001	Carreteras	Autopista	Major highway
0x0002	Carreteras	Ruta principal	Principal highway
0x0003	Carreteras	Otra ruta	Other highway road
0x0004	Carreteras	Avenida/Ruta Arterial	Arterial road
0x0006	Carreteras	Calle residencial	Residential street
0x0007	Carreteras	Callejón/Calle privada	Alleyway/private driveway
0x000a	Carreteras	Camino sin pavimentar	Unpaved road
0x000c	Carreteras	Glorieta	Traffic roundabout
0x0014	Carreteras	Ferrocarril	Railroad
0x0000	Descono...	Tipo indeterminado	Tipo desconocido
0x001c	Límites	Límite de provincia/región	State/province boundary

Figure 1.1. Manual creation of a vector layer. The manual creation of road network for a small town may take days to months.

GIS have two spatial data types to represent geographic data, raster and vector (Bolstad, 2016). A raster layer is a grid of regularly sized pixels. Each pixel is classified as an object or part of an object like tree, road, building, etc. The spatial resolution, also called the grid sampling distance (GSD), depends on the pixel size, the smaller the pixel the higher the spatial resolution. Raster is likely to represent continuous objects such as element concentration or surfaces like a digital terrain model (DTM). Other examples of raster data would be aerial photographs, and scanned maps. A vector layer, or vector map, consists of objects defined by coordinates in a given spatial reference system (Bolstad, 2016). They are of three types:

- Point: a point object in two dimensions is a pair of coordinates (x_i, y_i) that represent separate non-adjacent features. Examples are: volcanoes, offices, schools, and shopping malls.

- Line: a line is a set of coordinates of its vertices, used to represent linear features characterized by having starting and ending points. Some examples of linear objects are roads, rivers, and metro lines.

- Polygon: It is represented by a set of coordinates in which the first and the last pair of coordinates are the same according to a certain level of tolerance. A polygon is two dimensional and can be used to measure the area of the desired geographic object. Lakes, city boundaries, or forests are examples of polygons.

Vector is appropriate to represent discrete objects, for example a vehicle or a pipe. Vector objects have metadata that describes the characteristics — the name of a road, or the population of a region. These extra, non-spatial metadata of an object are called “attributes”, and are stored in an “attribute table” (Bolstad, 2016). Vector layers are commonly obtained by manual digitalization on the screen of a computer, after that, objects are also manually attributed one by one. Figure 1.2. illustrates raster and vector GIS computer graphic representations.

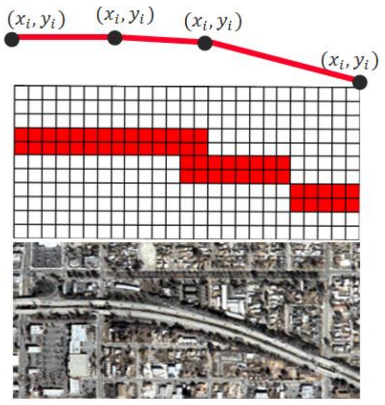


Figure 1.2. Raster and vector object representation.

GIS have different file formats for both vector and raster data. The most common are:

- Geographic JavaScript Object Notation (GeoJSON: .geojson) is the most used format for web-based mapping. It stores the coordinates as text in JSON form, which includes the vector points, lines, polygons as well as tabular information within curly braces “{}” (Butler et al., 2016).

- ESRI Shapefile (.shp) is widely accepted by all commercial and opensource organizations. It has become the industry standard (*The Home of Location Technology Innovation and Collaboration | OGC*, www.ogc.org).

- GeoTIFF (.Tif, .Tiff) stores raster data. It has become an industry standard for satellite and remote sensing imagery (Mahammad & Ramakrishnan, 2003).

Deep learning models have improved the performance of the already rapidly developing field of computer vision, which powers emerging technologies like facial recognition, augmented reality, and self-driving cars (Pashaei et al., 2020). To keep pace with the high speed of production of remote sensing imagery, and at the same time, increase the accuracy of mapping, deep neural network models have been lately included into the mapping workflow (Osco et al., 2021). Deep Learning based Semantic Segmentation, also called pixel classification, is the process of assigning a class to each pixel in the image, distinguishing for instance, roads from buildings, and vegetation (Xu et al., 2018). Semantic Segmentation has been extensively studied in ground imagery, and it is the state-of-the-art method to perform GIS layers extraction from orthomosaics (Ng & Hofmann, 2018). To accomplish this, semantic segmentation models input image tiles from an orthomosaic paired with corresponding masks called the ground truth. Masks are binary or color images that represent objects of interest in the neural networks. These masks are frequently obtained and annotated manually.

The goal of this thesis is to propose and study an alternative method to semantic segmentation, based on a generative model and data, to generate “cleaner” and less discontinue masks to automatically obtain vector layers and attributes. The rest of the thesis is organized as follows:

Chapter 2: presents a brief overview of existing work in geographic objects extraction from orthomosaics. Starting with GIS computer graphic representation of objects, it describes manual and semi-automatic methods, image semantic segmentation architectures and deep generative methods.

Chapter 3: is an introduction to our proposed methodology for GIS vector layer extraction from orthomosaics. It illustrates the three methods proposed: paired data, generative model, and post-processing.

Chapter 4: addresses the paired data workflow in detail. It describes the different steps to produce better data to improve the generative model for point, line, and polygon objects of interest, and presents examples of the created datasets.

Chapter 5: explores the results and findings of applying the proposed generative model, the image to mask translation model, to produce better quality masks. It also describes model hyperparameters and experiments in training.

Chapter 6: describes how proposed model can improve mask generation prior to vectorization using the concepts of primitive masks and double image to mask translation. This chapter also tackles the automatic attribution of objects via color encoding-decoding.

Chapter 7: studies the proposed image translation model performance and results.

Chapter 8: summarizes our most important findings and offers a discussion of the most promising directions for improving our system.

1.1 Problem Description

In GIS computer graphics, geographical objects are represented as point, line, or polygon based on scale and characteristics. Mapping communities, open source or proprietary, are particularly interested in automatically extracting them from aerial, and satellite imagery, and very recently from drone orthomosaics. Geographical objects extraction from aerial imagery refers to the process of transition of the data representation between a set of pixels (raster), and a set of georeferenced coordinates (vector) called a vector layer or layer (Girard & Tarabalka, 2018; Li et al., 2021).

With the advent of artificial intelligence, deep semantic segmentation and vectorization has become the default method to automatically extract vector layers from orthomosaics. Deep Learning Segmentation models like U-Net (Ronneberger et al., 2015), Pyramid Scene Parsing Network (PSPNet) (H. Zhao et al., 2017), DeepLab (Cheng et al., 2020), and FCN (J. Long et al., 2015), and their improved versions, have been adapted for the aerial, drone and satellite imagery domain to address automatic mapping of objects in images (Abdollahi et al., 2021; Pashaei et al., 2020). The resultant segmentation masks, inferred by the model, are then cleaned by a set of different heuristic methods and vectorized into layers. (Ng & Hofmann, 2018) have proposed a pipeline to obtain vector layers from aerial images using different semantic segmentation models. Figure 1.3 shows the methods of the mentioned pipeline.

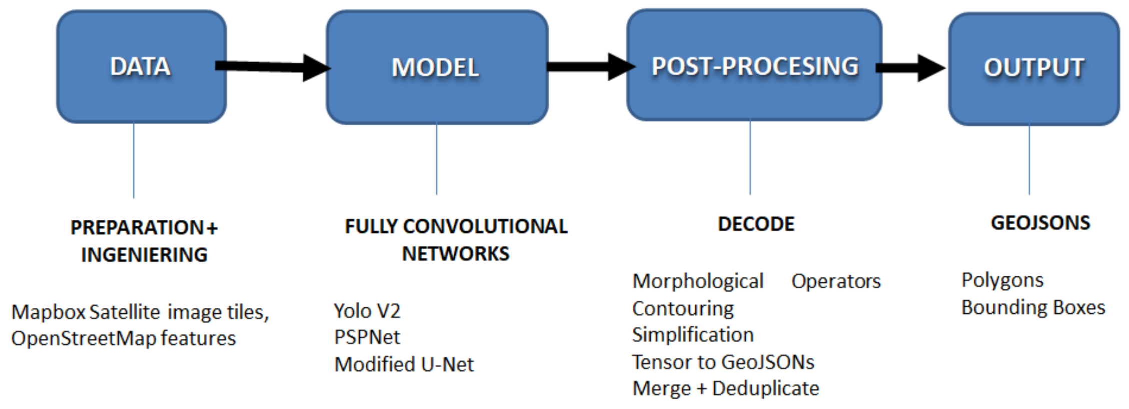


Figure 1.3. Robosat: A Computer Vision Pipeline for geographic objects extraction. Modified from (Ng & Hofmann, 2018).

However, geographical objects impose specific challenges to semantic segmentation, such as:

- They exhibit high interclass object variance of reflectance, texture, and shape. For example, building roofs have similar color to paved roads (Zhang et al., 2015).
- Different geographical object classes have a large overlapping in reflectance, texture, and shape. For instance, an unpaved road may look as a river.
- Geographic objects are commonly occluded by other objects, clouds, and shadows. For example, a vehicle can be occluded by a building roof or by trees (Zhang et al., 2015).
- Orthomosaics are geo-located and have a larger number of pixels compared to ground imagery. They have millions of pixels while ground imagery have thousands (Avola & Pannone, 2021; Osco et al., 2021).

Furthermore, the use of semantic segmentation to geographical objects still has the following issues:

- Creation of high number of false positives, and ineffective representation of straight lines and square corners, both common in man-made objects like roads, and buildings (Li et al., 2021).
- It needs a huge number of training examples, and thus it is heavily affected by imbalance pixel-classes, for example, between background and roads, which is frequently observed in datasets (Gao et al., 2018).

- Limited generalization, a segmentation model created for certain objects in the developed countries, does not perform the same in other latitudes (Maggiori et al., 2017).

- High specificity of semantic segmentation makes different objects and geometry require models' modifications (Y. Long et al., 2021; Marmanis et al., n.d.).

In summary, semantic segmentation may produce irregular and discontinuous masks that lead to an erroneous vector representation transition. This happens even using high quality training masks, and a multi-procedure cleaning postprocessing. Furthermore, creating a loss function that treats object geolocation as a direct objective of optimization is not feasible due to the unknown number of key points in input images, and the difficulty to handle irregular length outputs using Convolutional Neural Networks (CNNs) (Li et al., 2021). Figure 1.4. shows an example of a training and resultant segmentation road mask.

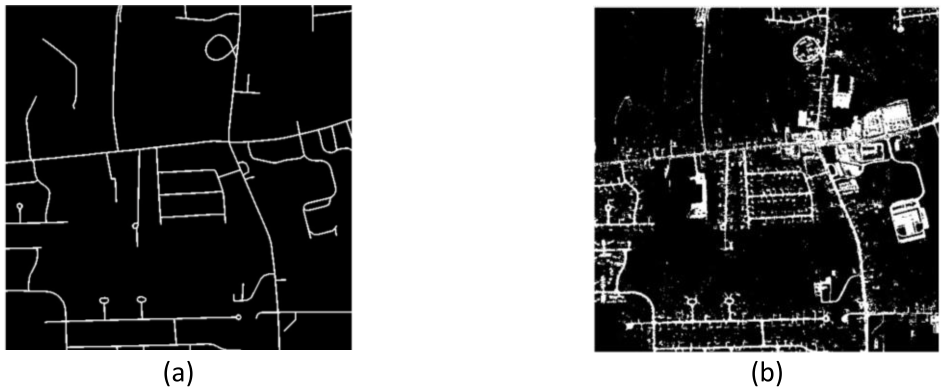


Figure 1.4. Segmentation mask example. (a) Training mask for road extraction, (b) Resultant mask by the U-Net semantic segmentation model. Modified from (Marmanis et al.,2017).

Due to the mentioned issues, our approach is to test a deep generative model instead of semantic segmentation, to generate cleaner and simpler masks of point, line, and polygon objects from input images, and create training data to simplify the vectorization process into layers.

A formal definition of the problem is as follows: Given an orthomosaic X and the simplest raster representation Y of an object class in X, a deep generative based methodology for GIS vector layers extraction consists of finding a function: $g\left(x_i^p, \theta, w_1^k\right)$, where θ are the hyperparameters, and w the weights of a CNN. That is optimized supervisely by pairs of $(x_i^p,\ y_i^p)$ ϵ $X\ and\ Y$ using a loss function $L(y,g) = \sum_i^p min_\theta E\ (y_i^p, g(x_i^p))$ that allows to synthetically creates a class $C_i\ in\ Y\ \forall\ pixel_i\ in\ X$. In other words, the generation of groups of pixels (objects classes) of the input image into a resultant mask of a buffer distance d conditioned by group of examples $(x_i^p,\ y_i^p)$. Furthermore, the masks $\tilde{y}\ \subseteq\ Y$ inferred by the model g can be then cleaned by another function $h(\tilde{y}\,_i^p)$ to produce z_i^p in a way that $(E\left(z_i^p, \tilde{y}\,_i^p\right) - E\left(g\left(x_i^p\right), y_i^p\right)) < 0$, and vectorize those into layers such as: $vect\left(z_i^p\right) = [a_{11}, \dots, a_{mn}]^T$.

1.2 Research Questions

This thesis addresses the following research questions:

- What are the steps and procedure to generate GIS vector layers from orthomosaics using a deep generative model?

- Which type of training masks: multiscale, overlapping, full size, height-augmented or index-augmented, help a deep generative model to obtain the best results in terms of less irregular and continuous masks.

- What are the algorithms and their application order for post-processing resultant masks in order to obtain GIS vector layers comparable to the ones created manually?

1.3 Objectives

Current thesis proposes the following objectives:

1.3.1 General

To develop a methodology to obtain GIS vector layers from orthomosaics using a deep generative model.

1.3.2 Specific

1. To develop a method for creating paired datasets for point, line, and polygon objects.
2. To choose a generative model to produce masks for point, line, and polygon geographical objects present in orthomosaics.
3. To define a post-processing method to clean up the masks of the model output, and the geometry of the corresponding vector objects.
4. To validate the proposed methodology using datasets reported in the scientific literature.

1.4 Contribution and Academic Products

This thesis presents contributions to the extraction of point, line, and polygon geographical objects from orthomosaics. These goals have been achieved by using an Image to Mask Translation Model instead of the traditional Deep Learning Semantic Segmentation approach.

- We developed a method to create paired data that allows to train a Deep Generative Image to Mask Translation Model.

- We pre-trained a model with satellite imagery of the domain, and augment data in a novel way using height of objects and indexes that combined or fusion bands, this showed to improve and speed up the Image to Mask Translation Process.

- We proposed the use of Primitive Masks as a target domain for the Image to Mask Translation Model.

- Proposed Primitive Masks may affect the dataset imbalance, so we implemented a double Image to Mask Translation model that not only used less imbalance data, but also enhanced the geometry of translation masks. This geometry is critical to the vectorization of point, lines, and polygon objects which was performed afterwards using simplification.

Part of this thesis is presented in the following publications:

- Ballesteros, John R.; Sanchez-Torres, German; Branch-Bedoya, John W. HAGDAVS: Height-Augmented Geo-located Dataset for Detection and Semantic Segmentation of Vehicles in Drone Aerial Orthomosaics. Data, April 14, 2022, MDPI.

- Ballesteros John R., Branch John W, Sánchez Torres Germán. Automatic road extraction in small urban areas of developing countries. The IEEE SCLA International Conference 2021. Medellín, Colombia, September 20-23, 2021.

- Ballesteros John, Sánchez Torres Germán, Branch John W. Modelo de generación automática de capas SIG a partir de aprendizaje profundo. Conferencia en Congreso Colombiano de Geología. Medellín, Agosto 2021.

- Ballesteros John, Branch John W. Generación automática de mapas usando IA: Conferencia presentada en el ESRI-SGC GISDay. Abril 2021.

- Ballesteros John R., Detección y Segmentación Automática de Infraestructura Urbana en Imágenes de Drone y Satélite usando Inteligencia Artificial. Ponencia en el X Congreso Internacional de Ingeniería Civil, CONCIVIL, julio 28 al 31, Cartagena, Colombia.

- Ballesteros John R., Sánchez-Torres Germán, Branch John W., Segmentación Semántica de Ríos y Erosión Mediante GANs. XV Semana de la geología, Barranquilla, Col., Agosto 2022, Abstract Accepted.

- Ballesteros, J.; Sanchez-Torres, G.; Branch, J., Road Semantic Segmentation by Fusion-augmented Drone Orthomosaics using a Conditional GAN. In progress. Drone, March 2022, MDPI.

Chapter 2

2 An Overview of GIS Vector Layer Extraction from Orthomosaics

Chapter makes a description of the state-of-the-art methods to automatically extract layers from orthomosaics. After that, it studies previous manual and semi-automatic methods, and details Deep Learning architectures, which are the present workforce, for the automatic extraction of vector layers.

2.1 State of the Art

Foundational mapping remains a challenge in the developing countries, especially in changing scenarios such as natural disasters when time is critical. Creating and updating maps is currently a highly manual process requiring a large number of human experts to either create features or validate automated outputs (Shermeyer & Van Etten, 2019). Next is a compendium of related work.

(Ng & Hofmann, 2018) proposed Robosat, a 4-step pipeline to obtain vectors from aerial and satellite imagery, they describe data preparation as acquisition plus annotation, model refers to the application of detection and semantic segmentation algorithms like Yolo V2, U-Net and PSPNet. Post-processing is composed by a set of methods applied to resultant masks and then to resultant vectors like morphological operators, contouring, simplification and deduplication. For different objects and results segmentation and post-processing was different and results were varied. (Crommelinck et al., 2016, 2017) introduces a workflow for automated cadastral boundary delineation from UAV data. This is done by reviewing and synthesizing approaches for feature extraction from various application fields. It consists of preprocessing, image segmentation, line extraction, contour generation and postprocessing. (Sahu & Ohri, 2019a) used semantic segmentation model based on U-Net to extract individual buildings in densely compacted areas using medium resolution satellite/UAV orthoimages and measure performance on the dice coefficient. However, the chosen metric lacks a geometric aspect involved in the problem. Then a post-processing pipeline for separating connected buildings and converting them into GIS layer for further analysis was performed manually. The vectorization (polygonization) was carried out by the simple and effective Douglas-Peucker method. However, it involved choosing of different threshold and hand engineering. (Bulatov et al., 2016) went further to vectorize networks, and modified Douglas-Peucker algorithm for street generalization after application of preprocessing, thinning, polygonization, and filtering. (Deigele and Brandmeier, 2020) tested different CNNs in conjunction with an integration and vectorization in ArcGIS to evaluate forest damage due to storms on high-resolution PlanetScope satellite data. A custom implementation of U-Net proved to be more accurate than transfer learning that completely failed (IoU=0.55). Forests and building polygon objects exhibited less imbalance classes. (Zhang et al., 2015) Combined UAV Ultra-High resolution Orthophotos with DSM to create land cover classification maps, their experiments demonstrate that the DSM information has greatly enhanced the classification accuracy from 63.93% with only spectral information to 94.48%. Also, (Al-Najjar et al., 2019) fusion DSM and UAV images for land use/land cover mapping using CNNs. Results confirmed that fusion data performed better than only RGB images, with an overall pixel accuracy of 0.98. Adding the heights of features such as buildings and trees improved the differentiation between vegetation specifically where plants were dense. (Girard & Tarabalka, 2018) moved from a typical segmentation-vectorization approach to directly learning and output vectorial semantic labeling of the image, they propose a deep learning approach which predicts vertices of the polygons of Solar photovoltaic array dataset. However, the method is restricted to 4-sided polygons, so that the network has a fixed-length output. The loss function used to train the network is the mean L2 distance between the vertices of the ground-truth polygon and the predicted polygon. This assumes that both ground-truth and predicted polygons have their vertices numbered in the same way (same starting vertex and same orientation), so a different loss function which is invariant to the starting vertex should be used. This approach seems to be stagnant until geometric deep learning is operative, since it allows to encode nodes and their coordinates into a neural network. As another option, (Li et al., 2021) uses the U-Net, Cascade R-CNN, and Cascade CNN ensemble models to obtain building segmentation maps, building bounding boxes, and building corners, respectively, from very high-resolution remote sensing images. They later used Delaunay triangulation to construct building footprint polygons based on the detected building corners with the constraints of building bounding boxes and building segmentation maps. The ensemble approach promises to have a good performance to extract accurate building footprint polygons from remote sensing images, but the architecture is to complex and customized to only building objects. (Xie et al., 2020) proposes a method, using a multifeature convolutional neural network (MFCNN) and morphological filtering, for treating the irregular building boundaries resultant from high-resolution images. They showed that the method improved IoU by 3.04% respect to other segmentation-heuristic combined methods. (Sester et al., 2018) put together computer vision and cartography domains for studying the building footprint generalization problem, they describe main issues of post-processing masks as the interplay between different operators. Small parts of the building are eliminated (dissolve), the outline is simplified (simplification), neighboring objects can be merged (aggregation), too small buildings can be enlarged (enhancement), buildings are displaced (displacement). A network for improving cartographic generalization is trained and good results are obtained when compared using IoU metric. However, the benchmark for cleaning masks is the human operator, who can design an aesthetic and correct representation of the physical reality.

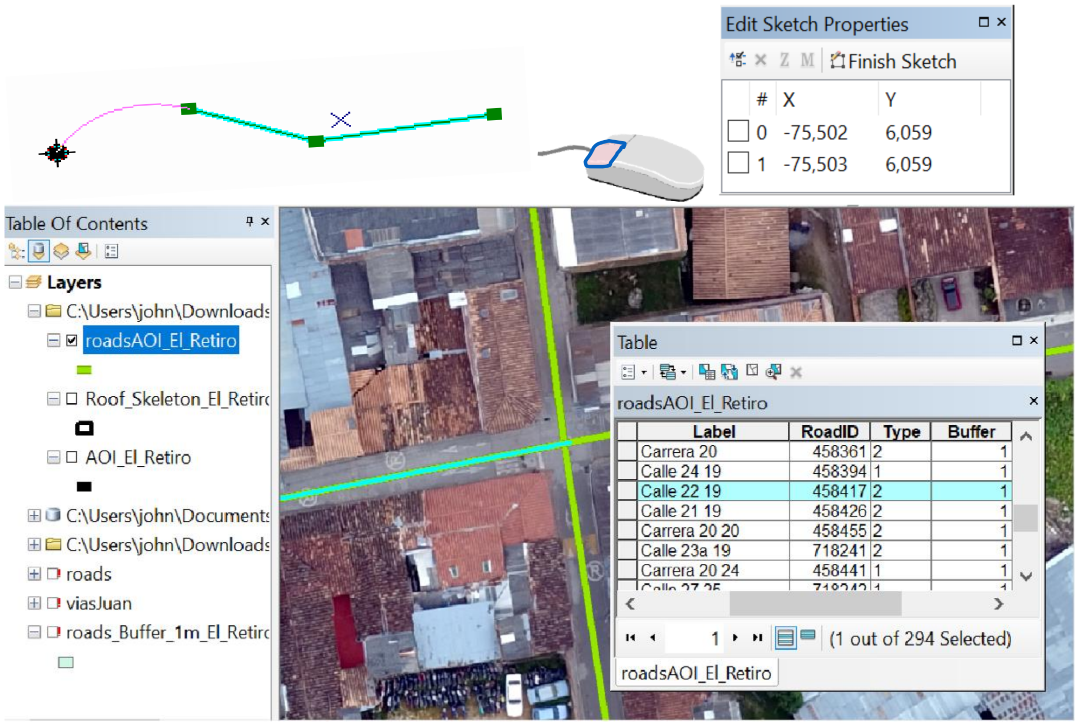
Extracting linear objects, like roads, from overhead imagery is dealt separately, and in contrast to polygon objects, it is characterized as imbalance. (Kearney et al., 2020) extracted unpaved roads from RapidEye imagery with 87 % precision and 89 % recall using CNN, hand engineering post-processing improved initial predictions. Results changed the public road database by 20 % through additions and removals. (Abdollahi et al., 2019) uses an immense range of image features, such as detectors for edge detection, filters for texture, filters for noise depletion and a membrane finder for road extraction from UAV imagery. Morphological operators are applied on the images for improving extraction precision and the road extraction precision is calculated based on manually digitized road layers. Customized methods little improved results and increased hand engineering involvement. Also (Abdollahi et al., 2020), compared four types of deep learning models: GANs, deconvolutional networks, FCNs, and patch-based CNNs. GAN based on U-Net model used was the second with best F1 Score performance on UAV and Google Earth images. Author highlights GANs as capable of achieving boundary information and smooth segmentation maps. Same author in (Abdollahi et al., 2021) presents the Road Vectorization Network (RoadVecNet), which comprises two interlinked U-Net networks to simultaneously perform road segmentation and road vectorization. The first network with powerful representation capability can obtain more coherent and satisfactory road segmentation maps even under a complex urban set-up. The second network is linked to the first network to vectorize road networks by utilizing all the previously generated feature maps. They utilize a loss function called focal loss weighted by median frequency balancing (MFB_FL) to focus on the hard samples, fix the training data imbalance problem, and improve the road extraction and vectorization performance over Google Earth imagery. (W. Yang et al., 2021) worked in a method of extracting roads and bridges from high-resolution remote sensing images, edge detection is performed, and the resultant binary edge is vectorized. Their network integrates binary cross entropy to deal with road class imbalance. (Gao et al., 2018) presented a weighted balance loss function over a PSPNet to solve the road class imbalance problem caused by the sparseness of roads. They compared with the cross-entropy loss function, and found that it can reduce training time dramatically for the same precision, especially for narrow rural roads. (Gong et al., 2020) uses a pre-trained VGG network into a U-Net and an attention module to solve road problems like tortuous shape, connectivity, occlusion, and different scales. Then the skeleton is extracted from the segmentation results and processed by denoising and straight-line connection and vectorized to realize the automatic road network. (Hartmann et al., 2017) proposed a GAN approach for road network synthesis called StreetGAN, with a post-processing step, they extract a graph-based representation from the generated images. Results were evaluated qualitatively and in terms of road network similarity measures. However, as a standard GAN its training uses random noise in generating resultant masks. This makes users are required to provide noise input, or rerun the generation several times due to the lack of control of the training process. (Pan et al., 2019) proposed a GAN with spatial and channel attention mechanisms (GAN-SCA) for the segmentation of buildings on the Inria and The Massachusetts Building Dataset. The generator is a U-Net-SCA networks, and the discriminator also uses an attention mechanism. Spatial and attention modules enable the segmentation to find features in specific positions and enhance results. Nevertheless, this architecture demands a larger number of examples and longer training times.

The increasing spatial resolution of overhead imagery, specially coming from drones, provides fine details for object extraction, but at the same time makes default segmentation methods to incur in ambiguities (Pan et al., 2019). Traditional segmentation, postprocessing and vectorization approach for GIS layer extraction is overloaded with hand engineering methods that are object and geometry specific.

2.2 Manual Digitalization and Object Attribution

Manual restitution and object attribution are the most time-consuming tasks in the process of creating a vector layer in a GIS software. The former consists of drawing on the pc screen, one by one, the shape of an object of interest. Prior to that, the user should have set a tolerance, or a distance within every location is considered coincident or identical, and it is commonly pre-defined in modern GIS Software as $\frac{1\,cm/100cm}{10} = 0.001$, this is, at least ten times smaller than the precision of digitalized data.

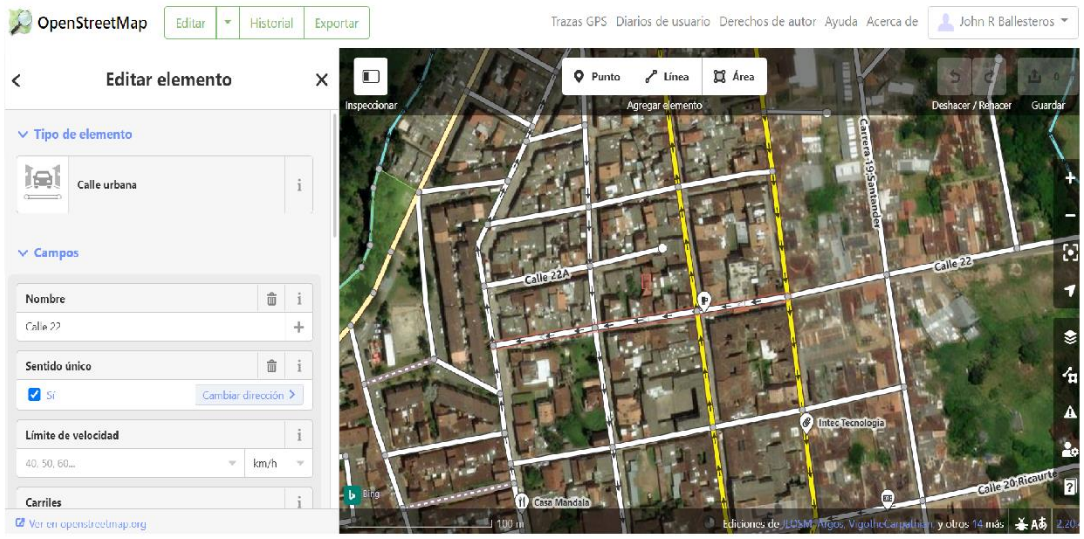
After vector objects have been drawn, characteristics should be assigned to each of them, for instance a road may have attributes like speed, name, lanes, surface, etc. These values should be written appropriately in an attribute table. Modern Web GIS applications have improved manual digitalization and attribution using predefined values that users can pick from to speed up the process. An example of this is Open Street Maps (OSM), an open source and community map database of the world. Figure 2.2. illustrates the manual digitalization and object attribution tasks.



#	X	Y
0	-75,502	6,059
1	-75,503	6,059

Label	RoadID	Type	Buffer
Carrera 20	458361	2	1
Calle 24 19	458394	1	1
Calle 22 19	458417	2	1
Calle 21 19	458426	2	1
Carrera 20 20	458455	2	1
Calle 23a 19	718241	2	1
Carrera 20 24	458441	1	1

(a)



Calle urbana

Nombre	Calle 22
Sentido único	✓ Sí
Límite de velocidad	40, 50, 60... km/h
Carriles	

(b)

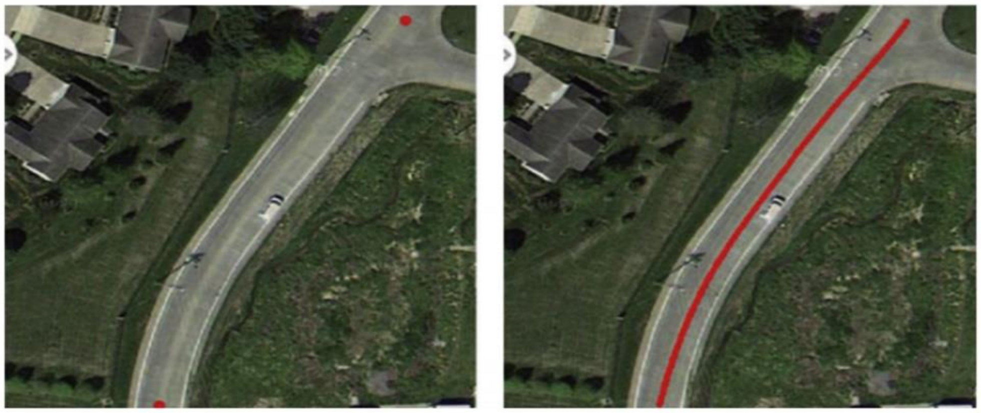
Figure 2.2. Manual digitalization and object attribution. (a) Manual sketch creation in a Standalone GIS, (b) Web GIS, OSM web app (*OpenStreetMap*/ www.osm.org).

2.3 Semi-automatic Methods for Object Extraction from Aerial Images

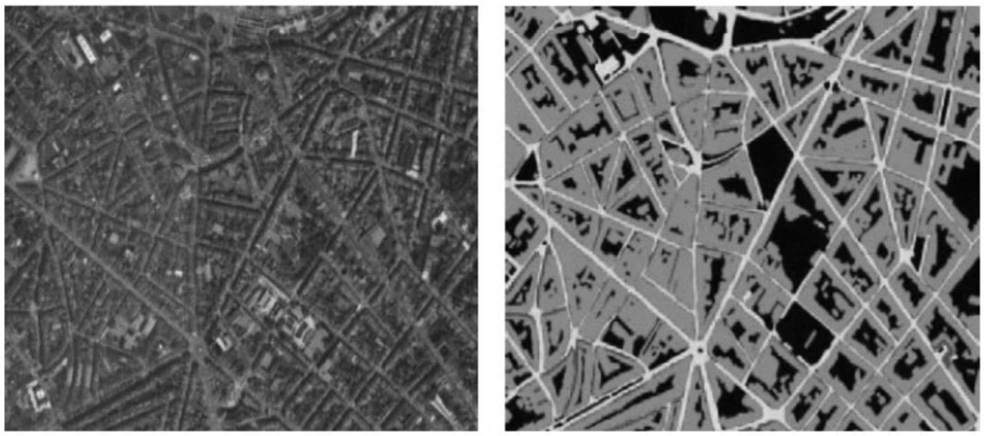
Semi-automatic methods refer to those that require user intervention, and the application of feature engineering to extract specific patterns on the image. This involves the tunning of a number of hyper parameters in the model or the combination of models

(Yan, 2019). Feature engineering consists of the use of spectral, texture, morphological, and boundary characteristics, as indicators to distinguish objects of interest present in imagery. Classification algorithms use texture, color, and geometry of objects, their performance is affected by the high variance of inter-class geographic objects and at the same time coincidence of these characteristics within the same class. For instance, an airport runway may have similar characteristics of a highway.

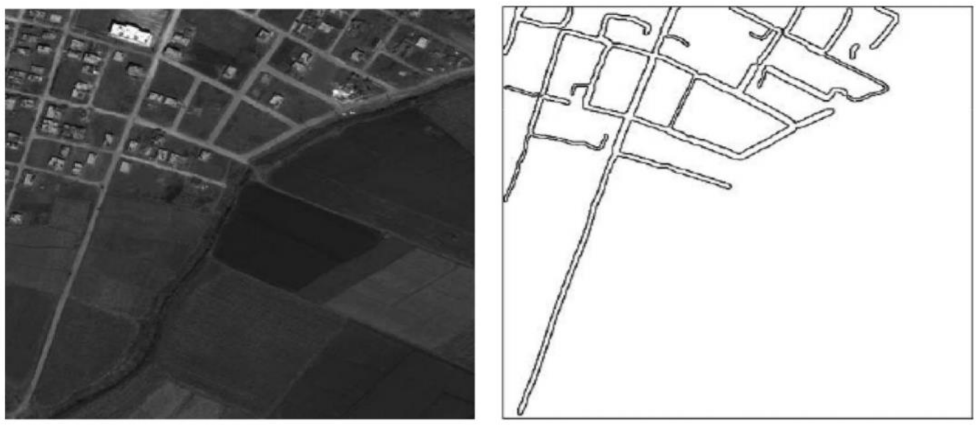
Dynamic Programming Algorithms like the LSB Snakes and the Geodesic (Gülch, n.d.; W. Wang et al., 2016) help in roads digitalization, the user must choose a road in the image, by pointing on the screen the initial and ending points of the object, these points are called seeds. Edge detection algorithms try to detect changes in spectral response of pixels, this is obtained by calculating gradients of the contrast in each image channel, three channels for RGB images, and only one for gray color images. For the gradient calculation, local filters show better results (ChiangYao-Yi et al., 2014). Graph-based algorithms have been used for the case of semi-automatic road extraction (W. Wang et al., 2016), they consider an image as a weighted graph, where a node is a pixel or region of pixels, and arcs are calculated by the difference between nodes. Result is obtained using a matrix of nodes. The limitation of this method is the number of nodes that can be processed and the different methods of pre-processing and post-processing that should be applied to obtain the results. Figure 2.3 results of some semi-automatic methods for road extraction from aerial imagery.



(a)



(b)



(c)

Figure 2.3. Semi-automatic methods for object extraction from aerial imagery. (a) Dynamic Programming Algorithms. Left is the input image, Right is the result, (b) Semi-automatic classification. Left is the input image, Right is the result, (c) Graph-based algorithms. Left is the input image; Right is the result.

2.4 Deep Learning for Image Semantic Segmentation

Today, deep convolutional neural networks (DCNN) have become the default algorithm for image semantic segmentation surpassing classical methods. Pixel-level classification is performed by converting input images into high dimensional arrays, after that, the process is reversed creating images with the same input dimensions. This encoding-decoding process is called autoencoding (Pashaei et al., 2020; H. L. Yang et al., 2018). Main architectures for semantic segmentation are presented next.

2.4.1 Fully Convolutional Network (FCN)

In image classification, an input image is re-scaled and goes through the convolution layers and fully connected (FC) layers, and outputs one predicted label for the input image. FCN turns FC layers into 1×1 convolutional layers and the image is not downsized; the output is not a single label. Instead, the output has a smaller size than the input image (due to the max pooling). Upsampling the output above, a pixelwise output (label map) can be obtained. Convolution produces smaller size output. Thus, the term deconvolution is coming from upsampling to get the output size larger. Deconvolution is misinterpreted as reverse process of convolution. It is also called up convolution, transposed convolution or fractional stride convolution. After going through up to seven levels of convolution layers (conv7) as below, the output size is small, then 32× upsampling (called FCN-32s) is done to make the output have the same size of input image. But it also makes the output feature map rougher. This is because, deep features can be obtained when going deeper, but spatial location information is lost. That means output from shallower layers have more location information. FCN combines both to enhance the result, it is done for each pixel, they are added from the results of different layers within a model. FCN is perhaps the most basic models for semantic segmentation (J. Long et al., 2015). Figure 2.4 illustrates its architecture.

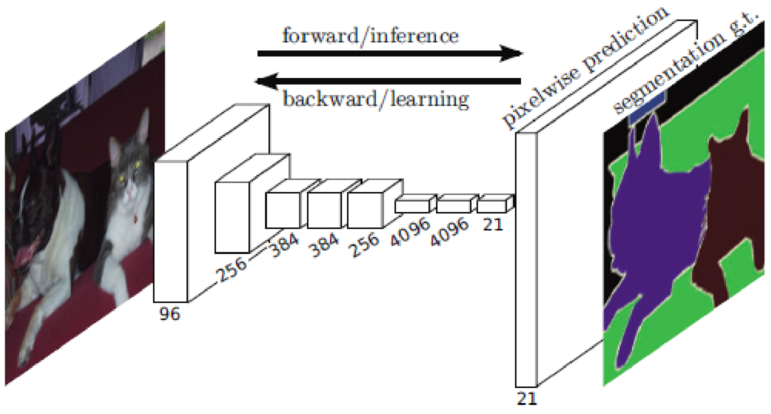


Figure 2.4. Fully Convolutional Network (FCN). Modified from (J. Long et al., 2015).

2.4.2 DeepLab

An input image goes through the network using an atrous convolution (r>1 in equation 2.1), or also called dilated convolution. The output is bilinearly interpolated and then goes through the fully connected conditional random field (CRF) layer that fine-tunes the result and get the final output. Equation 2.1. shows values in an atrous convolution.

$$y[i] = \sum_{k=1}^{K} x[i + r * k]w[k], \text{(2.1)}$$

When rate of sampling (r = 2), the input signal is sampled alternatively. It means padding two zeros at both left and right sides of filters. Sampling the input image every 2 inputs for convolution. Thus, output will have 5 outputs which makes the output feature map larger. The atrous convolution allows this network to enlarge the field of view of filters to incorporate a larger context. It is an efficient mechanism to control the field-of-view and finds the best trade-off between accurate localization (small field-of-view) and context assimilation (large field-of-view). The output is much larger than in FCN, it only needs to have 8× upsampling to up sample the output and bilinear interpolation has good performance for the 8× up sampling. Fully Connected CRF is applied at the network output after bilinear interpolation. With 10 times of CRF, small areas with different colors around the objects are smoothed out successfully. Since CRF is a post-processing task, it makes DeepLab not an end-to-end learning framework, and it is why CRF is not used in later versions of the architecture. This model presents some failure examples where the objects of interest consist of multiple thin parts (Cheng et al., 2020). Figure 2.5 shows the DeepLab architecture steps.

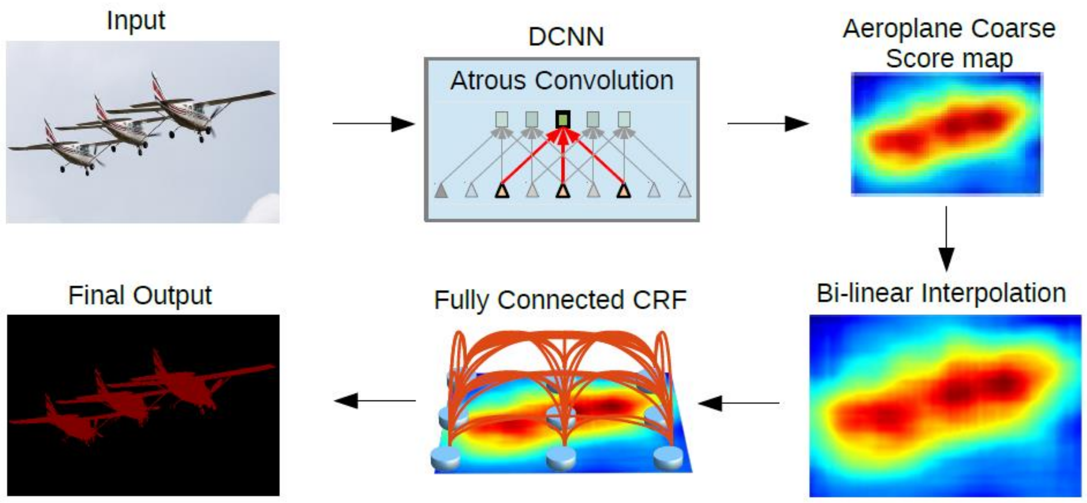


Figure 2.5. DeepLab steps. Modified from (Cheng et al., 2020).

2.4.3 Pyramid Scene Parsing Network (PSPNet)

The PSPNet challenges three issues observed in the FCN, these are:

- Mismatched Relationship: FCN predicts objects relationships wrongly. For instance, a boat can be predicted as a "car" based on its appearance, ignoring spatial relationships between objects or common knowledge, for instance, a car is seldom over a river.

- Confusion Categories: FCN predicts mixed objects. For instance, an object is either a skyscraper or a building, but not both.

- Inconspicuous Classes: Overlooking the global scene category may fail to parse small objects. For instance, a pillow has similar appearance with a sheet, so it is not segmented.

PSPNet highlights global information of the image using a pyramid pooling module that works as follows. An input image is passed to a dilated CNN based on ResNet, extracting feature maps of 1/8 of the input image. At (c) pyramid pooling module consists of four sub-region average pooling that is performed for each feature map (red, orange, blue and green blocks). Red is the coarsest level which perform global average pooling over each feature map, to generate a single bin output. Orange divides the feature map into 2×2 sub-regions, then performs average pooling for each of them. Blue divides the feature map into 3×3 sub-regions, then perform average pooling for each sub-region. Green is the finest level which divide the feature map into 6×6 sub-regions, then perform pooling for those regions. Then, 1×1 convolution is applied to each pooled feature map to reduce the context representation to 1/N of the original one (black) if the level size of pyramid is N. If the input feature map is 2048, then the output feature map will be (1/4) × 2048 = 512. Bilinear interpolation is performed to up-sample each low-dimension feature map to have the same size as the original feature map (black). All levels of feature maps are concatenated with the original feature map (black). These feature maps are fused as global prior. That is the end of pyramid pooling module at (c), finally, it is followed by a convolution layer to generate the final prediction map at (d). PSPNet is the champion of ImageNet Scene Parsing Challenge 2016. It also got the 1st place on PASCAL VOC 2012 & Cityscapes datasets. It is published in 2017 CVPR with more than 600 citations (H. Zhao et al., 2017). Figure 2.6 shows the PSPNet architecture.

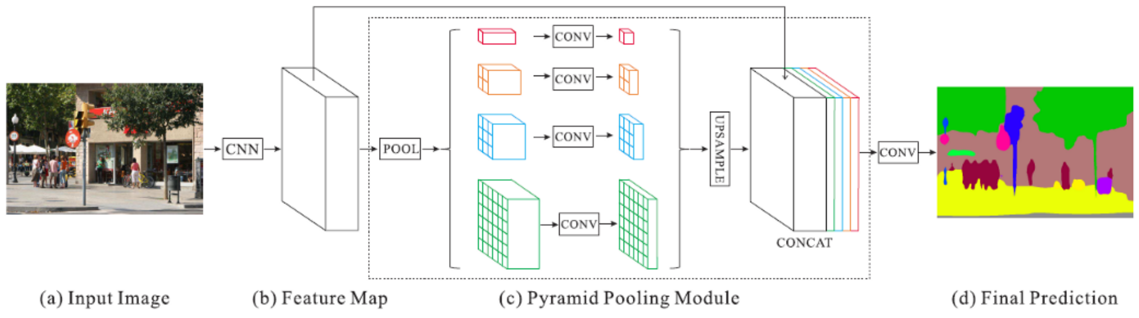


Figure 2.6. PSPNet architecture. Modified from (H. Zhao et al., 2017).

2.4.4 SegNet

The SegNet counts on an encoder-decoder network, followed by a final pixel-wise classification layer. At the encoder, there are 13 convolutional layers from VGG-16 and 2×2 max pooling layers that make indices (locations) are stored. At the decoder, up-sampling and convolutions are performed. During up sampling, the max pooling indices at the corresponding encoder layer are recalled to up sample. At the end, there is a K-class Softmax classifier that is used to predict the class for each pixel. It outperforms FCN, and DeepLab, obtaining a higher global average accuracy (G) and showing strength in large-size classes. It has low memory requirement during both training and testing and the model size is much smaller than FCN, however, SegNet is slower than FCN and DeepLab, because of the decoder architecture (Badrinarayanan et al., 2017). Figure 2.7 shows the architecture of SegNet.

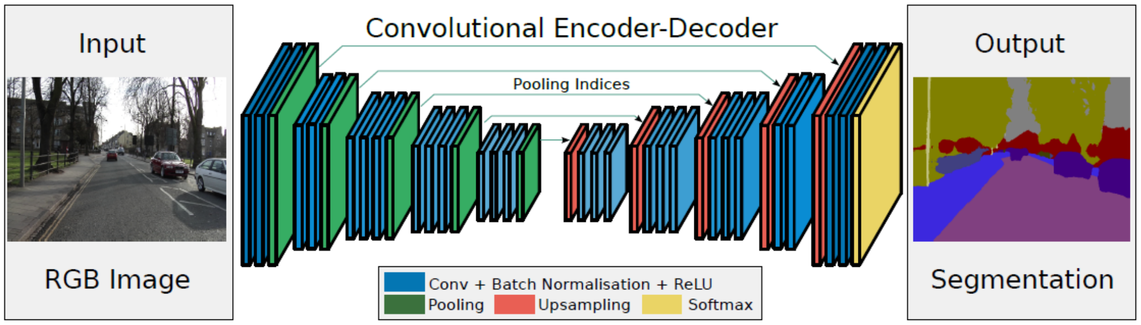


Figure 2.7. SegNet architecture. Modified from (Badrinarayanan et al., 2017).

2.4.5 U-Net

The U-Net was initially designed for the segmentation of Electron Microscopic (EM) biomedical images. It is based on a convolutional network that consists of contraction and expansion path with an ‘U’ shape, from which it takes its name. The encoding left side has a consecutive of two times of 3×3 convolution and 2×2 max pooling. This helps to extract advanced features and reduces the size of feature maps. The expansion right path is a consecutive of 2×2 up-convolution and two times of 3×3 convolutions. It is done to recover the size of segmentation map. However, the process reduces the spatial information though it increases the feature information. That means, it gets advanced features, but losses the localization information. Thus, after each up-convolution, it has concatenation of feature maps that are within the same level. This concatenation retrieves localization information from the contraction path to the expansion path. At the end, a 1×1 convolution maps the feature map size from 64 to 2, since the output feature map only have 2 classes in the original implementation (Ronneberger et al., 2015). Figure 2.8 shows the U-Net architecture, gray arrows represent the skip connections to perform concatenation of feature maps at the same level. The U-Net advantage, over other architectures for the semantic segmentation of geographic objects, is its capacity to segment using less training examples, but its performance is low for linear and small objects (compared to the total size of image) (Pashaei et al., 2020).

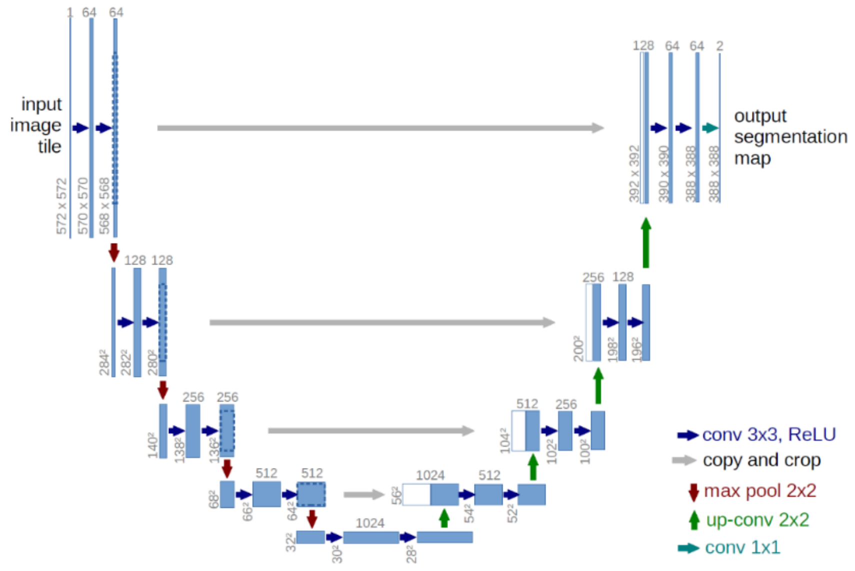


Figure 2.8. U-Net. Modified from (Ronneberger et al., 2015).

Table 2.1 compares the previous segmentation architectures over three important geographical objects: vegetation, roads, and water (Pashaei et al., 2020). Results show that U-Net surpasses the rest, except the FC-DenseNet, but it uses less training examples and inference time.

Table 2.1. Comparison of architectures for the segmentation of geographical objects. Modified from (Pashaei et al., 2020).

Algorithm	OA-Train	OA-Val	Prec	Recall	F1	mIoU	Veg	TF	Water	Road
FC-DenseNet	0.97	0.95	0.95	0.95	0.95	0.90	0.96	0.93	0.94	0.96
U-Net	0.96	0.95	0.95	0.94	0.94	0.91	0.95	0.93	0.95	0.95
DeepLabV3+	0.94	0.93	0.91	0.90	0.90	0.89	0.94	0.88	0.87	0.89
PSPNet	0.91	0.89	0.89	0.88	0.88	0.83	0.96	0.89	0.87	0.83
MobileU-Net	0.89	0.85	0.88	0.79	0.84	0.75	0.97	0.85	0.69	0.76
SegNet	0.88	0.82	0.91	0.81	0.82	0.69	0.97	0.77	0.65	0.85

2.5 Metrics for Semantic Segmentation

Pixel-level classification is evaluated in terms of number of pixels considered as true positive, false positive, and false negative. $Precision$ is the ratio of the true positive pixels to all detected positive class. $Recall$ is the ratio of the true positive pixels to the reference or ground truth pixels. These metrics are defined in the equations 2.2. and 2.3 (Deigele et al., 2020).

$$Precision = \frac{TP}{TP+FP} \qquad (2.2)$$

$$Recall = \frac{TP}{TP+FN} \qquad (2.3)$$

Where, TP or true positive denotes the number of pixels correctly classified (e.g., road pixels as road pixels), FP or false positive are the number of non-positive class pixels misclassified as positive class (no building pixels as building), and FN or false negative denotes the number of positive class pixels that are not detected.

The intersection over union (IoU) metric, also known as the Jaccard index, is scale invariant and provides a measure of the overlap between two objects by dividing the area of the intersection by the area of the union as stated in Equation 2.4.

$$IoU(A,B) = area(A \cap B)/area(A \cup B) \qquad (2.4)$$

IoU is the ratio of the true positive pixels to the total number of true positive, false positive, and false negative pixels. It ranges from 0–1 (0–100%), where 0 means no overlap, and 1 the perfect overlapping segmentation. Equation 7.3. can be written as Equation 2.5.

$$IoU = \frac{TP}{TP+FP+FN} \qquad (2.5)$$

IoU is widely used in the evaluation of model performance for image segmentation, as it provides a measure that penalizes false positive pixels. The (IoU) has been extensively used in the assessment of road and building extraction results from overhead imagery (Van Etten et al., 2019) due to its ability to incorporate the geometrical aspect of the resultant masks. Figure 2.9. illustrates the geometrical aspect of IoU.

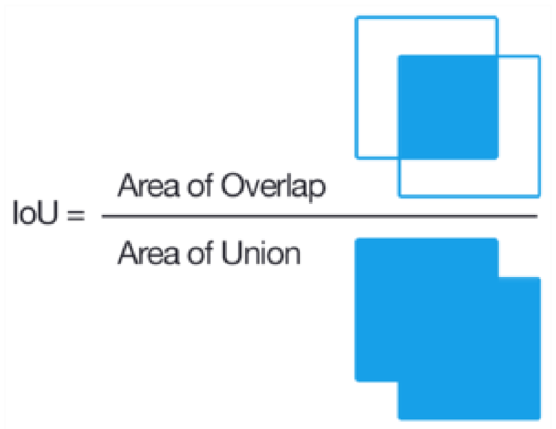


Figure 2.9. Geometrical aspect of IoU Metric. Modified from (Pashaei et al., 2020).

2.6 Deep Generative Models

Generative models address the challenging task of estimating a distribution of a given high dimensional data $P_d(x)$, generating samples from random which have an approximate distribution $P_\theta(x)$ of the training data, in such a way that $P_d(x) \doteqdot P_\theta(x)$. Considering a random input z sampled from a tractable distribution $p(z)$ supported in Rm and a training data intractable distribution supported in Rn. The objective of a trained generator g_θ can be written as Equation 2.6.

$$g_\theta: R^m \rightarrow R^n,\ such\ that, \min_\theta dist\ (P_d(x), P_\theta(x)) \quad (2.6)$$

The random vector z is referred as a latent variable sampled from a latent space that in general, follows a Gaussian distribution with a mean of zero and a standard deviation of 1 (Salimans et al., 2016). The distance minimization problem of Equation 2.6. can be tackled in the Equation 2.7. Once the generator g_θ is trained, the likelihood of the generated sample x from the latent variable z is:

$$P_\theta(x) = \int P_\theta(x|z)P(z)\, dz \quad (2.7)$$

Where $P_\theta(x|z)$ is the closeness of the generated $g_\theta(z)$ to the sample x (Goodfellow et al., 2014). Approximating the generator function utilize various deep neural networks architectures, e.g., CNN and RNN, by computing the generator parameters θ.

There exist two main classes of generative models, the explicit and implicit. The explicit models are likelihood-based and use an explicitly defined $P_\theta(x)$. Implicit models learn data distribution directly from training data without any prior model structure, using for instance, Generative Adversarial Networks or GANs (Goodfellow et al., 2014), which are unsupervised learning methods. The main purpose of generative models is to create synthetic data, this has many applications such as tackling datasets imbalance, perform image to text, image to image translation, image inpainting, synthesis of image-speech and sound, and super resolution.

2.6.1 Generative Adversarial Networks (GANs)

GANs learn a generation function from a training distribution using a two-player game of neural networks called the generator and the discriminator (or critic). The generator tries to fool the discriminator by generating images that look real. The discriminator tries to distinguish real-training data ($x \sim P_d(x)$) from generated images. The main challenge is to simultaneously train these two networks in a way that both get better in every iteration, for this, the minimax loss function calculates the difference between the estimated and the training distribution as in Equation 2.8.

$$\min_{p_{\theta g}} \min_{D_{\theta d} \in F} \mathbb{E}_{x \sim p_d}[D_{\theta d}(x)] - \mathbb{E}_{x \sim p_{\theta g}}\left[D_{\theta d}(G_{\theta g}(x))\right] \quad (2.8)$$

Where F is a set of functions and the loss computes the distribution $p_{\theta g}$ that minimizes the overall discrepancy, the first term of Equation 2.8 represents the discriminator which uses directly the training dataset, the second part corresponds to the generator that uses the latent space to create synthetic data. Equation 2.3 can be rewritten in terms of the two players via logarithm likelihood in Equation 2.9.

$$\min_{\theta_g} \min_{\theta_d} \left(\mathbb{E}_{x\sim p_d}[logD_{\theta d}(x)] + \mathbb{E}_{z\sim p_z}\left[log\left(1 - D_{\theta d}\left(G_{\theta g}(z)\right)\right)\right]\right) \quad (2.9)$$

The discriminator needs to maximize the objective function for θ_d such that $D_{\theta d}(x) \approx 1$, which leads the output to be close to the real data. The term $D_{\theta d}\left(G_{\theta g}(z)\right) \rightarrow 0$ as it is fake data, therefore, the maximization of the function for θ_d will ensure that the discriminator can separate real and fake data. The generator needs to minimize the objective function for θ_g such that $D_{\theta d}\left(G_{\theta g}(z)\right) \approx 1$. If this happens the discriminator will classify generated data as real. The optimization of the objective function is performed in two steps repeatedly. First, the gradient ascent is used for the discriminator term, then the generator term is minimized using gradient descent. In practice, the second step is not feasible because the term $log\left(1 - D_{\theta d}\left(G_{\theta g}(z)\right)\right)$ is dominant when $D_{\theta d}\left(G_{\theta g}(z)\right) \approx 1$. Since the opposite behavior is required, the gradient ascent should instead maximize the modified generator term $\max_{\theta_g} \mathbb{E}_{z\sim p_z}\left[log\left(D_{\theta d}\left(G_{\theta g}(z)\right)\right)\right]$, and then, the generator is trained and can be used separately. Figure 2.10. illustrates the process when training a GAN, the image is modified from (Goodfellow et al., 2014)

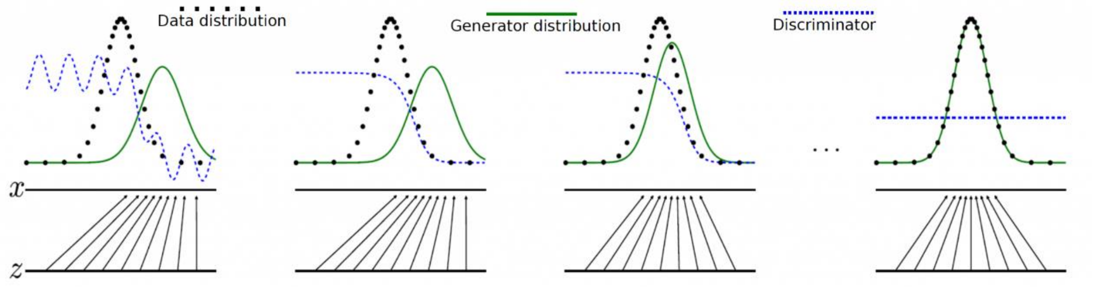


Figure 2.10. GANs training. A random vector z is input to the generator to create a fake output $x \sim P_d(x)$ (green curve), that is far from original distribution P_d (black curve). Therefore, the discriminator classifies this output as synthetic forcing the generator to generate outcomes closer to the data distribution. Finally, the discriminator is unable to detect between real or generated data. Modified from (Goodfellow et al., 2014).

A limitation of GANs is the unstable training process, where discriminator and generator train in parallel. The discriminator sends gradients to the generator, if these values are beyond a specific interval, the process diverges, the losses of the discriminator go to zero and the loss of the generator explodes. This causes the generator to create artifacts or images that are far from the training dataset (Radford et al., 2016). Doing data augmentation increases the risk of instability due to the change of data distribution, so a proper data augmentation for both discriminator and generator is required (S. Zhao et al., 2020).

2.6.2 Image Generation and Image to Image Translation

Image generation is the process of obtaining an image from a random space, this is, the obtention of synthetic images that plausibly resemble the characteristics of existing distribution of training samples. Main application of image generation is synthetic data creation for model training, computer graphics, and design (Isola et al., 2017). Image generation is typically performed by generative models.

Image to image translation is the conversion of a source image to a target image. For example, the conversion of satellite images to google maps images or the reverse (Isola et al., 2017). It is a challenging problem that requires custom models and loss functions for every translation task or dataset. Common approaches use pixel-wise classification models, but main issue is that each predicted pixel is independent of the pixels predicted before it and the broader structure of the image might be missed. These consider the output space as “unstructured”, in the sense that each output pixel is regarded as conditionally independent from all others of the input image. Image translation has demonstrated its power on a range of interesting tasks (Isola et al., 2017; Zhu et al., 2017), for example:

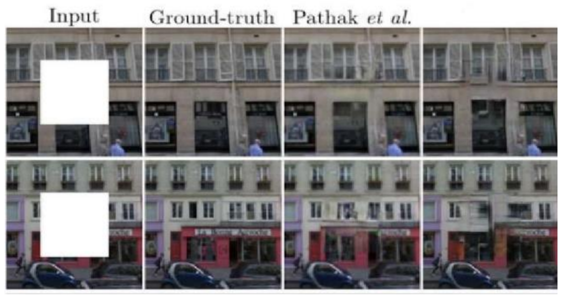
- Semantic labels to images: (Cordts et al., 2016) has successfully performed translation of semantic images to photographs using the Cityscapes dataset. (Isola et al., 2017) presented an example for the conversion of architectural semantic labels of building facades to photographs, it was trained on the Facades dataset.

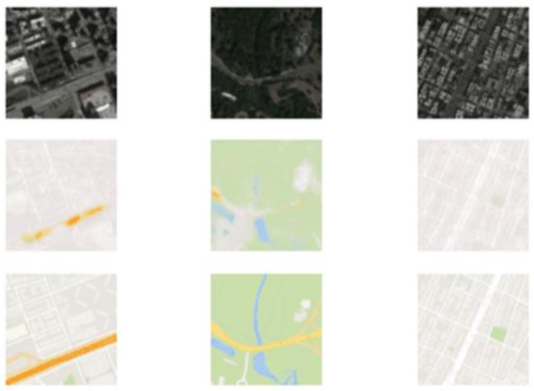
- Satellite to Map and reverse: Data can be scraped from google maps or other map service to perform the translation between maps and satellite (Isola et al., 2017).

- Black and white to color photographs, day to night photographs, thermal images to color photographs (*Machine Learning Mastery*/ www.machinelearning mastery.com).

- Product sketches to photographs or edges to photographs: this is an important application in fashion and retail (Goodfellow et al., 2014).

- Image inpainting: it consists of replacing white or black pixels present in an image to generated pixels in some sort that the resultant image has appropriate context. Figure 2.11 modified from (Isola et al., 2017) illustrates image translation examples.





(a) (b)

Figure 2.11. Image Translation Examples. (a) Image inpainting, (b) Satellite to Map Translation: Satellite to map image to image translation example using pix2pix. The model hardly distinguishes the street details in the first epochs of training (left image), after 100 epochs different type of roads, parks and buildings are well differentiated (central image), when 150 epochs of training are reached (right image), fine details are improved for all the objects.

2.6.3 Conditional and Unpaired Image to Image Translation

Conditional image to image translation is the controlled conversion of a source to a target image. Conditionally means that the loss function makes the generated image to be plausible both in the content of the target domain, and as a translation of the input image. A conditional generative adversarial network (cGAN) architecture (Isola et al., 2017) is an extension of the GAN architecture (Goodfellow et al., 2014; Isola et al., 2017) to train a generator model, used for generating images in a conditional manner. This is, the cGAN controls that the image that is generated belongs to a specific class. In the same way as GANs, the discriminator model of cGAN is trained to classify images as real (from the training dataset) or generated (synthetic images), and the generator of cGAN is trained to fool the discriminator. For cGANs to learn a conditional generative model of a data distribution it requires a paired dataset, described in a previous section.

Unpaired image to image translation is the conversion of a source image to a target image without the need of a pixel-to-pixel paired dataset (Zhu et al., 2017). This method can learn to capture special characteristics of one image collection and figuring out how these characteristics could be translated into other image collection, all in the absence of any paired training examples, or conversely using an unpaired dataset. A successful approach for unpaired image-to-image translation is the CycleGAN architecture. The CycleGAN (Zhu et al., 2017) is also an extension of the GAN architecture (Goodfellow et al., 2014) in which one generator takes images from the first domain as input, and outputs images for the second domain, and other generator takes images from the second domain as input and generates images from the first domain. Two discriminator models are then used to determine how plausible the generated images are and update the generator models accordingly. This extension alone might be sufficient to generate plausible images in each domain, but not enough to generate translations of the input images. This means that adversarial losses alone cannot guarantee that the learned function can map an individual input x_i to a desired output y_i. To solve that, CycleGAN uses an additional extension called cycle consistency, this is that an image output by the first generator could be used as input to the second generator, and the output of the second generator should match the original image. The reverse is also true: an output from the second generator can be fed as input to the first generator and the result should match the input to the second generator. Cycle consistency comes from neural machine translation (NMT) where a phrase translated from English to Spanish should translate from Spanish back to English and be identical to the original phrase. The reverse process should also be true. Cycle consistency is encouraged by adding an additional loss to measure the difference between the generated output of the second generator and the original image, and the reverse. This acts as a regularization of the generator models, guiding the image generation process in the new domain toward image translation (Zhu et al., 2017).

Unpaired image translation has many applications such as: Style Transfer, that refers to the learning of artistic style, often paintings, and applying it to another domain, such as photographs. Object Transfiguration refers to the transformation of objects from one class, such as apples into another class of objects, such as oranges. Season Transfer refers to the translation of pictures taken in one season to another, such as summer to winter. Photograph generation from paintings is the synthesis of photorealistic images given a painting, typically by a famous artist or scene. Finally, Image Enhancement refers to improvements of images in specific aspects (Zhu et al., 2017).

2.6.4 Pix2Pix

The Pix2Pix GAN architecture is an implementation of the cGAN, where the generation of a target image is conditional on a given input image (Isola et al., 2017). The generator model is provided an image as input to generate a translated version of the image. The discriminator model is given an input image and a real or generated pixel-paired image to determine whether the paired image is real or generated. Finally, the generator model is trained to both fool the discriminator model and to minimize the loss between the generated image and the expected target image. As such, the Pix2Pix GAN must be trained on image paired datasets that are comprised of source images (before translation) and target images (after translation). The Pix2Pix architecture is a general approach for conditional image translation, it has been trained and tested for a wide range of image translation tasks (Isola et al., 2017). It involves three components: a generator model, a discriminator model, and a model optimization procedure. Both the generator and discriminator models use standard Convolution-BatchNormalization-ReLU blocks of layers as in any deep convolutional neural networks.

The generator model is based on the U-Net, and unlike a standard GAN model (Goodfellow et al., 2014), it does not take a random point z from the latent space as input. Instead, it takes an image from source domain as input (e.g., satellite), and produces an image in the target domain (e.g., map). The source of randomness comes from the use of dropout layers that are used both during training and prediction (Isola et al., 2017).

The discriminator model takes an image from the source domain and an image from the target domain and predicts the likelihood of whether the image from the target domain is real or generated. Pix2Pix discriminator model uses a PatchGAN which is a network designed to classify patches of an input image as real or synthetic (Isola et al., 2017), rather than doing it over the entire image. The discriminator tries to classify if each N×N patch (70x70) of an image is real or generated by the model. The Patch is run convolutionally across the image, averaging all responses to provide an ultimate scalar value. The PatchGAN discriminator model is implemented as a CNN, but the number of layers is set such that the receptive field of each output of the network maps to a specific size in the input image (70x70). The PatchGAN output is a single feature map of real/synthetic predictions that are averaged to a score. A patch size of 70×70 pixels was found to be the most effective across a range of image-to-image translation examples (Isola et al., 2017).

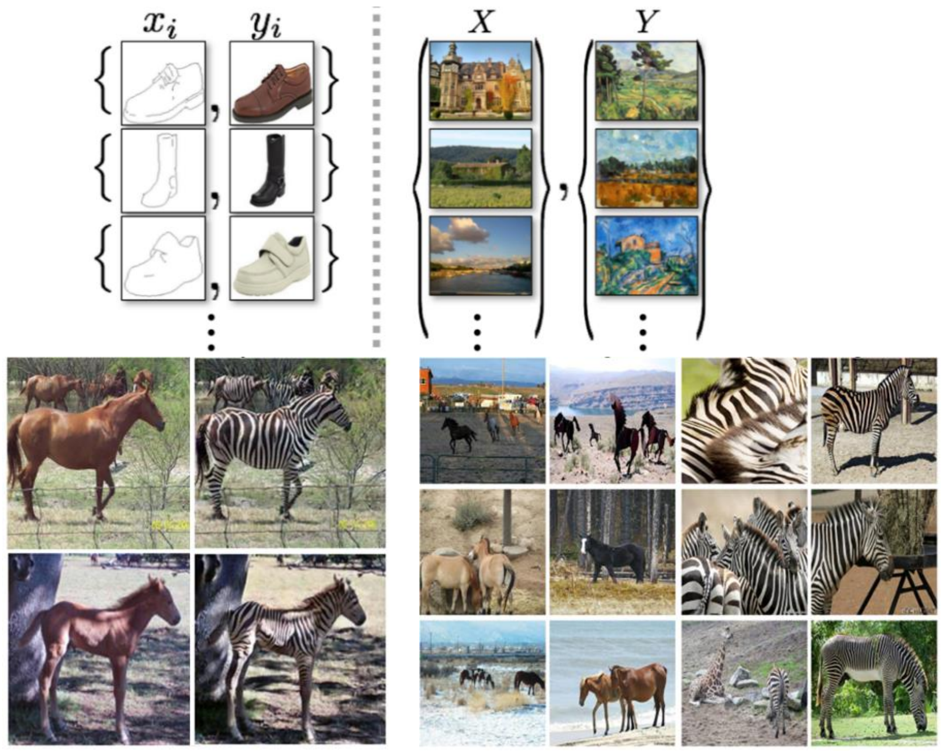
A standard GAN model is trained in a standalone manner (Goodfellow et al., 2014; Salimans et al., 2016), Pix2Pix discriminator model does it the same way, minimizing the negative log likelihood of identifying real and generated images, although conditioned on a source image. However, since the discriminator training is too fast compared to the generator, the discriminator loss is halved $\frac{Discriminator\_Loss}{2}$ in order to slow down the discriminator training. The generator model is trained using both the adversarial loss with the discriminator, and the mean absolute error (MAE) or L1, this is the pixel difference between each generated translation of the source image and the corresponding expected target image in the training dataset. The adversarial loss and the L1 loss are combined into a composite loss function, which is used to update the generator model. In the original paper by (Isola et al., 2017), L2 loss (sum of squared differences) was also evaluated and resulted in blurry images. The adversarial loss conditions the generator model to output images that are plausible in the target domain, the L1 loss regularizes the generator to output images that are a plausible translation of the source image. In this way, the objective of the discriminator does not change, but the generator needs to fool the discriminator, but at the same time, get outputs near to the ground truth. The composite loss function is controlled by a hyperparameter called lambda (λ), which ranges between 10 to 100, e.g., giving 10 to 100 times the importance of the L1 loss compared to the adversarial loss during the generator training. Equation 2.10. shows the composite loss function for the generator.

$$\text{Generator Loss} = \text{Adversarial\_Loss} + \lambda \times \text{L1\_Loss} \qquad (2.10)$$

2.6.5 Paired and Unpaired Datasets in Deep Learning

A paired dataset is formed by examples of input images x_i (e.g., shoes-sketches) and the expected target images y_i (e.g., shoes) coupled pixel to pixel. The target images also called masks or class labels exhibit the desired modifications of input images at a pixel-level. Paired datasets are challenging and expensive to produce, since it requires the pixel-to-pixel correspondence, for instance, taking pictures of different scenes under different conditions is difficult to comply. Moreover, in many cases, these datasets do not exist or cannot be produced due to copy rights, as with the famous artists paintings and their respective photographs (Isola et al., 2017). Paired datasets are specifically used in semantic segmentation problems, where an input image is classified into an output binary (black and white) or multi-color mask image.

In contrast, an unpaired dataset consists of two groups of images X (e.g., horses), and Y (e.g., zebras) that are not related in a pixel-wise manner, but they have general characteristics that can be extracted from each group and used to convert from X to Y. This type of dataset is more used in semi-supervised or un supervised learning. Figure 2.12. presents examples of paired and unpaired datasets (Zhu et al., 2017).



(a) (b)

Figure 2.12. Paired and Unpaired datasets. (a) Horses to zebras paired dataset, (b) Horses to zebras unpaired dataset.

Chapter 3

3 A Methodology to Extract GIS Vector Layers from Orthomosaics using a Deep Generative Model

This chapter is a brief description of the methodology, composed by a pipeline of three consecutive methods for vector objects extraction from orthomosaics. It starts with the production of image-mask paired datasets for points, lines, and polygons using drone orthomosaics. After that, a deep generative method is chosen and used to create masks for objects of interest in input images. The post-processing method applies a generative model to clean and improve the resultant masks of the second method and convert them to vector. The three methods form the core of the thesis, they are introduced next, and further described and tested in the coming chapters. Figure 3.2 illustrates our methodology.

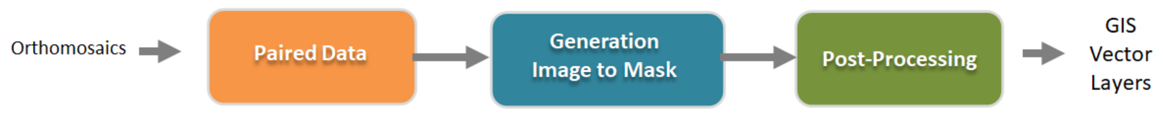


Figure 3.1. Proposed methodology.

3.1 Paired Data Method

This method produces balanced and feature-enriched datasets to make generative model be more robust to geometric, spectral, and multi-scale variations of geographic objects. It also pretends to speed up and improve the training of the generative model when applied to point, line, and polygon masks objects. Datasets consist of three channels images chips and corresponding masks (img, msk) paired pixel to pixel, i.e., (drone image chip, binary mask). Separate datasets are produced and used to extract point, line, or polygon objects. Figure 3.2. summarizes this method.

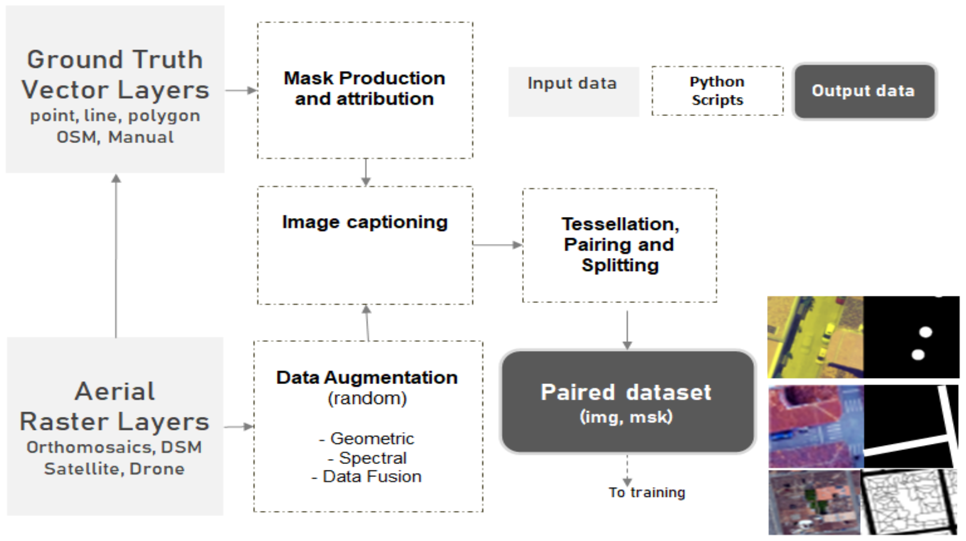


Figure 3.1. Summary of the method to produce paired datasets.

As shown in Figure 3.1, the workflow starts with two separate pipelines, one is the Ground Truth (GT), consisting of points, lines, or polygon objects of interest in vector format (Shapefiles, or GeoJSON) that can be queried from OSM. The Aerial Raster Layers are composed by the orthomosaics and the DSM obtained by satellite or drone. Both pipelines get together at tessellation and captioning steps, where specific chip sizes for images and masks are produced. The imbalance check is performed for every mask, and then pairing chips into single images of the form (img,msk) are produced. Finally, a random split into the training, validation, and testing datasets is applied using a desired percentage. All these steps are described thoroughly in the next chapter.

3.2 Generative Method for Image to Mask Translation

Image to image translation is a technique based on deep generative models, in which a model learns target labels (masks) from images in the source domain (e.g., satellite, drone orthomosaics) (Ballesteros et al., 2021). We propose to use image to mask translation as an alternative of semantic segmentation to extract vector layers from orthomosaics. Conditional and unpaired generative models for image-to-mask translation are implemented and compared. After results, the best option to generate masks for point, line, and polygon objects from drone orthomosaics is chosen. The chosen model needs to tackle the following challenges:

- **Quality masks:** Model should generate masks with a high similarity to the ground truth, especially in the case of objects with right angles (man-made objects). This should produce a smaller number of false positives compared to other models.

- **Data requirements:** Since most projects in geography does not have a huge amount of data, model should be less data hungry. Paired or unpaired datasets might be a requirement.

- **Class imbalance:** if the model is affected by imbalance, it should provide a way to mitigate it.

- **Same model architecture valid for point, line, and polygon masks:** The same model architecture can be trained to generate masks for point, line, and polygon objects without the need of manual engineering.

- **Cleaning and simplification of masks:** Model can support a way to clean and simplify masks.

- **Attribute extraction:** Model can be used to obtain binary masks, but also multi-class-colored masks employed to automatically extract attributes or object types.

3.2.1 Use of Height Information in Generative Models

Few recent studies have integrated CNNs with height geometric information coming from a DSM and normalizing with a DTM (Sun & Wang, 2018; Zhang et al., 2015). Typical two approaches are whether to input a fourth channel with the height information to a network, or having two networks that receive RGB and height information separated and concatenate results at the end (Marmanis et al., 2020). Results in building segmentation and land cover maps showed that the use of DSM highly improved the pixel accuracy (1.2% up to 1.8%) (Al-Najjar et al., 2019). Also, results were better when normalizing DSM with a DTM to obtain local height and when using a fourth channel in the network. In the future section dedicated to the generative model, our approach is to fuse height information into the three channels of RGB images and compare it with replacing the blue channel with the height information.

3.2.2 Automatic Attribute Extraction in Generative Models

The attribute extraction of objects is a part of the automatic layers extraction process. Attribution is the process of giving characteristics to each object and store them in a table in an organized manner. In a GIS, object attribution is as important as geometry, and coordinates of objects. Since masks can be colored, color can be used to encode a specific attribute (Van Etten, 2019). In the same way, color decoding is a way to perform attribute extraction. For instance, a road can be green for high speed, yellow for medium, and red for low-speed values. Figure 3.2. shows an example of attribute extraction via color encoding-decoding. In chapter 5, we will test color encoding-decoding in a generative model with the purpose of extracting attributes from our road dataset.

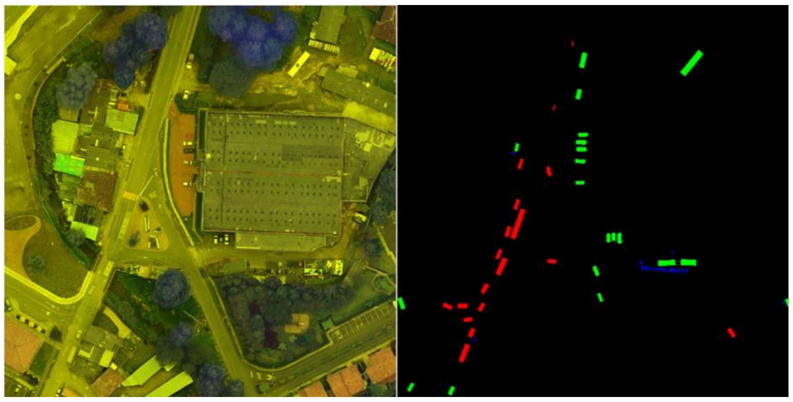


Figure 3.2. Color encoding-decoding for attribution. Image shows type of vehicles encoded by color. Blue are motorcycles (code=1), green represents cars (code=2), and red color are ghosts of cars (code=3). Image modified from (Ballesteros, 2022).

3.3 Post-processing Method

Normally, post-processing consists in the application of several heuristic and consecutive methods prior to vectorization for cleaning resultant masks of deep learning models (Pote, 2021; Ren & Xu, 2015; Saeedimoghaddam & Stepinski, 2020). It is also common to use a set of tools after vectorization to improve resultant vector layers.

3.3.1 Post-processing of Resultant Masks

There are a set of tools commonly used to post-process resultant masks, generally irregular and discontinuous, with the purpose of cleaning, refining, and convert them to vector (Ng and Hoffman, 2018), some of them are:

- Color equalization: It is accomplished by effectively spreading out the most frequent intensity values, i.e., stretching out the intensity range of the image.

- Morphological operations: They are of two types applied one after the other, erosion followed by dilation. Erosion removes speckle noise ("islands"), but it also shrinks objects. Dilation re-expands the objects.

- Fill in holes: The converse of the previous step, removing "lakes" (small false negatives that are topologically inconvenient) in the mask.

- Contouring: Continuous pixels having same color or intensity along the boundary of the ask are joined. The output is a binary mask with contours.

- Vectorization: It is the transformation of data representation in which objects are converted from in-tile pixel coordinates (masks) to GeoJSONs or Shapefiles formats in geographic coordinates (latitude and longitude).

3.3.2 Post-processing of Vector Objects

For cleaning vector objects after vectorization, another series of tools can be applied one after another (Ng & Hofmann, 2018; Sahu & Ohri, 2019; Touya et al., 2019), they are:

- Simplification or generalization: Douglas-Peucker algorithm is applied which takes line segments and gives a similar curve with fewer vertexes.
- Merging: This tool combines vector polygons or lines that are nearly overlapping, such as those that represent a single feature broken by tile boundaries, into a single polygon or line.
- Centerline: Roads, rivers and other objects are often represented by complex polygons. Since one of the most important attributes of a linear object is its length, extracting this attribute and linear geometry from a polygon is very useful.

Post-processing becomes cumbersome in this way and results are achieved by hand engineering (Filin et al., 2018). Figure 3.3. shows some examples of afore mentioned steps (Ng and Hoffman, 2018).

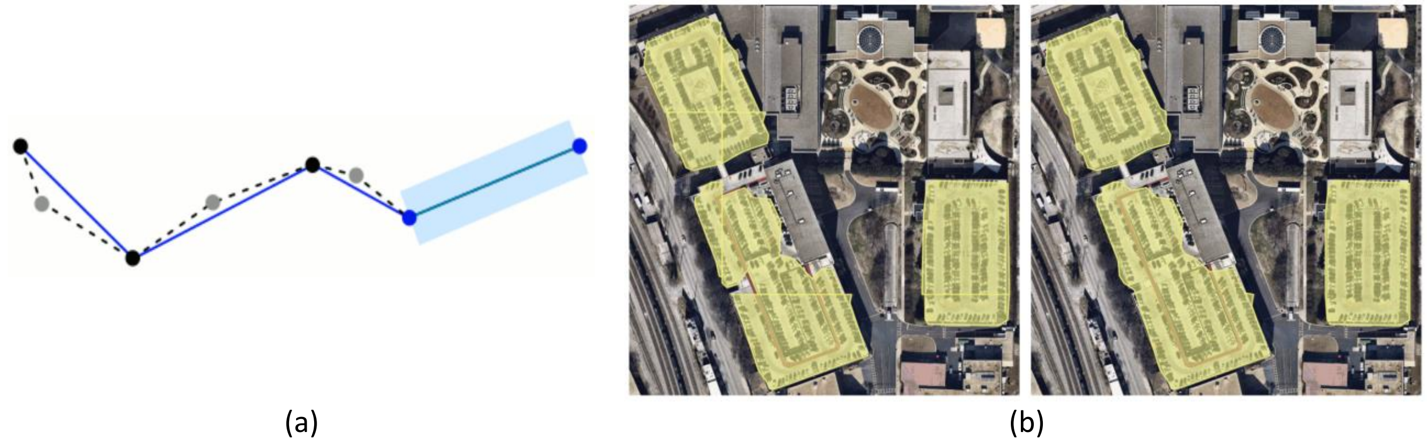


Figure 3.3. Post-processing steps. (a) Douglas-Peucker simplification of a line, (b) Merging of building polygons.

Compared to hand engineering, our post-processing method (chapter 6) uses a deep generative model and look to solve discontinuity (false negative pixels) and irregularity (false positive pixels) of resultant masks, caused by the original generative model. Obtaining better quality masks in this regard, for point, line, and polygon objects in orthomosaics, may lead to a better vector representation.

Chapter 4

4 A Method for Producing Paired Datasets

The first method has the objective of producing datasets with highly discriminative information for training a generative model to create masks for point, line, or polygon from orthorectified images. Next, the workflow of the method is described as well as datasets examples obtained by experimentation using this method.

4.1 A Workflow for Producing Paired Datasets

To create a paired dataset, vector ground truth data and orthomosaics-DSM are initially processed via Python scripts in two separate process lines. Figure 4.1 shows the different steps of the method for producing paired datasets.

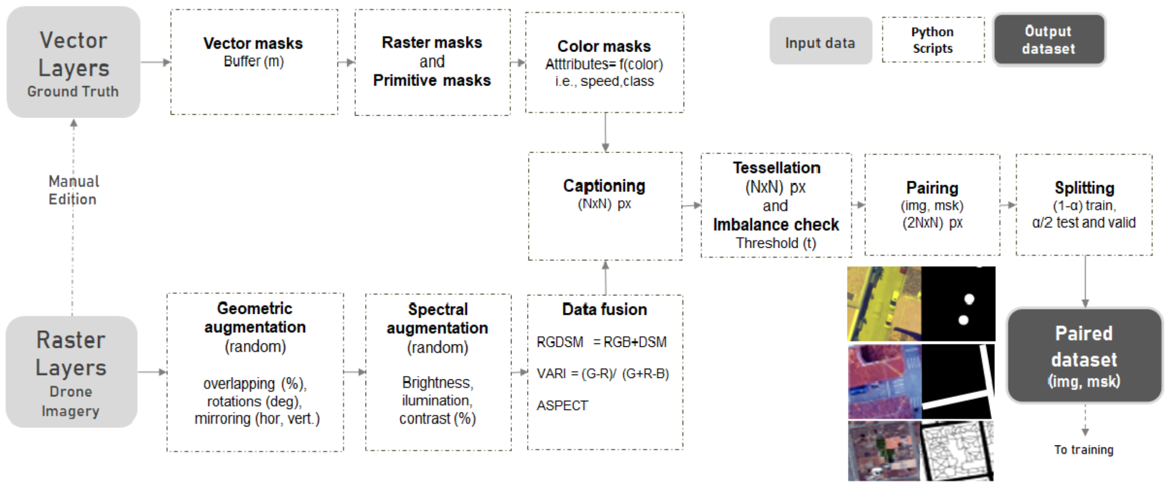


Figure 4.1. Method for producing paired datasets.

4.1.1 Raster Layers: Aerial Drone Imagery, DSM, DTM

Aerial imagery, especially satellite and drone orthomosaics, is becoming ubiquitous. This is due to ease of use and the affordable price of consumer and professional drones, and also to the rapidly increasing availability, quality and price of satellite imagery (Avola & Pannone, 2021; Weir et al., 2019). Orthomosaics are created by stitching images that partially overlap, using a method called Structure from Motion (SfM) (Kameyama & Sugiura, 2021). Drone orthomosaics have a very high spatial resolution, measured by the Ground Sample Distance

(GSD) (Heffels & Vanschoren, 2020; Shermeyer & Etten, 2019; Weir et al., 2019), which is the physical pixel size; a 10 cm GSD means that each pixel in the image has a spatial extent of 10 cm. The GSD of an orthomosaic depends on the altitude of the flight above ground level (AGL) and the camera sensor. Drone image data is acquired by executing several autonomous flights, using a commercial drone and a controlling App, for example: Dji Mavic 3Pro, Dji Phantom 4Pro V2 or other with the Capture App (*Professional Photogrammetry and Drone Mapping Software*/ www.pix4d.com). Raw drone photographs are commonly obtained at heights between 50 and 250 m AGL, depending on the GSD required for the specific application and local flight regulation by the authority (e.g., FAA). Mapping areas are covered with flight lines using a frontal overlap of 80-85% and a lateral overlap of 70-75%. An orthomosaic to cover one-hectare areal extent is obtained in around one minute of flight at 100 meters AGL. Individual images and GPS Log of the flights are processed in a photogrammetric software to obtain default photogrammetric products which are an orthomosaic, a DSM, and a 3D point cloud of a mapping area. We employed Open Drone Map (www.opendronemap.org), an open-source software, to obtain mentioned products when processing raw drone images (Ballesteros et al., 2021). The WGS1984 is the common Geographical Coordinate System (GCS) used to geo-reference the products. Figure 4.2 illustrates the workflow for the acquisition and processing of individual raw drone images.

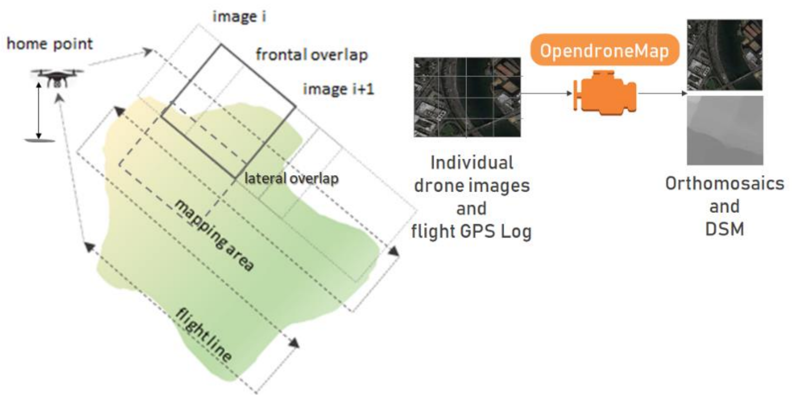


Figure 4.2. Drone imagery data acquisition and processing. Individual drone images and GPS Log are processed in Open Drone Map software to obtain drone orthomosaics and DSM.

Although the method is intended to be applicable to any aerial overhead imagery. We use drone imagery that was acquired over fifteen small to medium size urban areas in Colombia, South America. Figure 4.3. shows an example of the acquired drone imagery. A concise list of the drone acquired imagery and its metadata is presented in table 4.1.

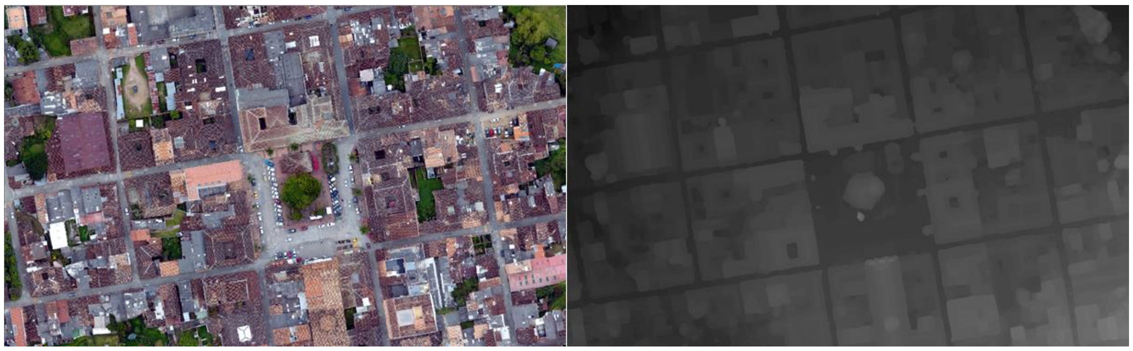


Figure 4.3. Drone orthomosaic and DSM [1] Aerial Raster Layers. This example corresponds to the urban area of El Retiro, Colombia, with a GSD of 7.09 cm/px.

Table 4.1. Acquired Drone Imagery

Settlement	Geographic Extension (Xmin,Ymin,Xmax,Ymax)	Flight Height (m)	GSD (cm/px)	Area (Hectares)
El Retiro, Ant.	-75,5057858485094 6,05456672000301 -75,4995986448169 6,06544416605448	120	7	82.9
La Ceja, Ant.	-75,4379001836735 6,03130980894862 -75,4332962779884 6,0342695019348	80	5.5	16.8
Prado_largo, Ant.	-75,5311888383421 6,15636546472326 -75,5226877620765 6,16018600622437	90	5.7	40
Rionegro, Ant.	-75,3809074659528 6,13947401033623 -75,3760197352806 6,14988050247727	80	5.5	62.7

4.1.2 Geometric Data Augmentation

Data augmentation improves performance of deep learning models (Buslaev et al., 2020), it helps model generalization (Blaga & Nedevschi, 2020; Y. Long et al., 2021; Song & Kim, 2020; Weir et al., 2019), increasing the number of available examples to train a model. However, there are not many studies on the influence of data augmentation on geographic data, and specifically, which of the augmentation methods is the best. Geometric augmentation consists of transformations in scale, angle, and form of images. These variations depend on the field of application and particularly, on the requirements imposed to a model. For instance, ninety degrees mirroring may not be applicable to common objects like dogs or bikes, but they are applicable to overhead imagery. Most important geometric augmentation methods for geographic objects are (Buslaev et al., 2020):

- **Rotation:** Consists of small clockwise rotations of images, suggested value is 10 degrees (Blaga & Nedevschi, 2020).

- **Mirroring:** It is a transformation in which, upper and lower, or right and left, parts of images interchange position. They are commonly referred as vertical and horizontal mirroring.

- **Resizing or Zooming:** It is the magnification of certain parts of an image, zooming in or out.

- **Crop:** It is the trimming of an image at certain place.

- **Deformation:** It is the elastic change of the proportion of image dimensions. It is a common phenomenon that occurs in the borders of orthomosaics.

- **Overlapping:** It is the repetition of a part of an image measured by a percentage (%).

Some of these transformations are implemented in open python libraries (Buslaev et al., 2020).

Geometric augmentation is applied to the Aerial Raster Layers. Overlapping of 10% and 20%, clockwise 10 degree increasing angle rotations, as well as mirroring (90 degrees) are used. Figure 4.4. shows examples of (img,msk) pairs obtained by the application of different types of geometric augmentation. Annex 1 contains our scripts implementation in Jupyter Notebooks for geometric overlapping, rotations, and mirroring.

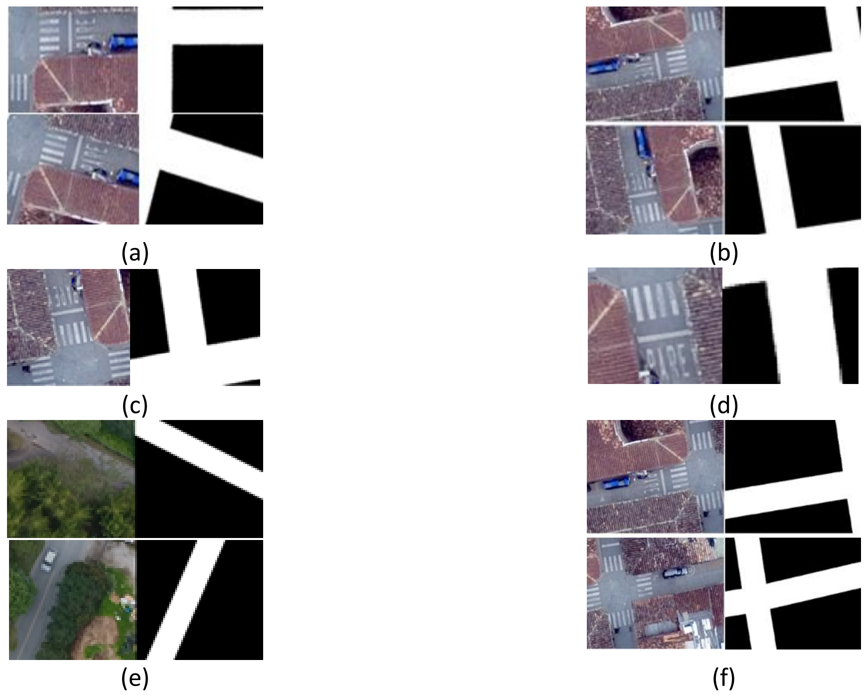


Figure 4.4. Geometric Augmentation. (a) Rotation, (b) Mirroring, (c) Zoom, (d) Cut, (e) Elastic Deformation, (f) Overlapping.

4.1.3 Spectral Data Augmentation

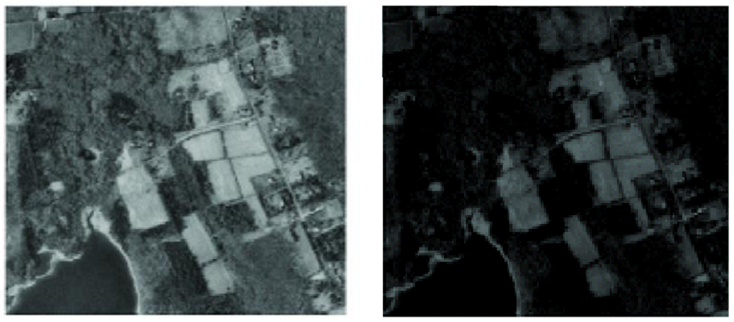
It is the change in brightness, contrast, and intensity (gamma value) of images (Buslaev et al., 2020). They are described as follows:

- **Brightness:** It is the amount of light of an image, it increases the overall lightness of the image—for example, making dark colors lighter, and light colors whiter (*GIS Mapping Software, Location Intelligence & Spatial Analytics | Esri*, www.esri.com).

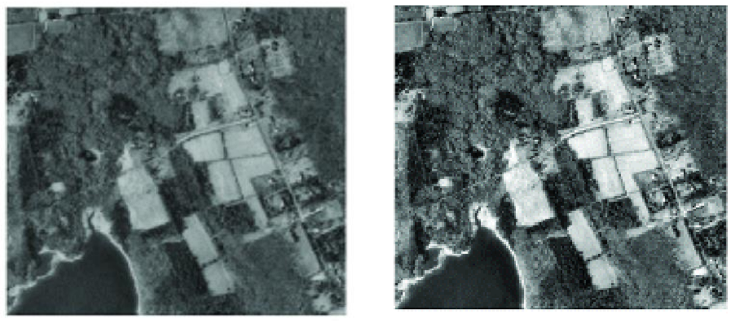
- **Contrast:** It is the difference between the darkest and lightest colors of an image. An adjustment of contrast may result in a crisper image, making image features may become easier to distinguish (*GIS Mapping Software, Location Intelligence & Spatial Analytics | Esri*, www.esri.com).

- **Intensity or Gamma Value:** It refers to the degree of contrast between the mid-level gray values of an image. It does not change the extreme pixel values, the black or white, it only affects the middle values (Buslaev et al., 2020). A gamma correction controls the brightness of an image. Gamma values lower than one decrease the contrast in the darker areas and increase it in the lighter areas. It changes the image without saturating the dark or light areas, and doing this brings out details in lighter features, such as building roofs. On the other hand, gamma values greater than one increase the contrast in darker areas, such as shadows from buildings or trees in roads. They also help bring out details in lower elevation areas when working with elevation data like DSM or DTM. Gamma can modify the brightness, but also the ratios of red to green to blue (GIS Mapping Software, Location Intelligence & Spatial Analytics | Esri, www.esri.com).

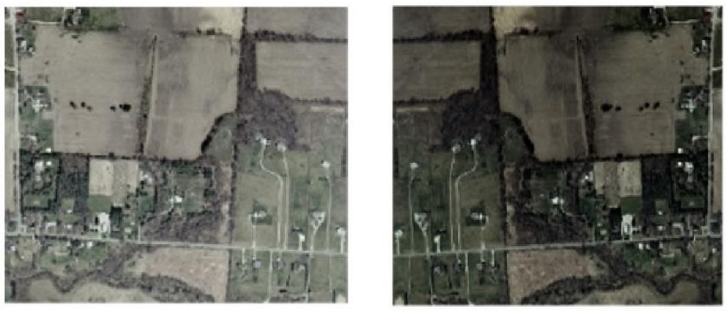
Figure 4.5 shows examples of the resultant imagen when different types of spectral augmentation are applied (Buslaev et al., 2020).



(a)



(b)



(c)

Figure 4.5. Spectral Augmentation. (a) Brightness, (b) Contrast, (c) Intensity (Gamma Value).

Spectral augmentation of Aerial Raster Layers is implemented in open python libraries like PIL (*Pillow*, https://pillow.readthedocs.io). Typically, 10% increment or decrement of current values is applied. Annex 1 contains our implementation in a Jupyter Notebook for increasing and decreasing 10% of default values of brightness, contrast, and intensity.

4.1.4 Data Fusion

Due to computational limitations, most of Deep Learning models for computer vision make use of images with three channels (bands), i.e., RGB images (Xu et al., 2018). Data Fusion is a way for incorporating additional discriminant information to the available channels. Objects height can be a discriminant where exist intricate spatial relations. For instance, the spatial relations between vehicles, roads, trees and buildings are good examples of such a case. Also, there are many popular vegetation indexes developed in remote sensing and mostly used in agricultural monitoring. The well-known Normalized Difference Vegetation Index (NDVI) quantifies the health of vegetation by measuring the difference between bands in a near infrared image (NIR) (Eng et al., 2019). Data fusion is applied to orthomosaics for integrating height (DSM, DTM) or indexes (NDVI, VARI) into a dataset as follows:

- Height: The DSM, which contains the height of objects in an image can be fused with the orthomosaics, by adding it whether algebraically or logarithmically, to each red (R), green (G), and blue (B) bands as stated in Equations 3.1. and 3.2. Other option is replacing any of the bands with DSM, for instance, the blue band, this comes from the idea that blue color is less frequent in objects of nature, RGDSM is calculated in Equation 3.3.

$$HRGB = (R + DSM), (\mathrm{G} + DSM), (\mathrm{B} + DSM) \quad (3.1)$$

$$HLRGB = (R + Log(DSM)), (G + Log(DSM)), (G + Log(DSM)) \quad (3.2)$$

$$RGDSM = (R), (G), (DSM) \quad (3.3)$$

  In any case, the resultant image is a three bands false color composite (López-Tapia et al., 2021) with values ranging between 0 and 255, so values of every band should be re-scaled to that interval. For instance, the DSM is re-scaled from its original values in meters (Zhang et al., 2015). The resultant image can be called NDSM, and it is obtained by Equation 3.4.

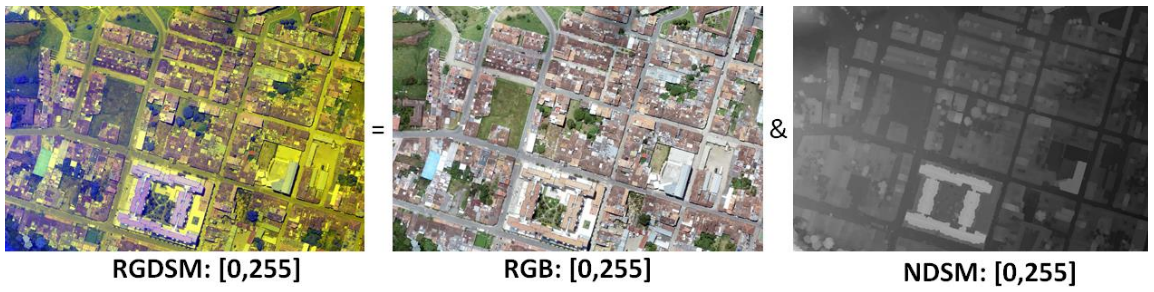
$$NDSM = (DSM - DSMminHeight) * 255/(DSMmaxHeight - DSMminHeight) \quad (3.4)$$

$DSMminHeight$ and $DSMmaxHeight$ are the minimum, and maximum values of the DSM respectively. More datasets are including these days height as a way to improve image understanding, for example, the NYU depth V2, the SUN RGB-D, and the HAGDAVS (Sun & Wang, 2018; Ballesteros et al., 2022).

- Index: It may replace one of the RGB channels of an orthomosaic with the value of an index. The Visible Atmospherically Resistant Index (VARI) was developed by (Gitelson et al., 2002), on a measurement of corn and soybean crops in the Midwestern United States, to estimate the fraction of vegetation in a scene, with low sensitivity to atmospheric effects in the visible portion of the spectrum. It is exactly what occurs in low-altitude drone imagery (Eng et al., 2019). Equation (3.5) allows to calculate the VARI for an orthomosaic using the red, green and blue bands of an image.

$$VARI = (Green - Red) / (Green + Red - Blue) \quad (3.5)$$

VARI Index should also be re-scaled to the RGB orthomosaics values interval [0.,255.]. Equation (3.4) can be used to obtain the NVARI Index. Data fusion is used to replace the green color with the VARI Index. This has the purpose of better and faster discriminate green objects, removing the effects of the atmospheric distortions. Figure 4.6. shows the resultant examples of height augmentation, RG-DSM, and index augmentation, the RVARIDSM false color composite images.



(a)

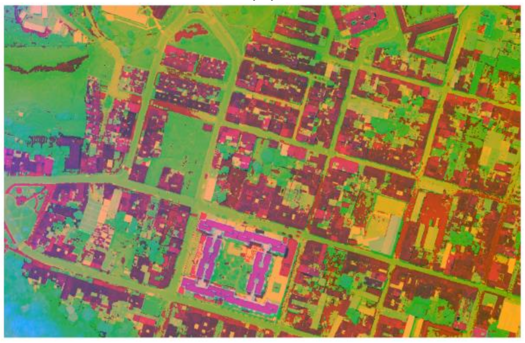

(b)

Figure 4.6. Data Fusion. (a) Height Augmentation: False color composite RG-DSM image obtained by fusion of red and green bands and use of DSM instead of the blue band, The RG-DSM is a bluish-looking image in terms of the height of the objects, the higher the objects in the image the bluish the color, would height aid to segment vehicles from roads, and roads from trees and buildings, even when they exhibit similar spectral responses, (b) Index Augmentation: resultant RVARIDSM false color composite image obtained by data fusion of Red, use of NVARI in the green band to discriminate green objects, and the NDSM in the blue band to discriminate different heights.

4.1.5 Vector Layers (Ground Truth): point, line, and polygon objects

Depending on the location in the world, vector ground truth data is obtained by querying for existing objects like roads and rivers, and less frequently for buildings in the OSM world database. A Python script to query vector objects from OSM database, and their conversion to shapefiles is provided in Annex 1. If inexistent, vector layers should be produced by a manual digitalization on the PC screen using the input orthomosaics and DSM as base layers and tracing point, line, and polygon objects. A description of a manual digitalization of each geometry is as follows:

- **Point:** Point vector objects are those than can be represented as (x, y) coordinates at a geographical extent. They are created by pinpointing them over the orthomosaic at a place that represents object extension and location. Some examples of point vector objects are vehicles, poles, trees, road signs, tanks.

- **Line:** Linear objects are those in which length is a lot larger than width. They are digitalized by adding vertices (x, y) at any change of direction, and have at least two vertices. Some examples of line objects are roads, rivers, and trains.

- **Polygon:** Vector polygons represent region objects; they are digitalized by creating vertices at each change of direction until last vertex coincides with the initial one. Buildings and forests are examples of polygon objects.

4.1.6 Vector Masks, Raster Masks and Color Masks

Point, line, and polygon ground truth vector layers, obtained from OSM or by manual digitalization, are buffered using a distance parameter. The resultant vectors are called vector masks. They are converted to raster (rasterized) to produce a raster mask. A raster mask is an image with the same geographic extension of the input orthomosaic or DSM from which objects are vectorized (typically millions of pixels). Raster masks can be binary (black and white), they represent only one object of interest (positive class) and its background (negative class). The positive class is encoded in white color (class = 1), it competes against the ground, dominant class, encoded as black (class = 0). A color raster mask may be used when extracting object attributes. For instance, road speed, vehicle type, roof material and many others. Producing raster masks semi automatically for vector objects needs an optimum buffer distance. Since it is used to increase the size of point and line vector geometries, it is a "tradeoff" between getting imbalance masks (small buffer distance) with no misclassified pixels, and including a high number of pixels with a certain degree of misclassification (large buffer distance). Since masks for polygon objects are generally not imbalance, their buffer distance is zero (0). Vector masks are rasterized, converted to binary masks, or if attributes are needed, they are rasterized into color masks.

4.1.7 Image Caption and Image Captioning

Image Caption is the automatic description of a given image by a sentence that has sense in a specific language. It has been a fundamental task in the Deep Learning domain and counts with many applications in web search and in the help of blind people. Image caption can be regarded as an end-to-end sequence to sequence problem (www.machinelearningmastery.com), as it converts images, which is regarded as a sequence of pixels to a sequence of words. State of the art Image Caption systems use a combination of a CNN and a Recurrent Neural Network (RNN). The first network classifies and retrieves the characteristics (words) found in the image, the second extracts a correct sequence of words that describes the image (Vinyals et al., 2015). Nonetheless, image caption is not yet widely used in the GIS domain, a proposed application in this field would be to automatically describe overhead images. Image Captioning is the process to assign a list of valid attributes to every image in a dataset. This is the labeling of images with a list of attributes. To train an image caption model, image captioning should be applied to a set of images that contains objects of interest and store them as $(image, [captions])$ list. The Ground Truth Vector Layers with attributes for an object of interest stored in vector files (shapefile or GeoJSON) are inherited by the images when they are both split with the same extension (desired output image size) at the tessellation stage, producing a captioned dataset. For instance, a vehicle may have attributes like type (truck, bus, car, motorcycle, etc.), service (public, private), color (red, green, black, white, etc.), and many others. Image captioning results in specialized datasets for describing point, line, or polygon objects, i.e., vehicles, roads, and buildings.

4.1.8 Image Tessellation, Imbalance Check, Pairing and Dataset Splitting

Due to computational restrictions, it is common to train deep learning models with square 256x256 px images. In this respect, orthomosaics and raster masks (binary or color) are huge, thus they should be tessellated at a desired size N, producing (NxN pixels) image chips, for example, 256x256 pixels or other. Since many geographical objects are scarce respect to the ground, they produce an im-balance mask. Class imbalance is a common problem that affects performance of deep learning models, moving the decision boundary towards the dominant class (S. Wang et al., 2016). The imbalance ratio of the positive class can be calculated for a specific dataset with n images as in Equation 3.6.

$$Imbalance\ ratio\ of\ positive\ class = \sum_{i=1}^{n} \frac{pixels\ of\ positive\ class}{pixels\ of\ positive\ class + pixels\ of\ negative\ class} \quad (3.6)$$

Values of around 0.5 in the Equation 3.5. correspond to a pixel balanced mask, values below 0.001 are an extremely imbalance mask. Every image–mask chips corresponding to a balance mask are saved as a whole image of ($2NxN$ pixels), for instance 512x256 pixels. Imbalance ratio is checked on every mask using a threshold t, instead of calculating it on the whole raster mask. Finally, a random splitting of the dataset is done using a proportion value α. An initial partition into $(1-\alpha)$ for training, and α for validation-testing is performed. This last dataset is again random split into $\alpha/2$ to produce separate validation and testing datasets.

Different sizes can be used for tessellations, starting bottom up from 256x256 pixels, to 512x512, and ending up at 1024x1024 pixel images, or the other way around. Images and corresponding masks are then paired into (img,msk) with sizes of 512x256, 1024x512, and 2048x1024 pixels. Since imbalance can affect the performance of the generative model, every (img,msk) pair should be checked to pass an imbalance ratio threshold t, for example: 10%, 20%. This is performed in the imbalance check step. Figure 4.7. shows an example of paired (img,msk) for point (vehicles), line (roads), and polygon (buildings) objects in the drone orthomosaics.

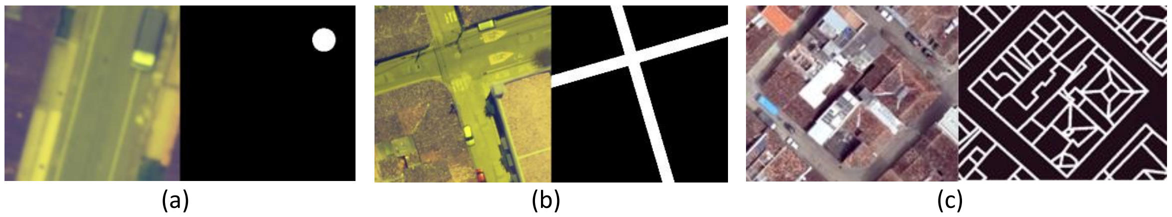


Figure 4.7. Paired (img, msk) for point-line-polygon in orthomosaics. (a) mask for point objects, ex. vehicles, (b) mask for line objects, ex. Roads, (c) mask for polygon objects, ex. Buildings.

4.2 Experiments and Results

To challenge the proposed method to produce geographic paired datasets, we created one dataset from acquired drone imagery per each geometry. Vehicles, roads, and buildings are excellent examples of common objects of interest that are represented in those geometries.

4.2.1 Point Objects Masks: A Vehicle Paired Dataset Example

One of a good examples of point vector layers in GIS are vehicles. Semantic Segmentation of vehicles is a widely studied area in computer vision and Artificial Intelligence, driverless cars are good evidence of that. A few research studies have proposed datasets for vehicle segmentation in satellite or drone imagery (Bisio et al., 2021; Blaga & Nedevschi, 2020). They constitute the base for training segmentation models, where pixel-level classification is dominantly performed by the U-Net architecture (Abdollahi et al., 2021; Pashaei et al., 2020; H. L. Yang et al., 2018). Figure 4.8. shows training image and segmentation masks in the Potsdam satellite dataset (Song & Kim, 2020) and in the HAGDAVS drone dataset (Ballesteros et al., 2022).

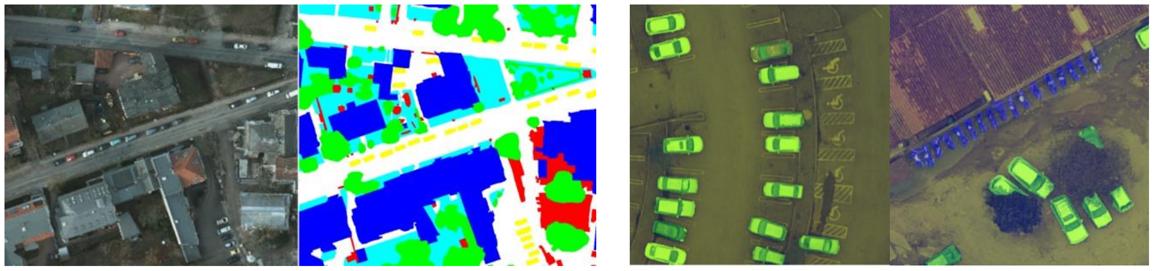

(a) (b)

Figure 4.8. Segmentation datasets in Satellite and Drone imagery. (a) Potsdam Dataset, (b) Hagdavs Dataset.

Vehicle geolocation in drone aerial orthomosaics is also becoming particularly important due to its application in different fields like security, traffic and parking management, urban planning, logistics, and transportation, among many others. However, different sizes or masks shapes for vehicles limit to accurately geo-locate vehicles position in a GIS (Fan et al., 2016). Also, vectorized masks with different shapes are difficult or impossible to handle in a GIS vector layer (Li et al., 2021). The "Domino Dataset" for vehicle geolocation is a paired dataset obtained by the proposed method. It is semi automatically labeled; vehicles are pinpointed around its central position, over the roof, using orthomosaics as the base layers. This point shapefile (*.shp) is created in the QGIS open-source software (www.qgis.org). Some attributes can be added to the point vector layer, like: type of vehicle as an integer value, i.e., 1:car, 2:motorcycle, 3:bus. After this, a buffer vector layer is obtained using 1 meter (radius value). The buffer is rasterized afterwards, producing an 8bit raster binary mask in .tiff image format. Circle buffer raster layer is made of white pixels that represent vehicles, and the black pixels form the background (negative class). It has the same geo-reference and extension of the base orthomosaic. Values of 25% above and below normal contrast and brightness were used as spectral augmentation of orthomosaics. Images and corresponding masks were rotated by 10-degree increments. Height augmentation is included, placing the DSM into the blue channel, height would make possible to segment vehicles a lot easier than when that information is not present (Ballesteros et al., 2022). As stated before, the buffer distance is relevant because a larger value impedes to segment neighbor vehicles, meanwhile a small radius creates a more imbalance positive class. Datasets using different radius values can be created and used in the model to obtain the optimum value. Figure 4.9. illustrates the Domino dataset obtention process. Table 4.2. describes the number of images of the dataset and link to download it.

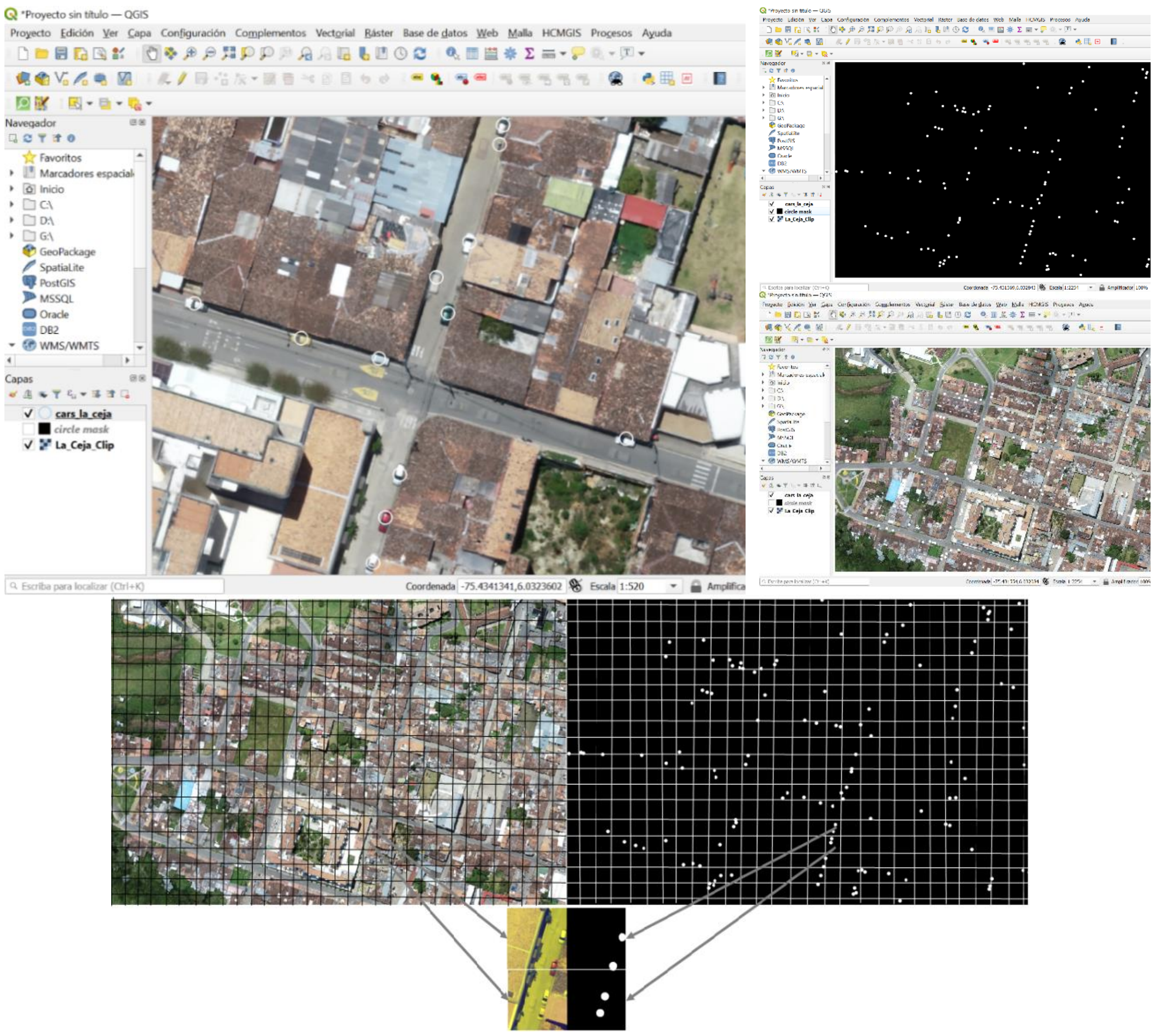

*Proyecto sin título — QGIS
Proyecto Edición Ver Capa Configuración Complementos Vectorial Ráster Base de datos Web Malla HCMGIS Procesos Ayuda
Navegador
Favoritos
Marcadores espaciales
Inicio
C:\
D:\
G:\
GeoPackage
SpatiaLite
PostGIS
MSSQL
Oracle
DB2
WMS/WMTS
Capas
cars_la_ceja
circle mask
La_Ceja_Clip
Escriba para localizar (Ctrl+K)
Coordenada -75.4341341,6.0323602
Escala 1:520
Amplifica

Figure 4.9. Domino dataset for vehicle geolocation.

4.2.2 Line Objects Masks: A Road Paired Dataset Example

Roads are one of the best examples of linear features studied in GIS. Intelligent transportation systems generally require fast processing of road conditions, and it has numerous potential applications such as: driver assistance, transportation scheduling, and route optimization (Batra et al., 2019). Road infrastructure evaluation is a demanding task, it is performed in terms of road quality, type, and total length (Brooks, 2017; Pinto et al., 2020). To this end, computer vision is becoming ubiquitous through the use of semantic segmentation using satellite imagery with full size road masks datasets (Gerke et al., 2014; Song & Kim, 2020). However, road segmentation is challenging due to the different road types, and various background, weather, and illumination conditions (Y. Long et al., 2021; Van Etten, 2019; Van Etten et al., 2019), but also for little or no attributed data availability. Drones are being explored by many Departments of Transportation (DOTs) for innovative applications, as an emerging and cost-effective solution for road asset inspections (Aldana Rodriguez et al., 2021), and mapping (Ballesteros et al., 2021). Compared to ground images, obtained by vehicle mounted cameras and sensors, drones can travel at very high speeds and cover large areas in a short amount of time. With the aid of drone imagery new insights are being developed thanks to its high resolution, but, at the same time, computer vision algorithms should be more robust due to the gathering of new information contained at centimeter-level spatial resolution (Avola & Pannone, 2021). Having as many labels for roads as possible is desirable, for instance, labels for road speed, road surface and its quality, road class, and type, are the support for the development of better supervised deep learning algorithms applied in road assessment. Figure 4.10. shows examples of satellite and aerial datasets for road segmentation.

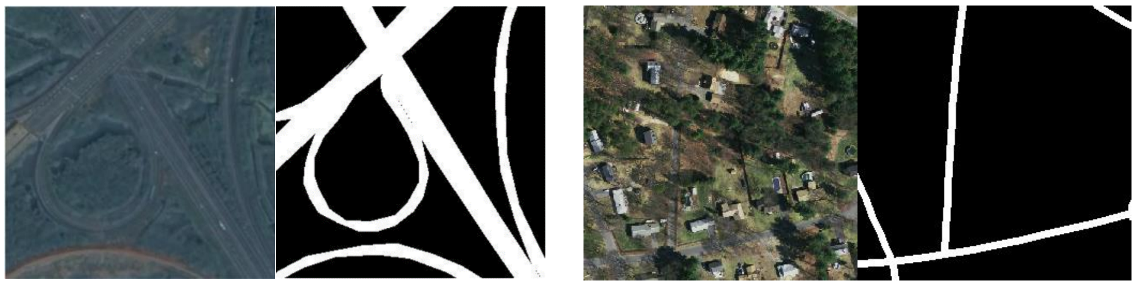


Figure 4.10. Road segmentation datasets. (a) RSI: Satellite, full size dataset (b) Road Massachusetts Dataset: Aerial, no attribution, equal size, binary dataset.

The Type Surface Quality and Speed Road Dataset (TSQS Road Dataset), is created by our proposed method for paired datasets. It consists of five drone orthomosaics and corresponding raster multi class color masks of roads, making 16 km the total length of different road types in a developing country. The orthomosaics are acquired at various locations, flight heights (80 to 200 m), and spatial resolution (GSD between 5 and 10 cm/px). Classes and speed are created semi-automatically using python scripts (Annex 1), and quality-surface manually labeled by civil engineering students. Roads are coded in a sequence manner (color:class_name:class_number), with the following attributes: road type (red:paved:1, green:unpaved:2), road surface (red:concrete:1, green:asphalt:2, blue:grava:3, white:earth:4), quality of the surface (red:good:1, green:regular:2, blue:bad:3), road speed (red:5km/h:1, green:20km/h:2, blue:40km/h:3, white:60km/h:4, gray:80km/h:5), and road class (red:residential:1, green:highway:2, blue:secondary:3, white:tertiary:4, gray:path:5, yellow:pedestrian:6). Roads are buffered using 1-, 2-, or 3-meters buffer distance and represented with different color. Background pixels correspond to negative class (black:background:0) for all the cases. Table 4.1. summarizes name of location, extension in WGS84 geographic coordinate system, GSD, flight height and area of the orthomosaic. Figure 4.11. shows examples of the different segmentation road masks by attributes.

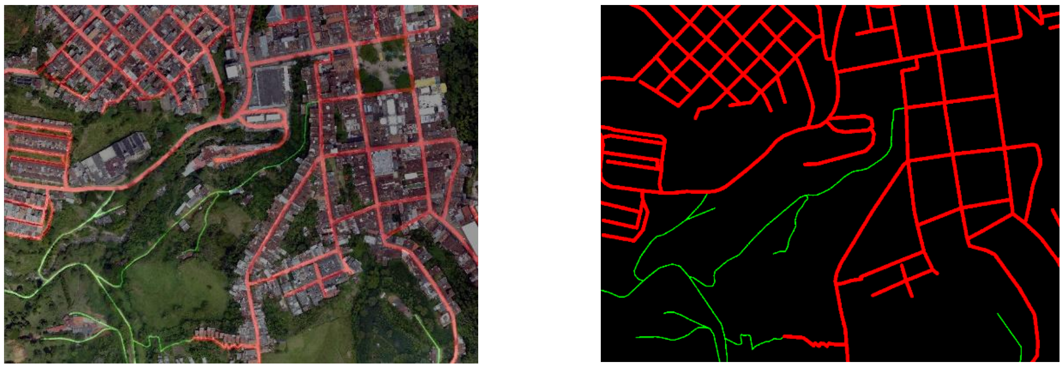

(a) (b)

Figure 4.11. TSQS Road Dataset. (a) Orthomosaic-mask, (b) mask using attribute type of road: red color represents paved roads, green represents unpaved, and black the background class.

4.2.3 Polygon Objects Masks: A Building Paired Dataset Example

In GIS, buildings are represented as polygons. Accurate building polygons provide essential data for a wide range of urban applications such as construction, solar energy allocation, and environmental studies in heat island impacts (Murtiyoso et al., 2020). There are many datasets that have served as the base for the application of deep learning models to extract pixel-based building areas from remote sensing imagery. Some examples of those are: the Massachusetts Building Dataset acquired by plane, the satellite-based Wuhan University building dataset (WHU), and the ISPRS Vaihingen dataset (Gerke et al., 2014). These datasets have annotated the building footprint in vector format and converted to binary ground truth masks, most of their examples have been collected in developed countries, where there are building separation and uniformity of materials. Figure 4.12 illustrates the Massachusetts and WHU dataset (Mnih & Hinton, 2010).

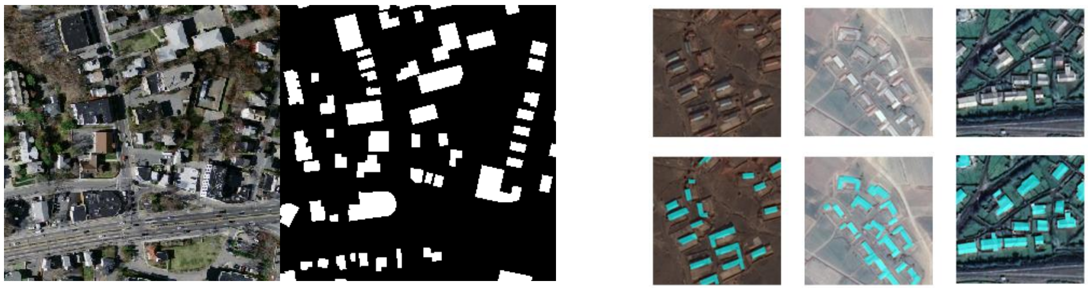

(a) (b)

Figure 4.12. Building footprint datasets. (a) Massachusetts dataset, (b) WHU Building dataset.

Our proposed dataset focuses on building roof structure rather than the footprint, this is the differentiation of all the parts of a building roof. Roof structure also allows to extract and incorporate additional information for geospatial analysis like orientation (respect to the sun), type of material (metal, bitumen-asphalt, pvc, tar and gravel, acrylic, wood, clay tile, solar panel, concrete, green backyard, buildup backyard), number of floors, and runoff direction. Ground truth vector polygons are obtained by manually dissecting buildings from the street blocks. This is labeled over acquired drone imagery using binary and colored masks. Figure 4.13. illustrates the proposed roof information dataset, also called the RIF dataset. Table 4.2. describes the number of images of the dataset and link to download it.

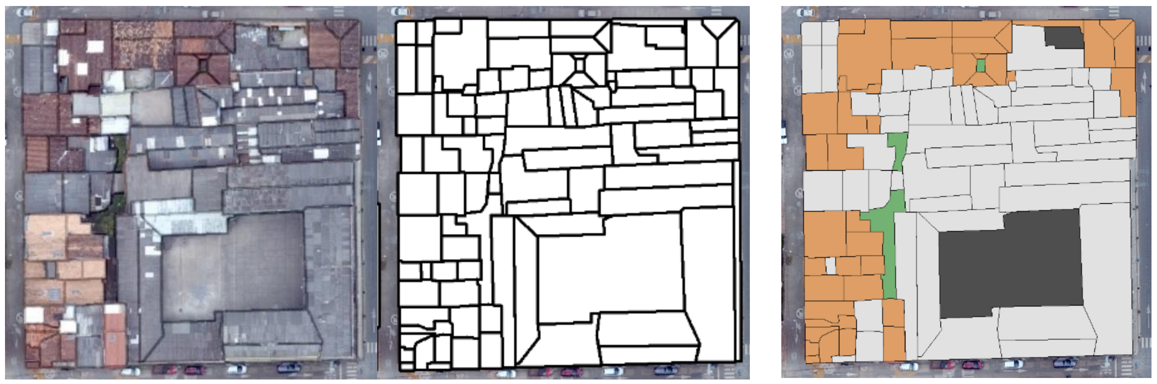


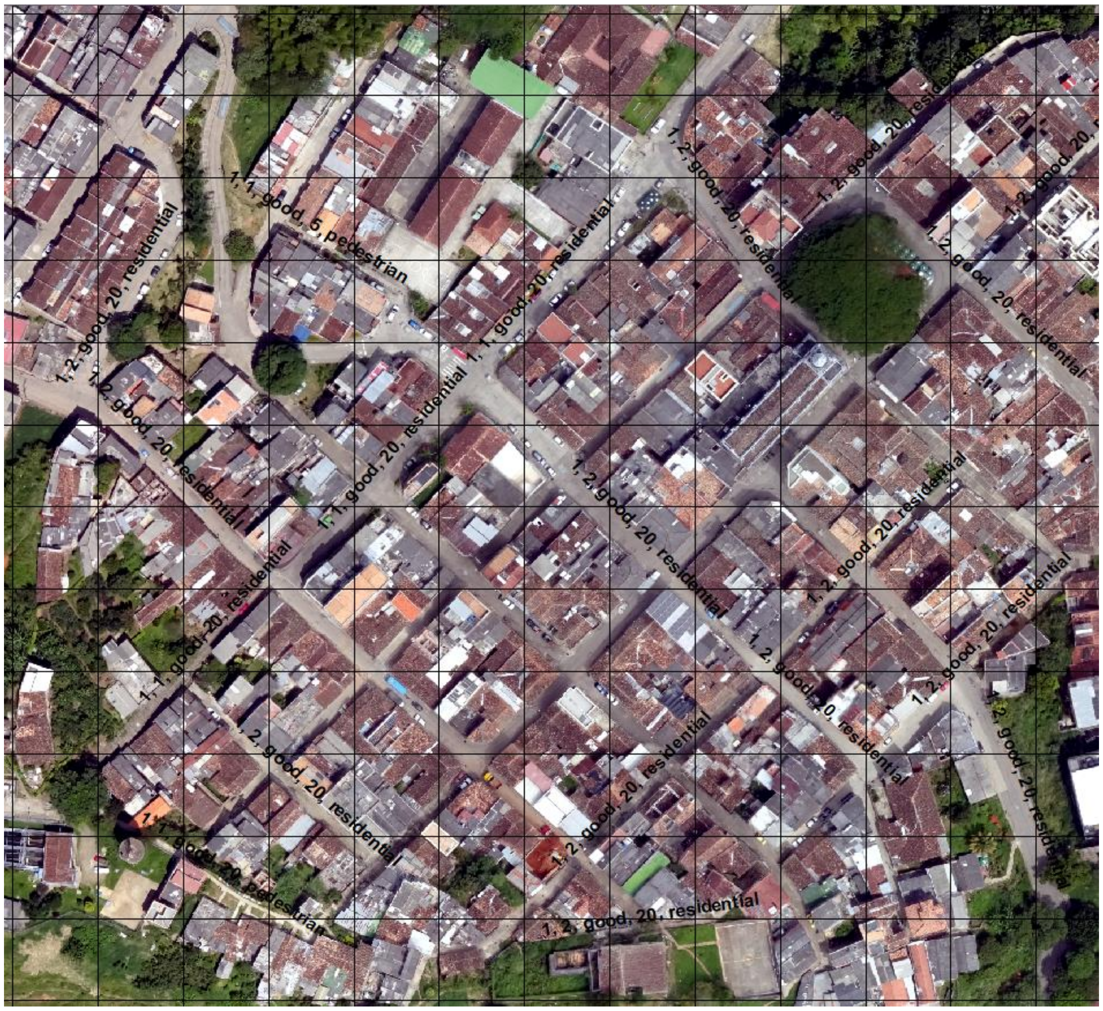
Figure 4.13. RIF dataset. (a) Image-binary mask, (b) Roof material mask, 4 types of roofs are found in the depicted block.

4.2.4 Image Caption Dataset for Roads from Drone Imagery

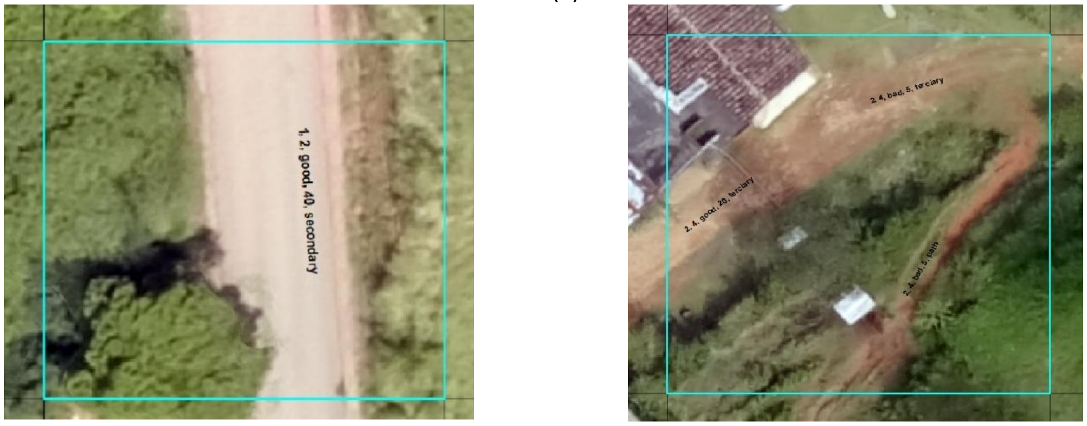
Our image caption dataset uses the TSQS Road Dataset vector layer GT to cut corresponding orthomosaics, obtaining in this way, a set of images of roads that are captioned with type, surface, quality, and road speed attributes. If the Vector GT has attributes originally, or they are annotated manually, our method can create semi automatically a specialized caption road dataset at a desired image size. Annex 1 contains a script developed for image captioning, where a square and uniform size net is created and used to clip GT Vector Layers as well as Raster Layers. The net has a parametric size to create different size images, e.g., 256x256, 512x512 pixels and so on. The manual association of the images with corresponding attributes could be cumbersome, in the script the corresponding list of attributes are assigned to each clipped image, obtaining the set of pairs image - attribute list (img, [captions]). Figure 4.14. shows the image caption dataset for roads. Image captions can be retrieved for an input image using a classifier architecture. Figure 4.15. shows a proposed classifier architecture for obtaining image caption of our proposed datasets. Table 4.2. describes the number of images for the described datasets, and the link to download it.

Table 4.2. Paired Datasets Created by Proposed Method

Name of the dataset	No. of images	Link
Domino Dataset	1000	https://doi.org/10.5281/zenodo.5718809
TSQS Road Dataset	2500	https://zenodo.org/deposit/6878854
Caption Dataset	1220	https://zenodo.org/deposit/6949339
RIF dataset Road Drone	850	https://zenodo.org/deposit/6950521


1, 1, good, 5, pedestrian
1, 2, good, 20, residential
1, 1, good, 20, residential
1, 2, good, 20, residential
1, 1, good, 20, pedestrian

(a)



(b)

Figure 4.14. Image caption dataset for roads in drone imagery. (a) Uniform squared net to clip, (b) Examples of captioned image chips of 256x256 pixels.

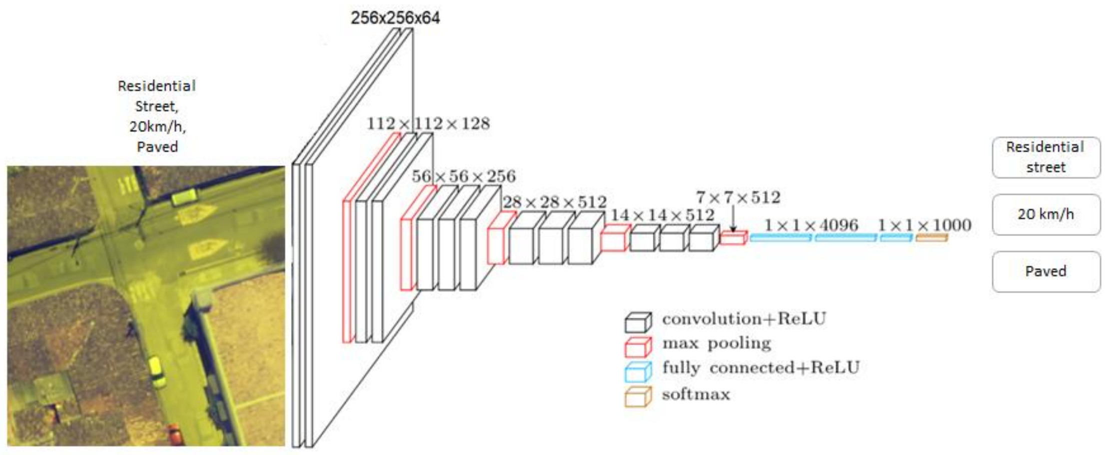


Figure 4.15. A classifier architecture for image caption of proposed datasets.

4.3 Conclusions and Future Work

Proposed method to create paired datasets is of general application to satellite, aerial, or drone acquired imagery. It consists of two streams, the first one allows to acquire satellite information, and the vector ground-truth from querying the world's biggest open-source GIS database. The other stream brings user information in the form of raster, for example, drone imagery consistent of orthomosaics, DSM, and DTM. Vector stream is easy to couple with raster, and the use of the customizable buffer distance parameter can obtain balanced datasets. The level of imbalance can be checked for every image-mask pair. It is also applicable to any object geometry and to the production of binary or multi-class color masks for attribute extraction. The novel inclusion of height values, as well as the VARI Index, in the channels of RGB images help to discriminate tree occlusions and spectral similarities. Our method includes the possibility of creating geographic image caption datasets. These are scarce in the geographic realm and constitute the base for building specialized web search services.

Future work is suggested on the development of a pipeline for automatically labeling of geographic data. Creating a benchmark of obtained datasets by the proposed method would aid in the improvement of models for geographic objects extraction.

Chapter 5

5 A Generative Method for Objects Extraction from Orthomosaics

Images to masks translation using a deep generative model can be performed conditionally or unpaired on the training dataset (Isola et al., 2017; Zhu et al., 2017). The differences between them are the use of a simpler and smaller model architecture that requires less examples of aligned image-mask pairs in the former case. Or in the second case, employing a more complex architecture based on the cycle consistency, that needs a larger number of unpaired examples to learn the translation between the two latent spaces. Both approaches are evaluated next, and the results are compared to choose the most appropriate one for quality mask generation with less manual engineering.

5.1 Conditional and Unpaired Image to Image Translation

The description of Pix2Pix, and Cycle GAN implementations, and the inclusion of transformers and U-Net is presented next.

5.1.1 Conditional Image to Image Translation using Pix2Pix

Pix2Pix model is used over the same dataset, during 100 epochs and with a dropout of 50%. This model is complemented including attention modules. The Pix2Pix architecture was implemented in Keras for Tensorflow library, see code in Annex 1. The Generator uses a U-Net architecture, it takes source images (orthomosaics chips) and outputs target images (masks) of the given examples of paired datasets. The PatchGAN of 70x70 pixels discriminator is provided with an input image (a drone image chip) and a real (training example mask) or a generated mask, and it must determine whether it is real or generated by the network. Figure 5.1. illustrates the architecture of the PatchGAN discriminator, and Figure 5.2. shows the architecture for the generator model.

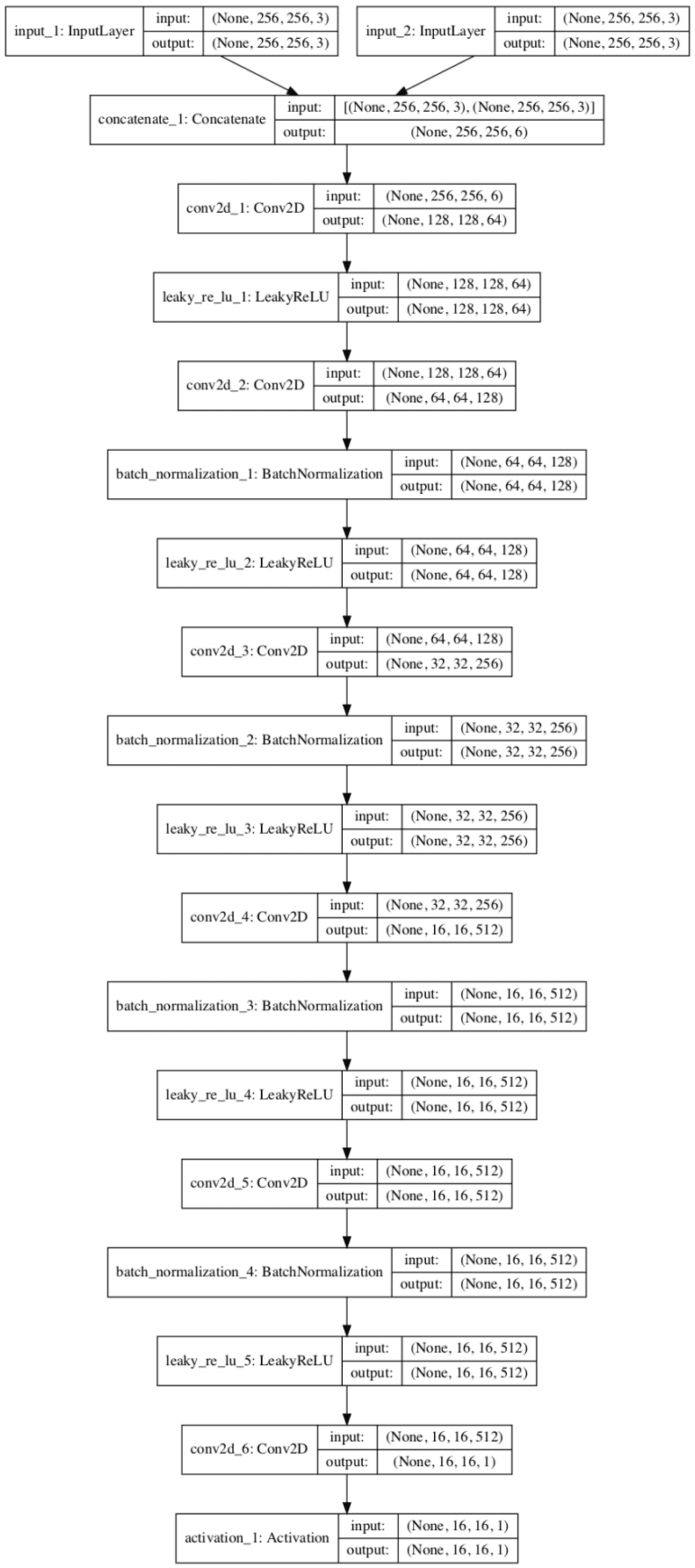

input_1: InputLayer | input: (None, 256, 256, 3) | output: (None, 256, 256, 3)
input_2: InputLayer | input: (None, 256, 256, 3) | output: (None, 256, 256, 3)
concatenate_1: Concatenate | input: [(None, 256, 256, 3), (None, 256, 256, 3)] | output: (None, 256, 256, 6)
conv2d_1: Conv2D | input: (None, 256, 256, 6) | output: (None, 128, 128, 64)
leaky_re_lu_1: LeakyReLU | input: (None, 128, 128, 64) | output: (None, 128, 128, 64)
conv2d_2: Conv2D | input: (None, 128, 128, 64) | output: (None, 64, 64, 128)
batch_normalization_1: BatchNormalization | input: (None, 64, 64, 128) | output: (None, 64, 64, 128)
leaky_re_lu_2: LeakyReLU | input: (None, 64, 64, 128) | output: (None, 64, 64, 128)
conv2d_3: Conv2D | input: (None, 64, 64, 128) | output: (None, 32, 32, 256)
batch_normalization_2: BatchNormalization | input: (None, 32, 32, 256) | output: (None, 32, 32, 256)
leaky_re_lu_3: LeakyReLU | input: (None, 32, 32, 256) | output: (None, 32, 32, 256)
conv2d_4: Conv2D | input: (None, 32, 32, 256) | output: (None, 16, 16, 512)
batch_normalization_3: BatchNormalization | input: (None, 16, 16, 512) | output: (None, 16, 16, 512)
leaky_re_lu_4: LeakyReLU | input: (None, 16, 16, 512) | output: (None, 16, 16, 512)
conv2d_5: Conv2D | input: (None, 16, 16, 512) | output: (None, 16, 16, 512)
batch_normalization_4: BatchNormalization | input: (None, 16, 16, 512) | output: (None, 16, 16, 512)
leaky_re_lu_5: LeakyReLU | input: (None, 16, 16, 512) | output: (None, 16, 16, 512)
conv2d_6: Conv2D | input: (None, 16, 16, 512) | output: (None, 16, 16, 1)
activation_1: Activation | input: (None, 16, 16, 1) | output: (None, 16, 16, 1)

Figure 5.1. Implementation of the PatchGAN architecture in Keras (The CNN-patch discriminator).

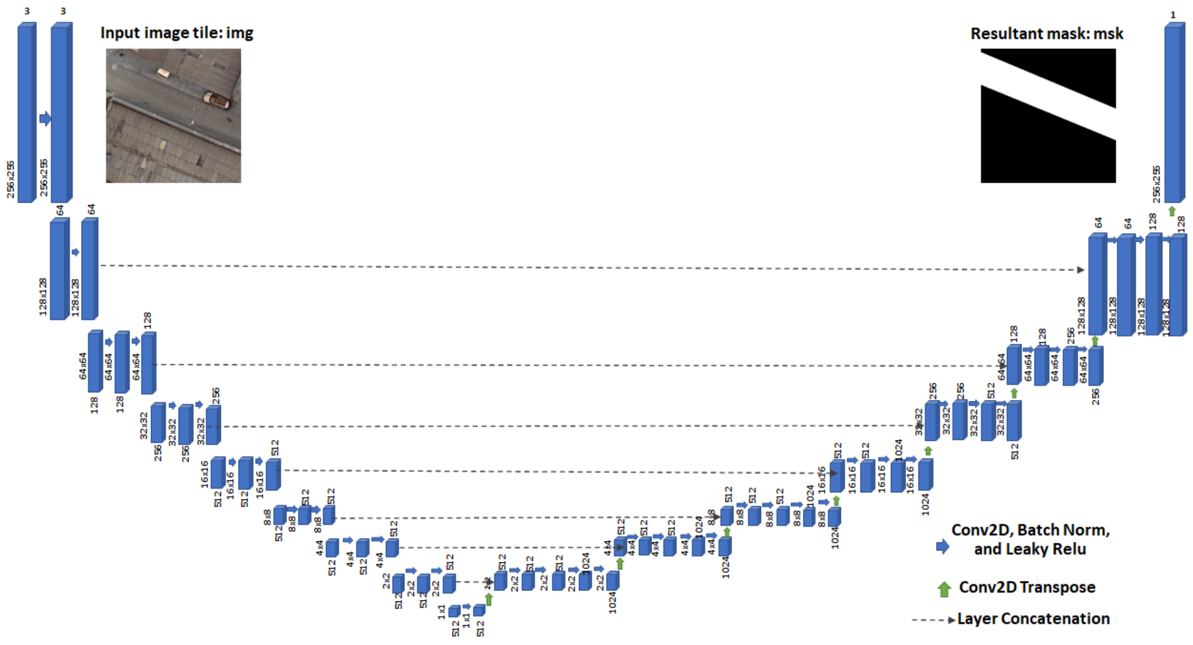


Figure 5.2. Architecture of the generator model based on U-Net. Modified from (Ronnenberg et al, 2015).

Equation 2.2 of section 2.1 can be re-written as Equation (5.1) (Isola et al., 2017).

$$Generator\_Loss = Ex,y[LogD(x,y)] + Ex,z[Log(1 - D(x,y'))] + \lambda\, Ex,y,z[||y - y'||] \quad (5.1)$$

Equation 3.1 is the sum of the discriminator loss on the real masks examples (d1_loss), the discriminator loss on the generated masks (d2_loss), and the differences between pixels in the generated mask and the ground truth mask (L1_Loss). An easier way to re-write the Equation 3.1 is as Equation 5.2.

$$Generator\_Loss = d1\_loss + d2\_loss + L1\_Loss \quad (5.2)$$

In general, Pix2Pix conditional translation model achieves good learning results when $d2\_loss$ is smaller than $d1\_loss$ and the value of $Generator\_Loss$ is small (less than one) even with λ equal to 100 as stated in the original paper (Isola et al., 2017).

5.1.2 Unpaired Image to Image Translation using CycleGAN

Training a generative model for image-to-image translation normally requires a dataset of paired examples. These kinds of datasets are typically created by an expensive manual procedure which is limiting. A way to perform image translation without the need of paired datasets is The CycleGAN. It is trained in an unsupervised way using a collection of images from the source and target domains that do not need to be coupled. So, the model extracts the characteristics of both latent spaces and use them to make the translation (Zhu et al, 2107). The Cycle GAN involves the simultaneous training of two generators and two discriminator models. One generator takes images from the first domain and outputs images for the second domain, and the other generator takes images from the second domain and generates images from the first domain. Discriminator models are used to determine how plausible the generated images are and update the generator models accordingly. This might be enough to generate plausible images in each domain but cannot guarantee that the learned function can map an individual input xi to a desired output yi. To ensure that, Cycle GAN uses cycle consistency, which is that an image output by the first generator could be used as input to the second generator and the output of the second generator should match the original image. The reverse is also true. This is achieved by adding an additional loss to measure the difference between the generated output of the second generator and the original image, and the reverse. The Generator and discriminator use the same configuration as the Pix2Pix model (Isola et al, 2017). The discriminator architecture is: C64-C128-C256-C512m, referred to as a 3-layer PatchGAN, where C means a convolution block and m a max pooling layer. The model does not use batch normalization; instead, instance normalization is used as a very simple type of normalization and involves standardizing (e.g., scaling to a standard Gaussian) the values on each feature map. The intent is to remove image-specific contrast information from the image during image generation, resulting in better generated images. Generator model uses a sequence of down sampling convolutional blocks to encode the input image, a few residual network (ResNet) convolutional blocks to transform the image, and a few upsampling convolutional blocks to generate the output image. Figure 5.3. shows a training diagram for the Cycle GAN.

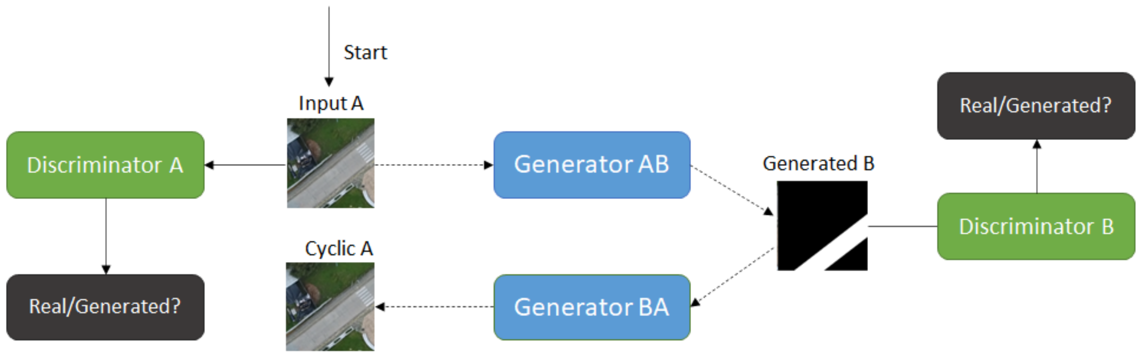


Figure 5.3. Unpaired image to image translation training diagram.

An unpaired image to image translation model based on a Cycle GAN is trained using a dataset of 649 images of 256x256 pixels, 100 epochs, and learning rate of 0.02 hyperparameters. A python script to train the Cycle GAN can be found in Annex 1.

5.1.3 Semantic Segmentation (U-Net) and Computer Vision Transformers (CvT)

Image semantic segmentation and image generation are two related processes in deep learning, the first one encodes image information, then it decodes information, which is a generative procedure (Ronneberger et al., 2015). Generation also performs encoding and decoding over input images, but it assigns a score for every generated image until generation is improved. This extra step ensures quality of the generated images, and it leads to believe that generation tends to produce a better version of inference masks when compared to segmentation. The default U-Net architecture for semantic segmentation (Ronneberger et al., 2015) is implemented based on a ResNet34 backbone with a Learning Rate of 0.02, a batch size (bs) of 8. It is trained during 100 epochs and used ninety degrees rotations for data augmentation.

We employed State-of-the-art Computer Vision Transformers (CvT) uses three blocks of attention modules and a VGG11 back bone. This approach is trained during 150 epochs in 1000 images of 500x500 pixels since it demands more data and longer training. CvT implemented architecture is presented in Figure 5.4 based on (Wu et al., 2021).

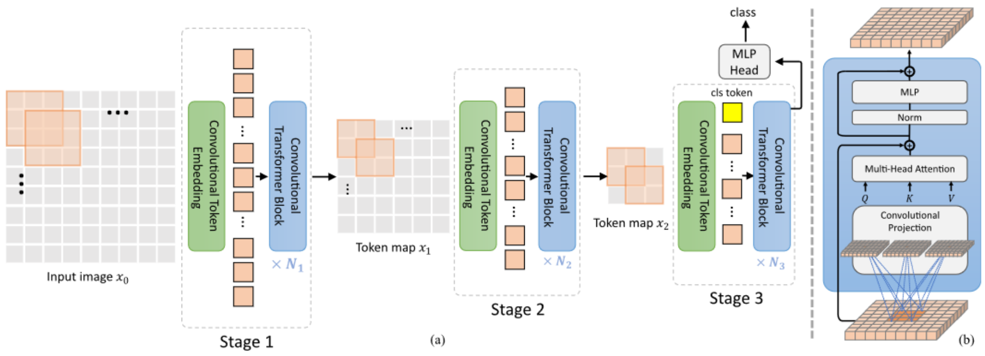


Figure 5.4. Computer Vision Transformers architecture.

5.2 Experiments and Results

The following tests are performed to challenge the previously chosen conditional image to image translation generative model. Since the generation of good quality binary masks pursues the same objective as semantic segmentation, the metrics to evaluate it can be the same. For a raster mask, we evaluate image to image translation model performance based on the average of IoU metric ($mIoU$), over the generated masks raster format calculated for all the images of a test dataset.

5.2.1 Comparing Results of Conditional and Unpaired Generative Models

Image Semantic Segmentation (U-Net), Computer Vision Transformers (CvT), conditional (Pix2Pix), and unpaired image translation (CycleGAN) generative approaches are tested over the same dataset, the Massachusetts Building dataset, and qualitative and quantitative results are compared with the aim of choosing the right model for image to mask translation in the geographic object realm.

Qualitative results in Figure 5.5. show that the U-Net creates irregular segmentation masks, and rounded buildings, but also false positive pixels, even in the green objects over the training dataset. Transformers obtained rounded borders of buildings, and several false positives specially in pixels that connect separate buildings. Generation based on conditional translation seems to create better masks than U-Net and Transformers which seem to worsen results of generation when added, perhaps due to the small number of examples used here compared to the reported in the original CvT paper. Figure 5.6 illustrates qualitative results of U-Net, CvT, and Pix2Pix to image to mask translation over the Massachusetts Buildings dataset.

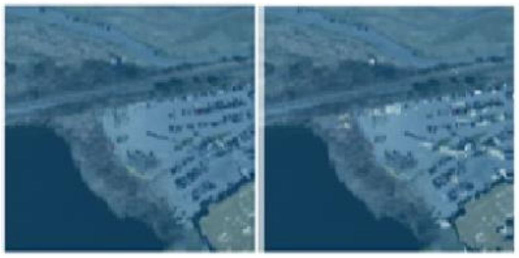


Figure 5.5. Qualitative results of U-Net on the Massachusetts Building Dataset. GT and segmentation masks by U-Net are represented in white pixels and placed over the aerial image source.

Quantitative results in Figure 5.6. show that conditional image to mask translation based on Pix2Pix generates more regular, cleaner and right-angle masks compared to the other models. Pix2Pix model exhibits masks closer to the ground truth for buildings compared to the other approaches. Also, the false positive pixels are less clustered which can be removed after a post processing cleaning procedure. Pix2Pix model. Experiments on this method applied to point, line, and polygon objects are presented in a coming section. Figure 5.6 shows quantitative results, based on IoU, when unpaired and conditional image to mask translation models are compared. Table 5.1 compares the results.

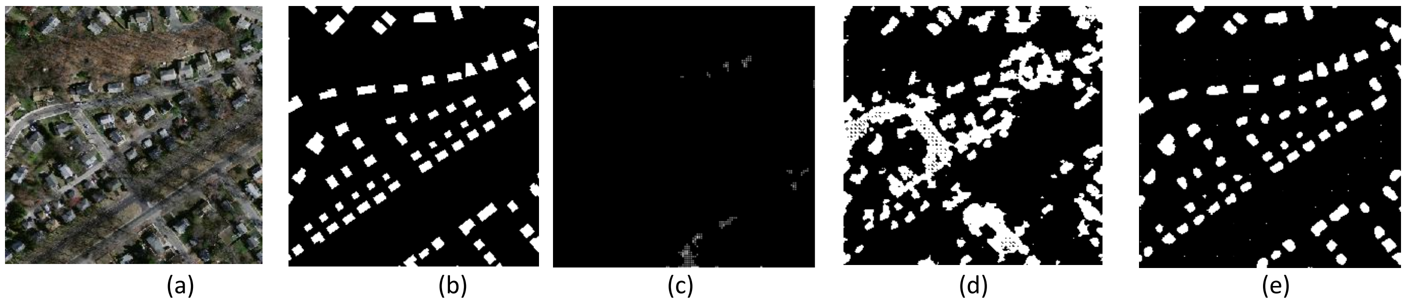


Figure 5.6. Results for Semantic segmentation, (a) Image, (b) Ground truth, (c) CvT, (d) unconditional and, (e) conditional image to mask translation. Results are obtained using the same 649 image examples of the Massachusetts Building dataset of 256x256 pixels over 100 training epochs.

Table 5.1. Unpaired, conditional translation and others comparative results

Model	**mIoU**
CvT	0.11
Unpaired translation CycleGAN	0.19

U-Net	0.67
Conditional translation cGAN (Pix2Pix)	0.79

5.2.2 Image to Point Mask Translation

For testing the image translation to point objects, we use the domino dataset described in a previous section, in which vehicles are represented as point masks using 1 meter buffer distance. Figure 5.7. illustrates the Image-to-Image Translation Model components to generate point masks.

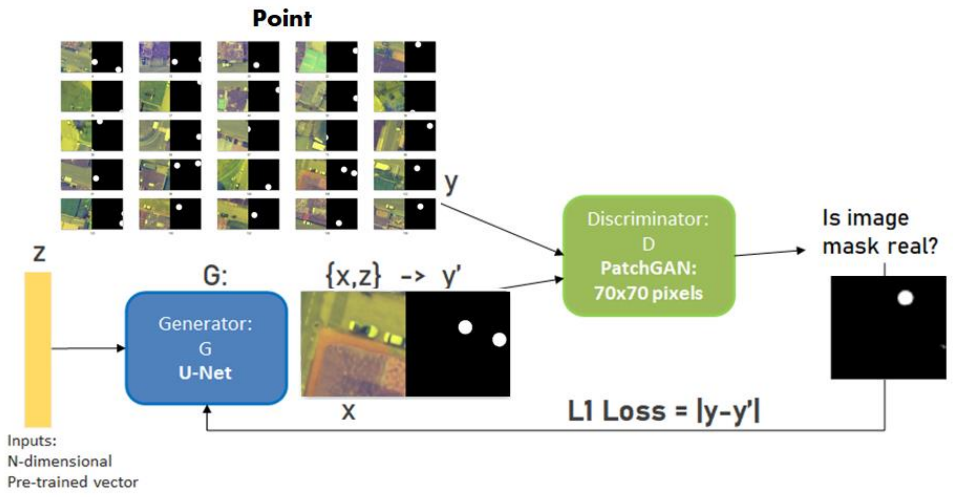

Point
y
z
G:
{x,z} -> y'
Discriminator: D PatchGAN: 70x70 pixels
Is image mask real?
Generator: G U-Net
x
L1 Loss = |y-y'|
Inputs: N-dimensional Pre-trained vector

Figure 5.7. Pix2Pix training diagram for image-to-point-mask translation.

Figure 5.8 shows how the generator loss for the testing dataset using 1m buffer distance behaves better than larger distances for vehicle translation. Figure 5.9 shows the vehicle generation using 100 epochs of training on a dataset of 250 examples of 1 meter of buffer distance, image size of 256x256, and a batch size (bs) of 1.

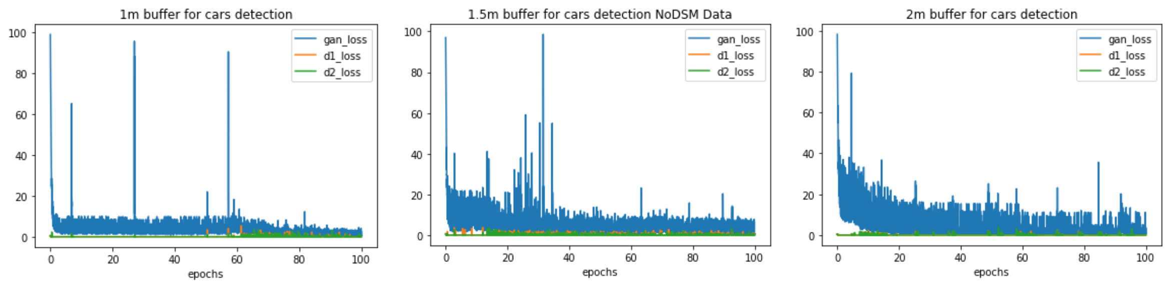


Figure 5.8. Generator and discriminator losses for image to point mask translation. 1m buffer distance vehicle dataset exhibits less variance in generator loss (blue curve) than in the both discriminator losses, d1 (loss of discriminator in real images) and d2 (loss of discriminator in generated images).

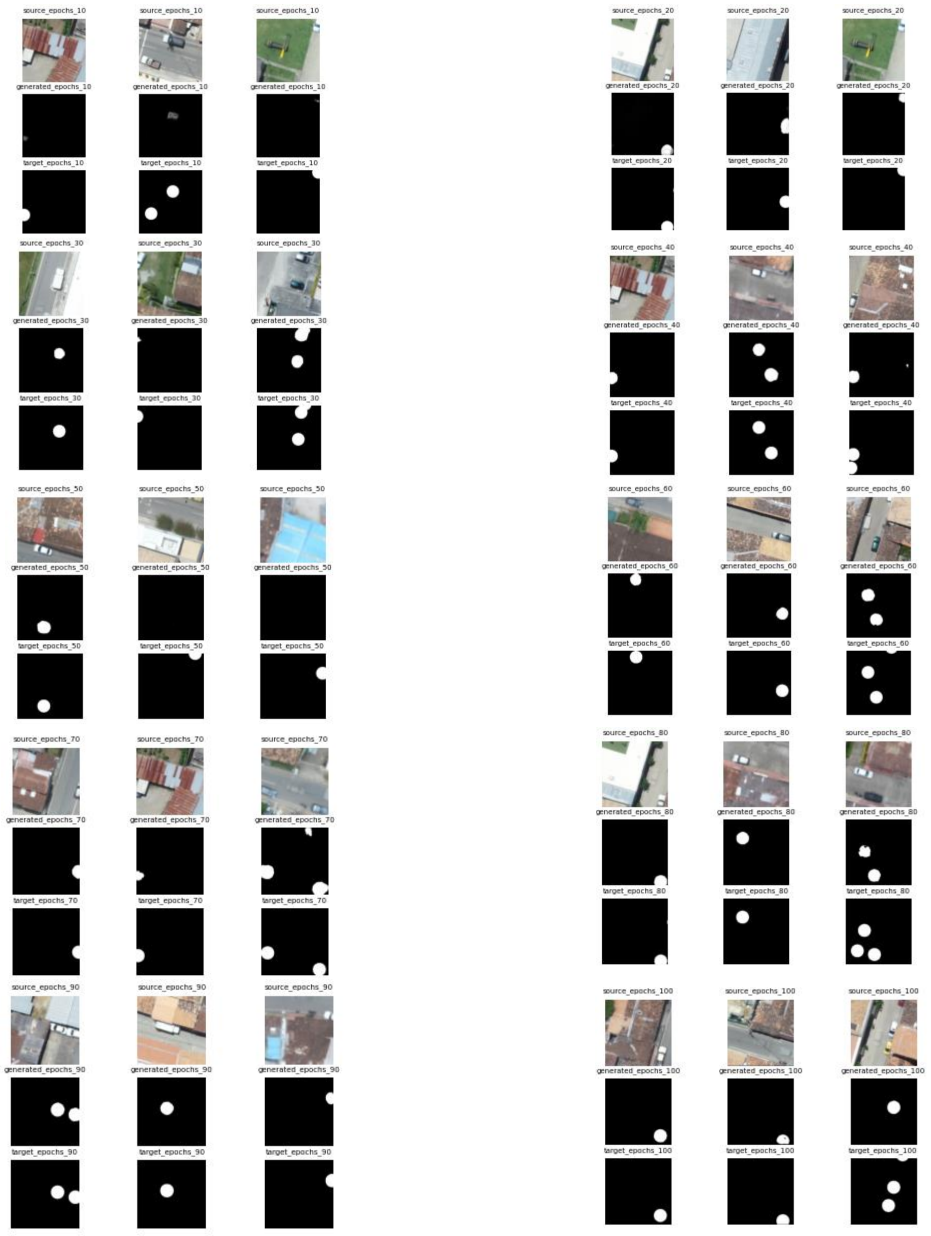


Figure 5.9. Qualitative results for image to point-mask. Upper images are drone, the images in the bottom are the target latent space and the ones in the middle are the generated or translated target images. Successive epochs of training show better qualitative results. Different images are shown in each training example.

5.2.3 Image to Line Mask Translation

For image to line mask translation, we use the TSQS road dataset produced by us, in which roads are represented by line masks of 3m buffer distance from the centerline. Figure 5.10. illustrates the Image-to-Image Translation Model components to generate point masks.

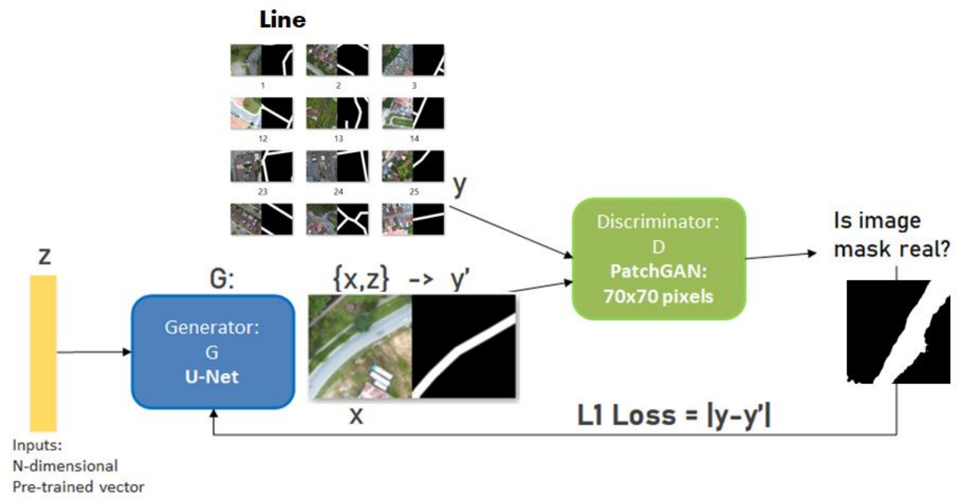


Figure 5.10. Pix2Pix training diagram for image-to-line-mask translation.

Figure 5.11 shows the qualitative results of translated line masks using 100 epochs, 250 examples, image size of 256x256, and a bs of 1.

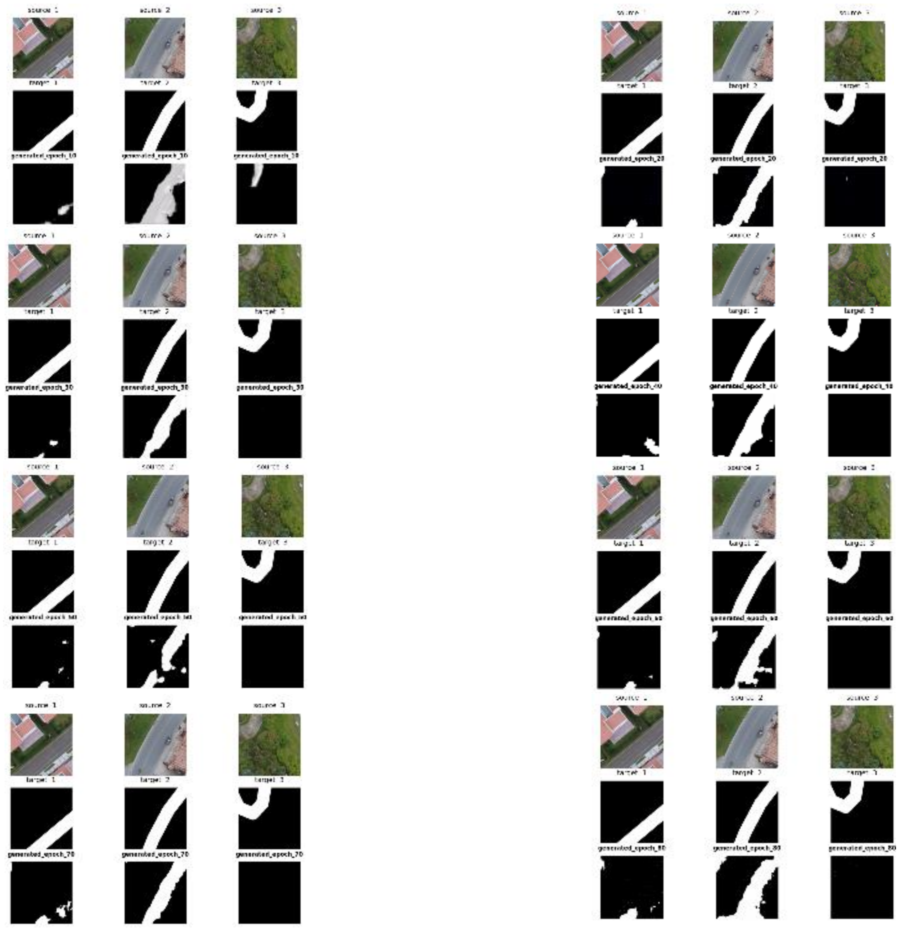


Figure 5.11. Qualitative results for image to line mask translation. 3m buffer distance road dataset. Up most images are drone, middle are real images, and images in the bottom correspond to translated road images. Successive training shows an improvement in the continuity of translation masks. The same image is shown in each training epoch to show learning evolution, third image case shows no learning due to a small number of examples of that type.

5.2.4 Image to Polygon Mask Translation

For polygon mask translation, the Massachusetts building dataset is tested with 100 epochs, 649 examples, 256x256 image size, and bs of 1. In the case of polygons, the buffer distance parameter is zero (d = 0) since class imbalance ratio is larger compared to points and lines, 16% of pixels are positive class. Figure 5.12. shows the Image-to-Image Translation Model components to generate polygon masks. Qualitative results for the polygon mask translation are shown in Figure 5.13.

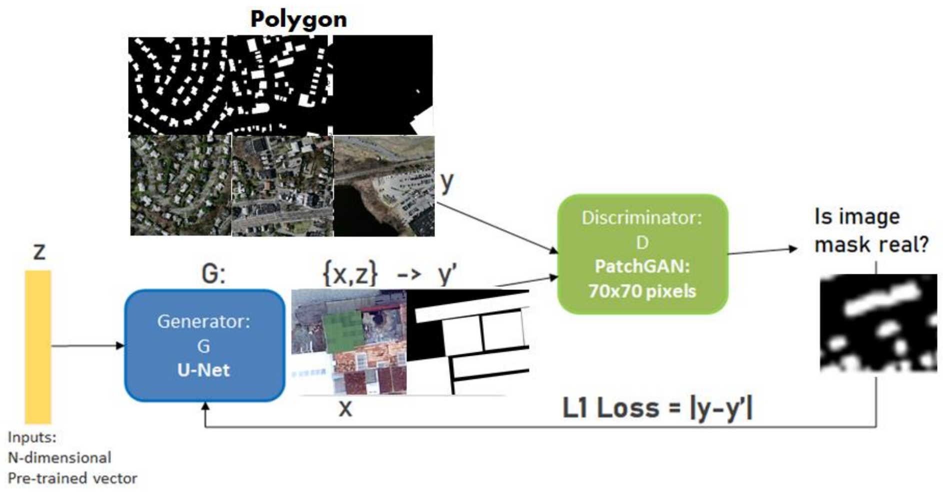

Polygon
y
z
G:
{x,z} -> y'
Generator: G U-Net
Discriminator: D PatchGAN: 70x70 pixels
Is image mask real?
x
L1 Loss = |y-y'|
Inputs: N-dimensional Pre-trained vector

Figure 5.12. Pix2Pix training diagram for image-to-polygon-mask translation.

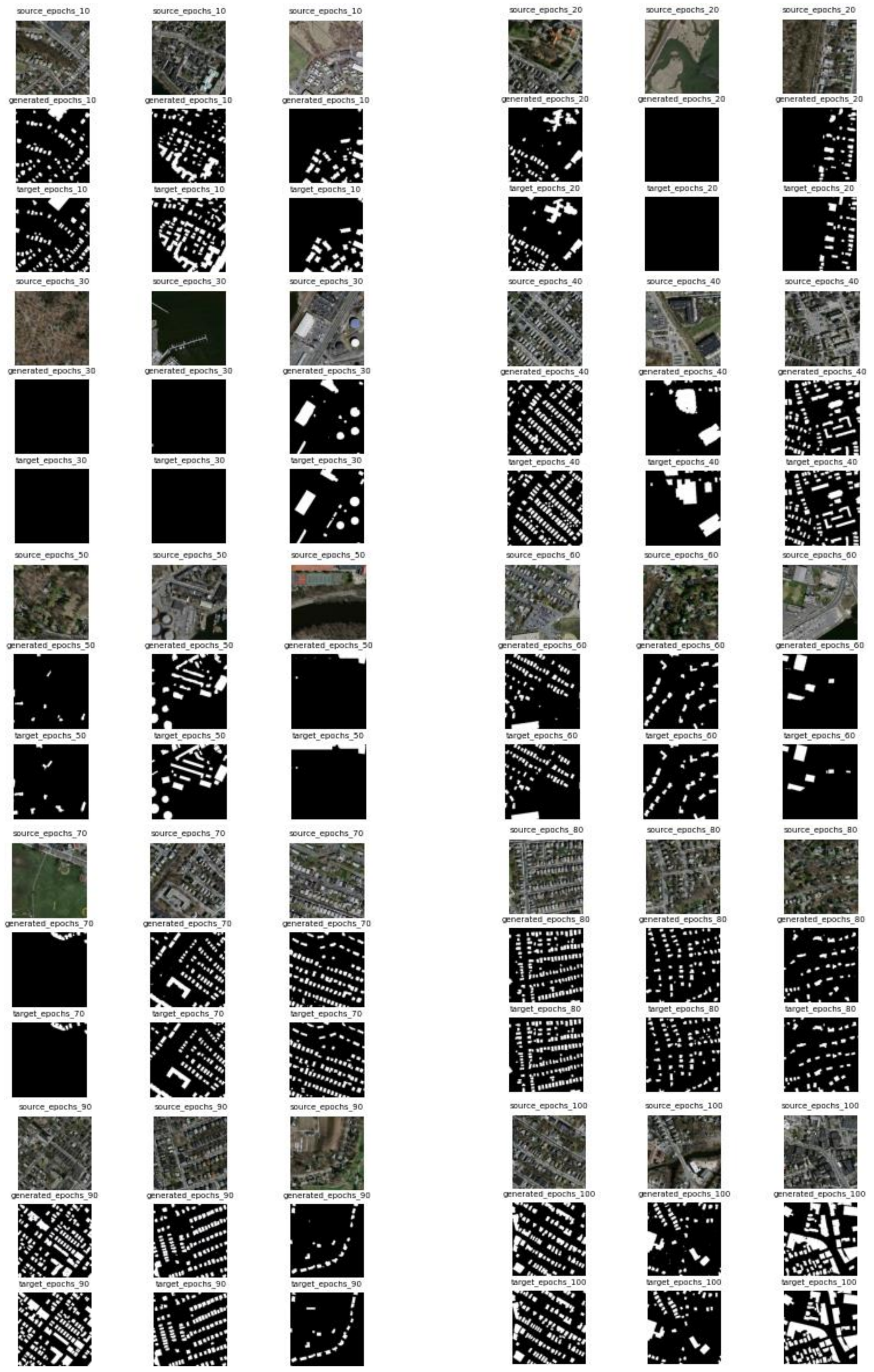
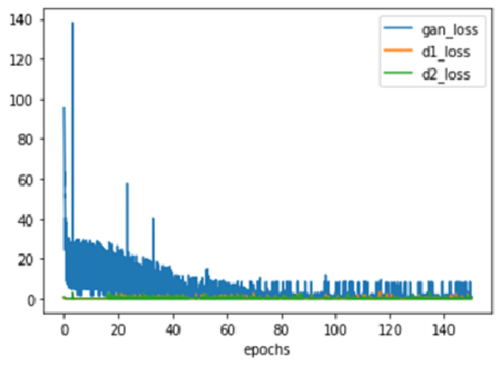

source_epochs_10
source_epochs_10
source_epochs_10
generated_epochs_10
generated_epochs_10
generated_epochs_10
target_epochs_10
target_epochs_10
target_epochs_10
source_epochs_20
source_epochs_20
source_epochs_20
generated_epochs_20
generated_epochs_20
generated_epochs_20
target_epochs_20
target_epochs_20
target_epochs_20
source_epochs_30
source_epochs_30
source_epochs_30
generated_epochs_30
generated_epochs_30
generated_epochs_30
target_epochs_30
target_epochs_30
target_epochs_30
source_epochs_40
source_epochs_40
source_epochs_40
generated_epochs_40
generated_epochs_40
generated_epochs_40
target_epochs_40
target_epochs_40
target_epochs_40
source_epochs_50
source_epochs_50
source_epochs_50
generated_epochs_50
generated_epochs_50
generated_epochs_50
target_epochs_50
target_epochs_50
target_epochs_50
source_epochs_60
source_epochs_60
source_epochs_60
generated_epochs_60
generated_epochs_60
generated_epochs_60
target_epochs_60
target_epochs_60
target_epochs_60
source_epochs_70
source_epochs_70
source_epochs_70
generated_epochs_70
generated_epochs_70
generated_epochs_70
target_epochs_70
target_epochs_70
target_epochs_70
source_epochs_80
source_epochs_80
source_epochs_80
generated_epochs_80
generated_epochs_80
generated_epochs_80
target_epochs_80
target_epochs_80
target_epochs_80
source_epochs_90
source_epochs_90
source_epochs_90
generated_epochs_90
generated_epochs_90
generated_epochs_90
target_epochs_90
target_epochs_90
target_epochs_90
source_epochs_100
source_epochs_100
source_epochs_100
generated_epochs_100
generated_epochs_100
generated_epochs_100
target_epochs_100
target_epochs_100
target_epochs_100

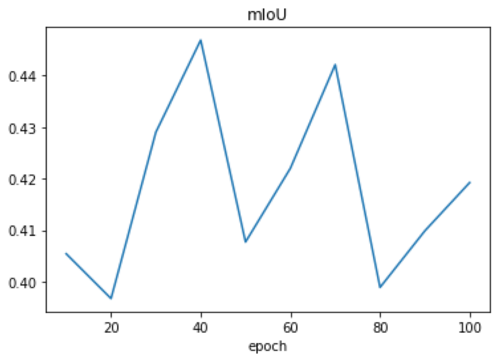
Figure 5.13. Qualitative results for image to polygon mask translation for the Massachusetts Building Dataset. Up most images are aerial imagery with low resolution compared to drone, bottom are real images of buildings, and images in the middle correspond to translated building images. In this case, training shows good qualitative results, comparing generated versus ground truth, since the very first epochs. This is probably due to a less imbalance binary class dataset. Special attention is deserved for round shape buildings, the apparition of some true negatives and false positives.

5.2.5 Model Hyperparameters

Translation model hyperparameters are learning rate, lambda value, number of epochs, and batch size. A dataset of 250 images of 256x256 pixels, with suggested values of learning rate = 0.002 (lr) and lambda value = 100 by (Isola et al., 2017) were used. Experiments over the last two parameters are shown in Figure 5.14. It illustrates that increasing the number of epochs above 100 does not reduce generator loss or its variance and does not generate better results. A graph of discriminator and generator losses and mIoU metric against number of epochs show that increasing the number of epochs does not necessarily improve the mIoU metric. As expected, a higher bs improved mIoU in 0.01. An experiment with bs=1, bs=6, and bs=10 only produced an increase of 0.01 (1%) in mIoU between the two extreme values. Figure 5.15. exhibits the results of bs vs mIoU with a dataset of 1100 images of 256x256. However, bs=10 decreased the training time of model in around five times compared to bs=1.

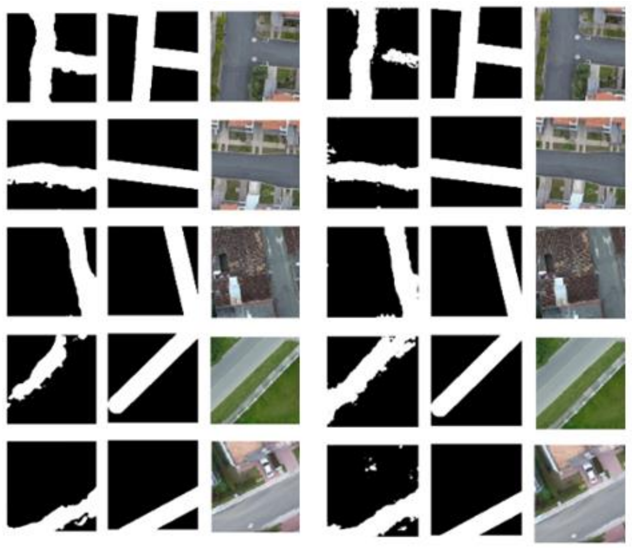


(a)



(b)

Figure 5.14. Effect of number of epochs. (a) Discriminator and generator losses vs epochs, (b) mIoU vs epochs. As shown in graphs, increasing the number of epochs to 150 does not seem to improve generator or discriminator losses values, and not their variance. On the other hand, more epochs of training do not imply a model with better metrics.



(a) (b)

Figure 5.15. Effect of batch size. (a) batch size = 1 mIoU = 0.713, (b) batch size = 10 mIoU = 0.723.

5.2.6 Effects of Imbalance on Image to Mask Translation

Imbalance pixel classes affects deep learning models including image to mask translation (Gao et al., 2018). Figure 5.16 presents how imbalance changes with the increase in the buffer distance parameter of the road dataset to create the binary mask. The larger the buffer distance the more balance dataset is obtained.

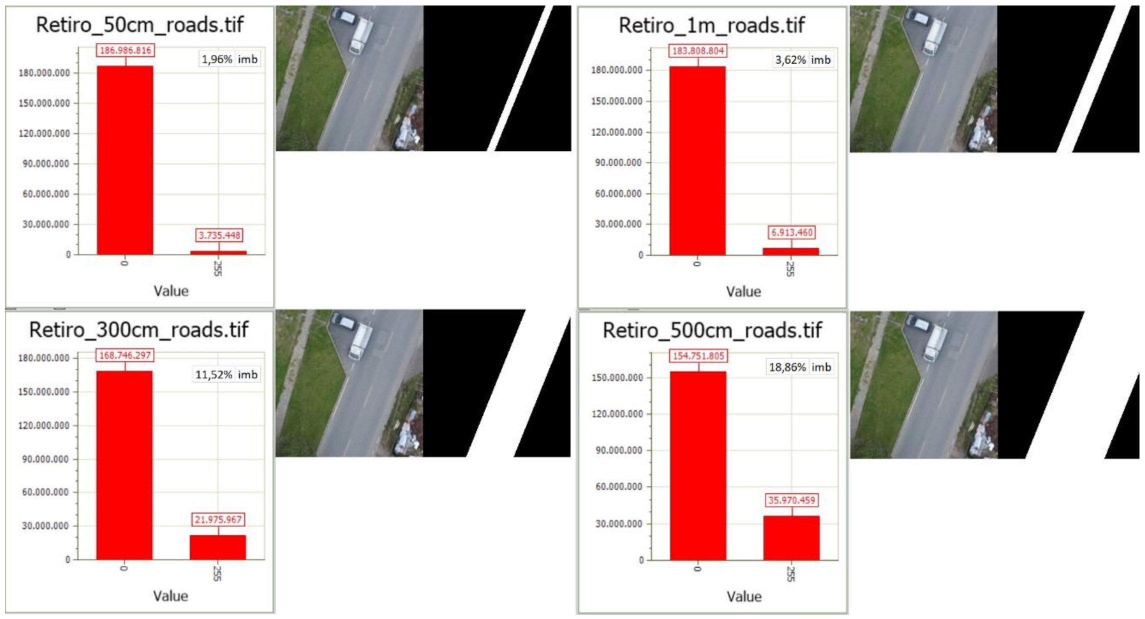
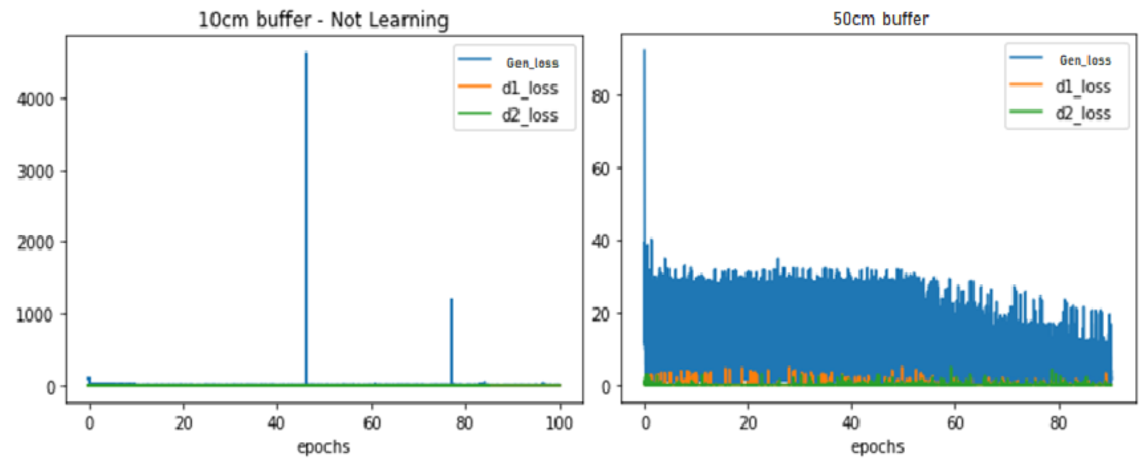
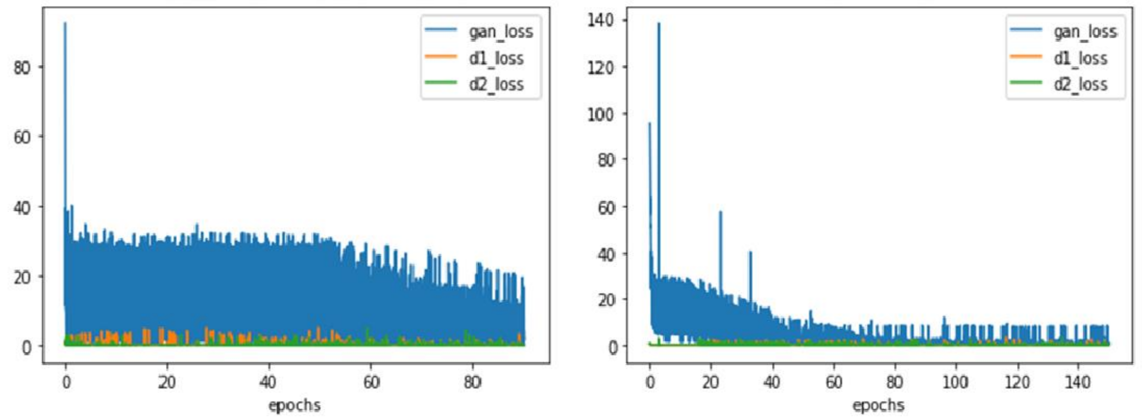


Figure 5.16. Imbalance vs buffer distance in roads. These values are averaged for the entire raster masks not yet tessellated.

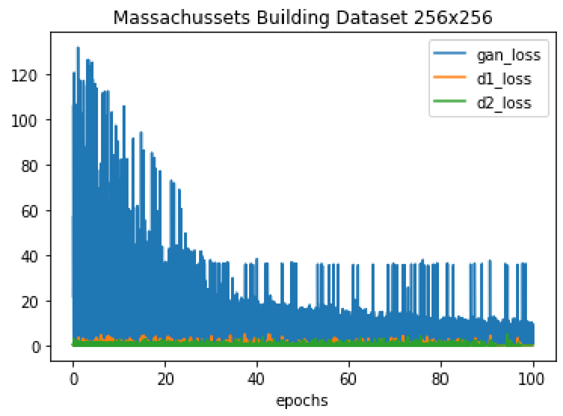
Figure 5.17. illustrates how point, line, and polygon masks generation from images is affected by imbalance. Generator and discriminator loss graphs show the impact in model learning behavior. Different examples of imbalance classes show that the less imbalance the class is, the sooner the model starts learning. This is, the generator loss decreases with training, and it does not diverge as it is in the case of an extremely imbalance, i.e., the 10 cm buffer dataset (< 0.05% of positive class).



(a)



(b)



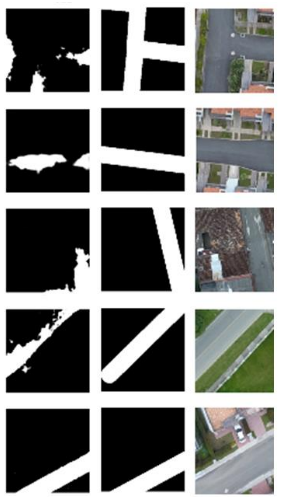
(c)

Figure 5.17. Effects of imbalance pixel classes on image translation. (a) Left image shows losses of a 10cm buffer distance domino dataset for vehicles with an imbalance ratio of 0.04%, right image shows 50 cm buffer distance with an imbalance ratio of 1%. (b) Left image shows losses of a 50cm buffer distance for roads dataset

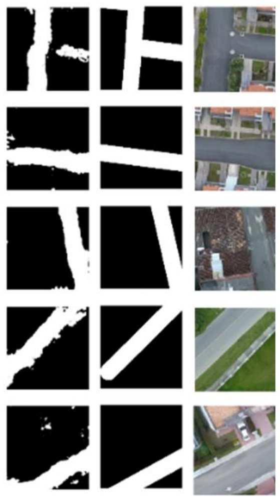
with an imbalance ratio of 7%, right image shows 1 meter buffer distance roads dataset with an imbalance ratio of 13.5%. (c) Massachusetts Building dataset has an imbalance ratio of 16%. Polygon objects datasets tend to be less imbalance, it means they have a larger imbalance ratio.

A less imbalance dataset, it means a larger imbalance ratio, allows the translation model to learn faster (allowing the generator and discriminator losses to decrease in equilibrium between discriminator and generator). A python script to calculate the imbalance ratio of a (img-msk) dataset, delete the complete black masks, and to obtain a dataset with a pre-defined imbalance ratio is presented in Annex 1. An experiment with four different values of imbalance ratio is carried out. Imbalance ratios less or equal than 1% are not able to generate masks regardless the number of training pairs (tested up to 1000). Results for Imbalance ratios greater than 5%, 10%, and 20% are exhibited in Figure 5.18. It seems that having more pixels per class improves mIoU results.

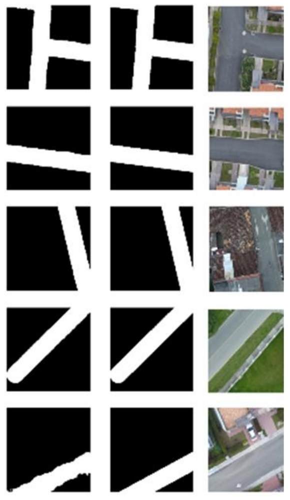
(a)



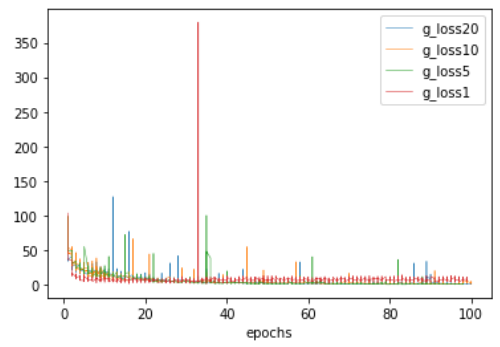
(b)



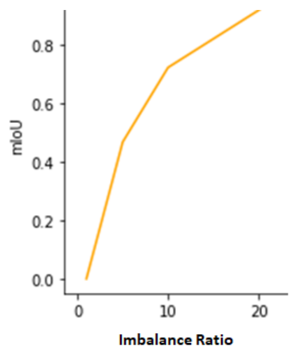
(c)



(d)



(e)



(f)

Figure 5.18. Effects of imbalance on the mIoU. (a) Highly Imbalance masks, (b) Imbalance masks, (c) Medium Imbalance masks, (d) Balanced masks, (e) graphs of generator losses of different imbalance datasets: they show similar behavior except for the 1% imbalance ratio dataset, which exploded, (f) graph of mIoU vs Imbalance ratio. Datasets with imbalance ratio >=20% showed the best results.

Table 5.2. Imbalance vs mIoU in road dataset

Imbalance for roads	mIoU
<=1%	0-0.1
<5%	0.353
>=5%	0.468
>=10%	0.723
>=20%	0.939

5.2.7 Effects of Number of Examples and Size of Images in Mask Translation

There has been a tendency to believe that the larger the training dataset the better the results. However, that depends on the model configuration in terms of its hyperparameters and depth (Abdollahi et al., 2019). To challenge this, the same model configuration is trained using the same dataset with different number of examples. A model training with 250, 500, 1000 and 1500 images of 256x256 pixels and roads masks of 3 meters was carried out, and the results were compared. Hyperparameters used were: 100 epochs, LR=0.02, Dropout=0.5 and no data augmentation was done on the dataset. Figure 5.19. illustrates the mIoU metric results for different number of examples of the training road dataset. All the models exhibit problems in the borders. Figure 5.20. presents two graphs, the mIoU vs number of examples and the training time vs number of examples.

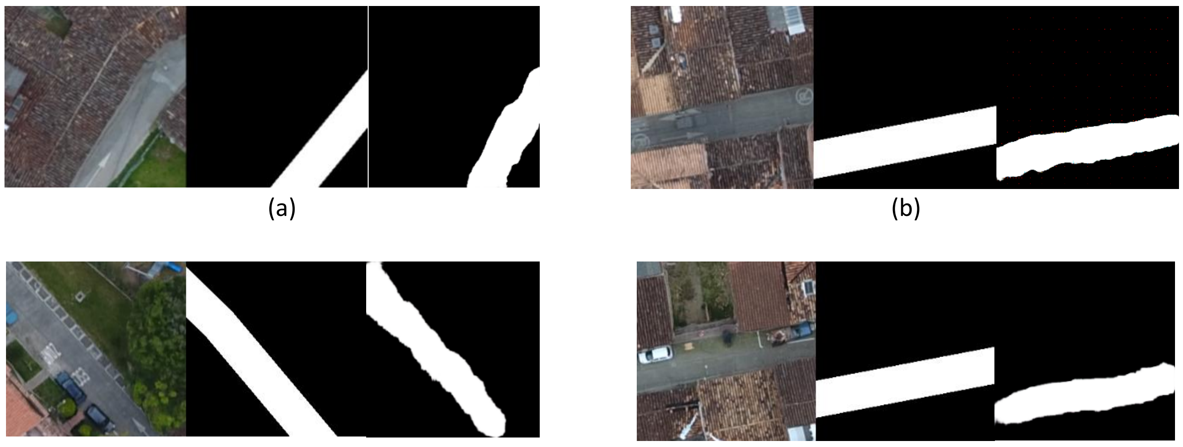
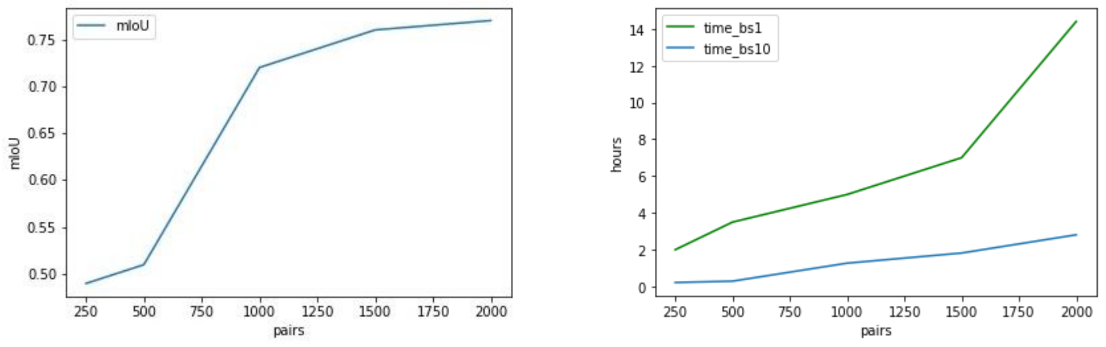


Figure 5.19. Effect of number of examples in translation model performance. (a) 250 examples (b) 500 examples, (c) 1000 examples and (d) 1500 examples. As it is expected the higher the number of (image,mask) examples the better the metric. Time required to train different models in Google Collab started in 2 hours for 250 (img,msk) pairs of 256x256 px, and ended in 5 to 7 hours for 1500 (img,msk) pairs. A model for 2000 examples took 13 to 15 hours of training and obtained a mIoU of 0.76 in the test set. However, when a batch_size of 10 was used, the training time decreased five times, so a model with 250 images was trained in around 25 minutes.

Table 5.3. Examples vs performance in road dataset

Number of pairs	mIoU
250	0.35-0.48
500	0.49-0.55
1000	0.60-0.72
1500	0.75-0.90

>=2000	>=0.939



(a) (b)

Figure 5.20. Effect of number of examples in training time. (a) mIoU vs number of (img,msk) pairs, (b) Training time vs number of (img,msk) pairs examples using a bs = 1. As graphs suggest, the best trained models can be obtained within 1000 and 1500 examples.

Also, different image size is employed for training models to challenge which is the most appropriate image size for mask translation. Image size is increased to 512x512 pixels, and then to 1024 pixels, and different models were trained. All the hyperparameters are constant and the model performs image resampling to 256x256 pixels. 256 pixels correspond to 15.4 m at 6 cm/px average GSD, 30.7 m for images of 512, and 61.4 m for 1024 pixels, this is almost the size of a block in a standard city. Results seem to show that the larger the image size the better the IoU metric of the test dataset. Figure 5.21. presents the mIoU metric results for 512, and 1024 pixels images pairs of the training road dataset. As stated in (Van Etten et al., 2019), IoU metric penalizes more false positives or irregularities (generated image 1, IoU=0.58), than false negatives or discontinuities of masks (generated image 2, IoU=0.73). As observed, major cases of false positives and false negatives are due to spectral similarities or spectral changes for models trained on both sizes. Table 5.4. compares size vs performance.

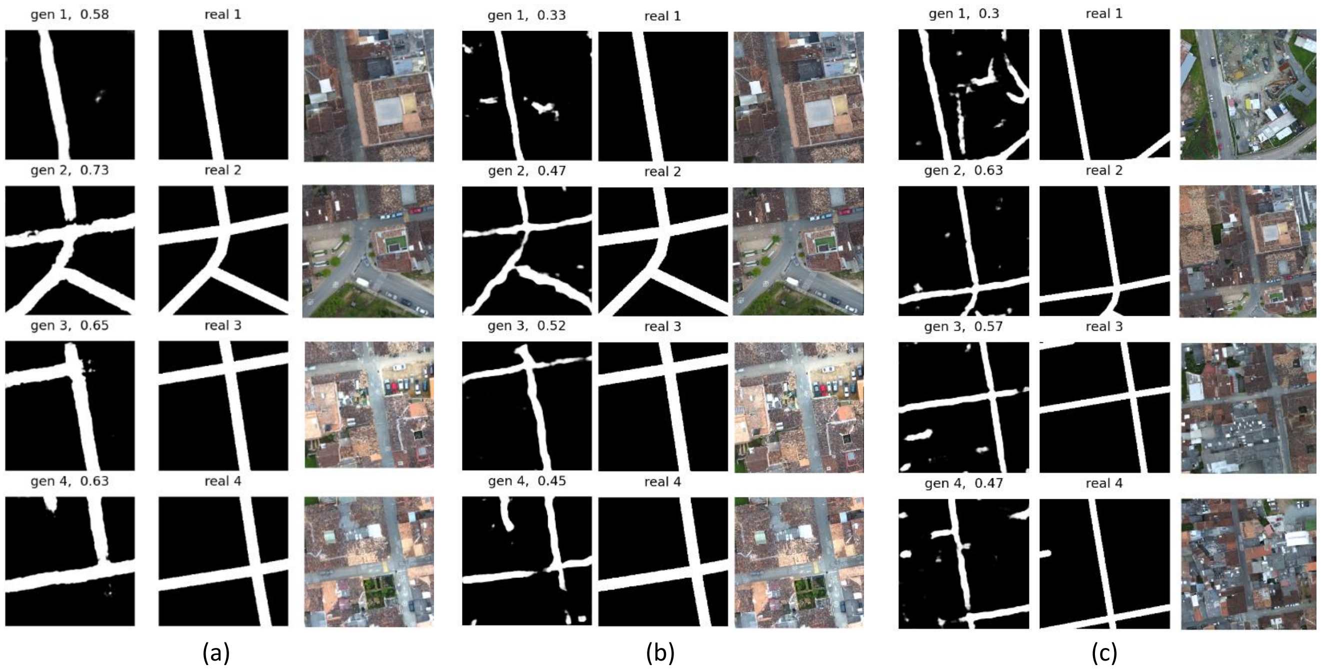


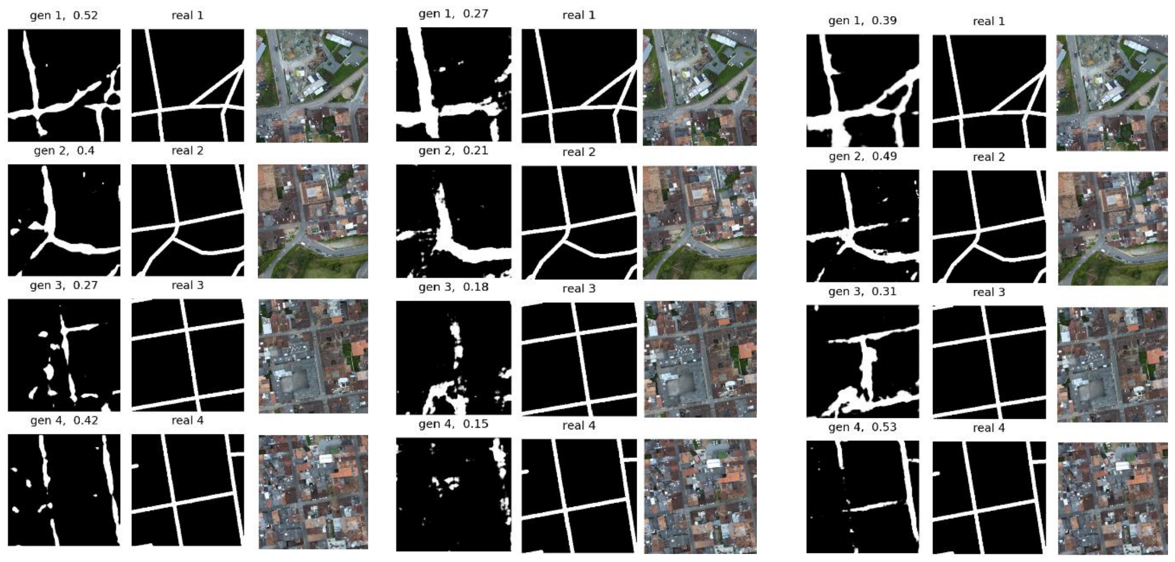
Figure 5.21. Effect of image size in translation model performance. (a) Model trained with 512x512 px, (b) Model trained with 1024x1024 px, (c) model trained with 1500x1500 px (90m). These results suggest that training with larger image sizes improves the performance but generates more discontinuities and data availability may become an issue.

Table 5.4. size vs performance in road dataset

Size (px)	mIoU	Test pairs
512x512	0.65	378
1024x1024	0.55	329
1500x1500	0.59	329

5.2.8 Multi-Scale Training

To challenge the effect of multi-scale training in model performance, two models are train over the same dataset but in different scale order. Starting first with images of 256x256, then 512x512 and ending up training with images of 1024x1024 pixels. After that, training in reverse order is performed, and a comparison of both models' results is carried out. Figure 5.22. illustrates both cases. Table 5.5 shows the results when training bottom up and top down.



(a) (b) (c)

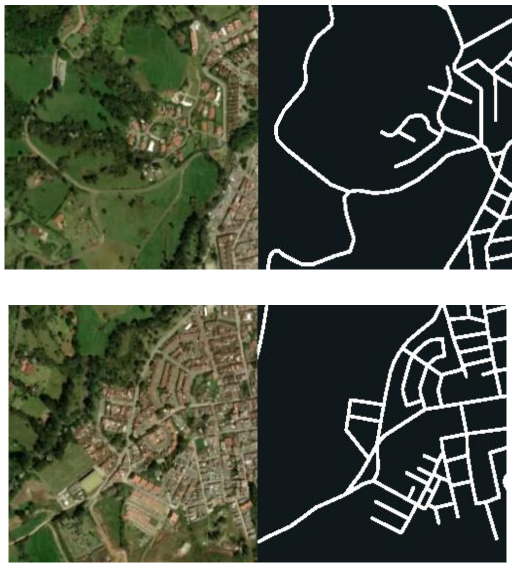
Figure 5.22. Multi-Scale Training. (a) Training bottom-up 256x256 to 1024x1024 px (b), Training top-down 1024x1024 to 256x256 px, (c) Training with 256x256 px and inferring at 1500x1500 px.

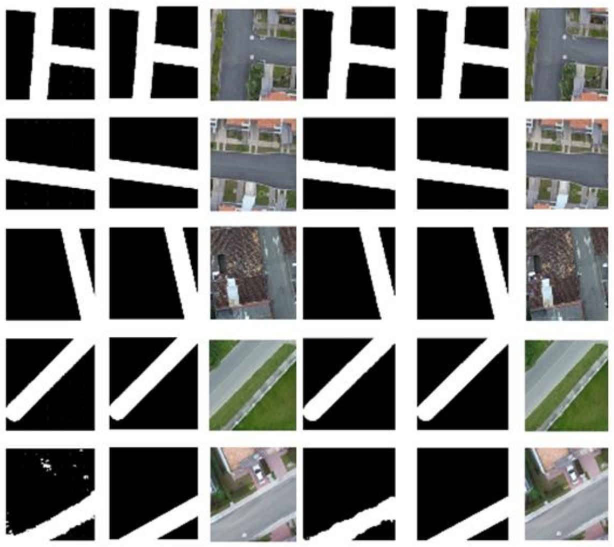
Table 5.5. Bottom-up vs top-down training

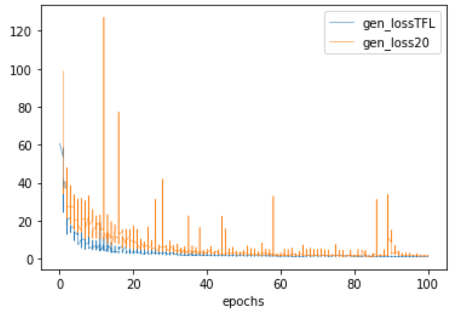
Size (px)	mIoU	Test pairs
256x256 to 1024x1024	0.40	378
1024x1024 to 256x256	0.28	378
Training only at 256 but inferring at 1500x1500	0.45	378

5.2.9 Transfer Learning using Satellite Imagery

The experiment for transfer learning consists of using the weights of a pre-trained model with 300 satellite images road masks pairs of 256x256 pixels with a GSD of 2.25 m/px over 100 epochs. The pre-trained model is then retrained it with an imbalance ratio dataset >20% used in the imbalance experiment. Figure 5.23. shows the effect of applying transfer learning, comparing the results with the model that is not previously trained.


(a)


(b)



(c)

Figure 5.23. Effect of Transfer Learning. (a) Satellite images and corresponding road masks, obtained from Mapbox and OSM respectively for model weights transferring, (b) Left image, Model with transfer learning (using weights from satellite), mIoU is: 0.909. Right image, model without transfer learning, mIoU is: 0.939, (c) generator loss of the transfer learning model exhibits less variability and smaller values in the training process compared to base model.

5.2.10 Effects of Data Augmentation on Image to Mask Translation

Data augmentation has been considered the horse power to deal with the class imbalance problem (S. Wang et al., 2016), multiplying the number of examples by many times and reducing in this way the dominant class in proportion to the positive class in a dataset. Which method of augmentation is better for drone imagery in image-to-image translation is still an open question. How is it possible to include data fusion with the purpose of reducing variance of generation and discrimination losses, and in this way, to reduce the training, and obtain better results. Experiments for different augmentation methods are presented next.

- **Geometric and Spectral Augmentation on Image to Mask Translation**

Geometric augmentation is implemented as ten degrees clockwise rotations, mirroring right to left and up to down, and horizontal overlapping of 30 %. Changes in 10% up over contrast, brightness, and gamma values are performed over a separate dataset with an imbalance ratio >10% to compare the impact respect to the geometric augmentation. Figure 5.24. shows a comparison of data augmentation results using different methods. As expected, the maximum performance of the model was obtained when all the augmentation methods were applied, however, the application of overlapping and rotation plus mirroring obtained almost the same results. Table 5.6. summarizes augmentation results.

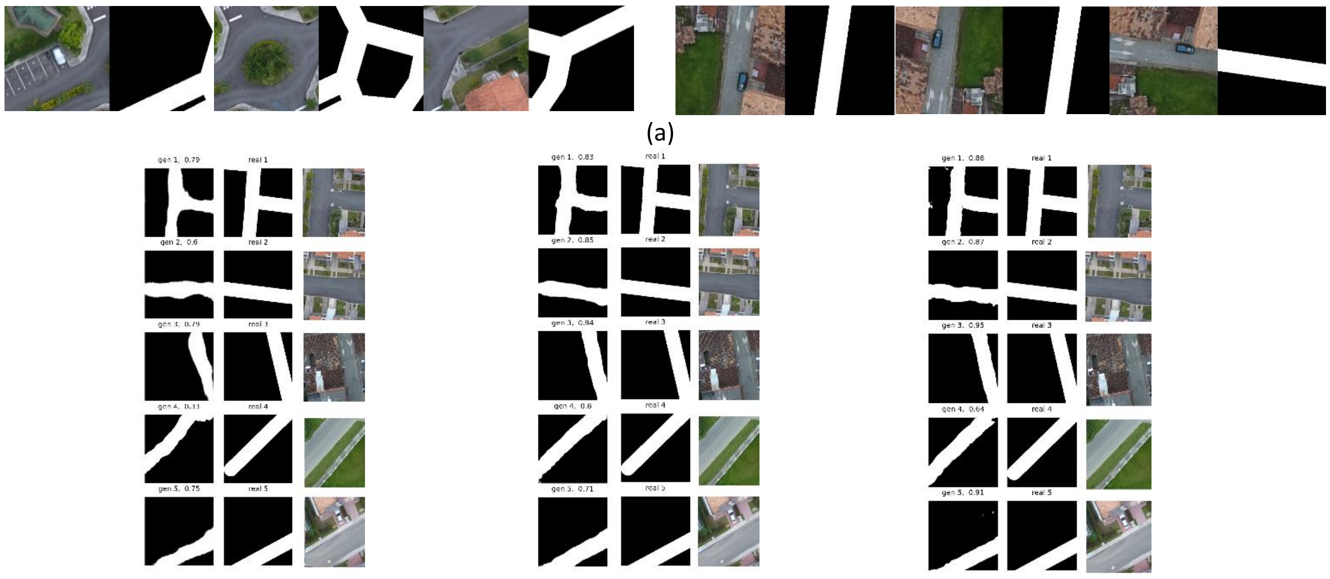


Figure 5.24. Data Augmentation Comparison. (a) 30% overlapping (left), mirroring (right), (b) Spectral augmentation, (c) Rotation, (d) All augmentation together

Table 5.6. Influence of data geometric and spectral augmentation

Augmentation Method	mIoU
Overlapping (30%)	0.794
Mirroring (vert. and hor.)	0.789
Rotation (10 degree increase clockwise)	0.779
Spectral (10% random increase in brightness, intensity and contrast)	0.659
All augmentation together	0.847

- **Data Fusion on Image to Mask Translation**

Data fusion is used to include additional information, in this case height, that comes from the DSM and the VARI index into one channel of the RGB three channel images, without the need of creating additional channels, which may increase the size of architecture, the training time or the computational resources.

Height augmentation: we want to evaluate the power of height information for the discrimination against spectral similarities of vehicles, roads and buildings against their surrounding objects. Also, which channel to replace by height data or it is better to add height to existing color channels. Our rationale is to initially remove blue channel since it is the less in importance and compare this with the sum of height to each channel values and re-scale values to [0,255] interval. Figure 5.25. shows how the DSM data depletes the variance of the three training losses in image translation model for roads. Variance of generator loss (blue graph), but also of the discriminator loss on the real images (orange graph) and the discriminator loss on the generated images (green graph) decreased, and losses are smaller than when height is not present in the data. This behavior keeps the same even after 100 epochs of training, and as it was studied before, no improvement is exhibited.

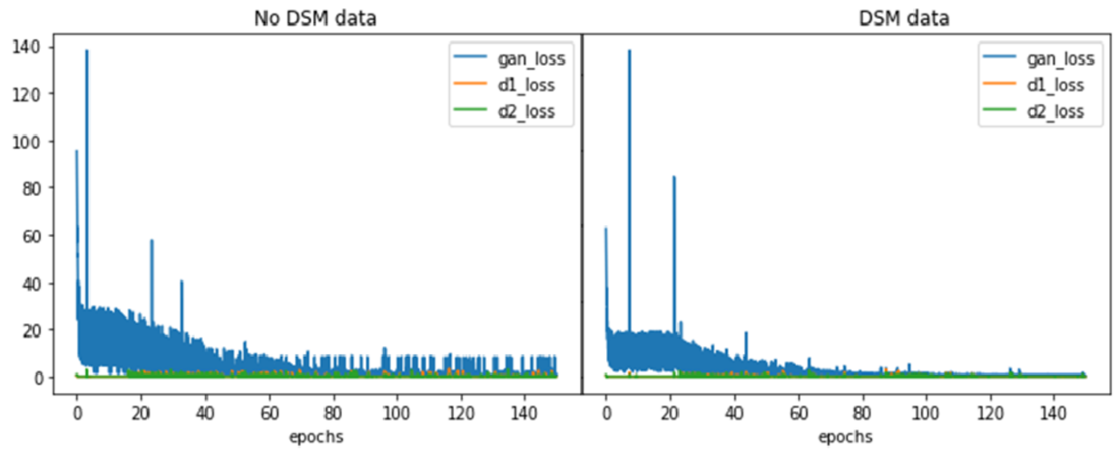
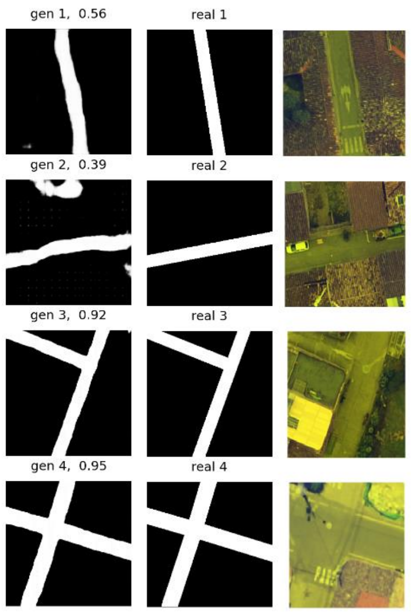
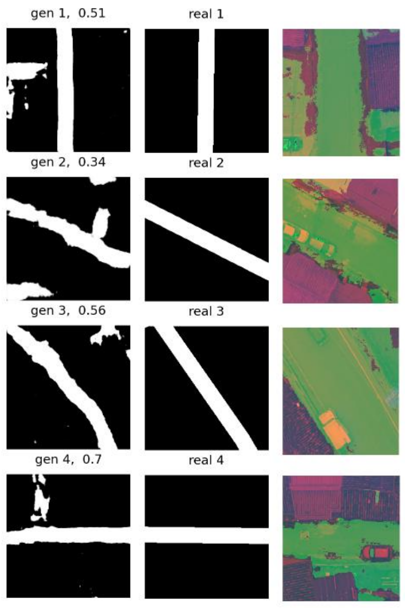


Figure 5.25. Effect of height information in image to mask translation of roads. Left image shows the constant decreasing of generator loss from the very first training epoch up to the 70th epoch, after that epoch, the three losses exhibit a high and constant variance even after 100 epochs. Right image shows an initial constant value of losses, they start decreasing after 20 epochs and continue with this behavior for the whole rest of training epochs, not apparent improvement is shown after the 100 epochs.

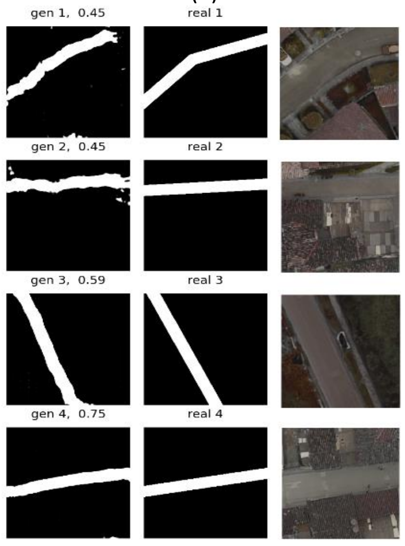
VARI Index: Vehicles, roads, and buildings have a close spatial relationship between them and with other neighbor objects like trees. Green color is a spectral discriminant of trees, thus including information how green is an object and removing atmospheric influence is keen to the extraction model capabilities. We exchange the green color channel with the VARI Index in the search of a better and faster learning of the green objects by the model. Figure 5.26. shows results including DSM and VARI index into the model. Table 5.7. summarizes the results obtained with different data fusion.



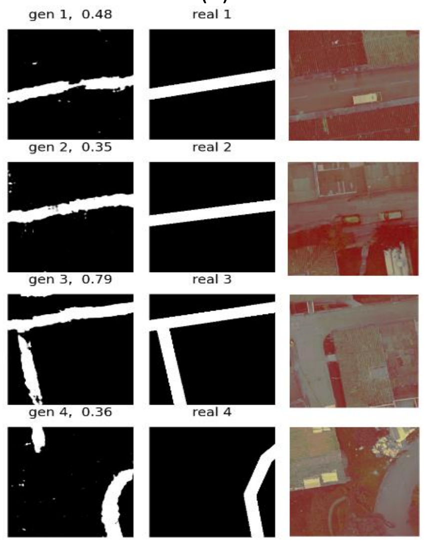
(a)



(b)



(c)



(d)

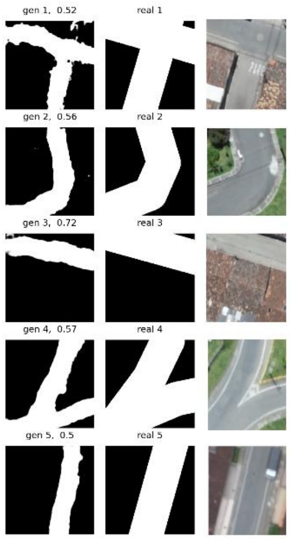
Figure 5.26. Data fusion on Image to Mask Translation. (a) RG-DSM, this model gets confused in unoccupied zones that are at the ground level and have a similar spectral response to roads, (b) RVARIB, this model seems to not discriminate heights very well, (c) HRGB, summing height values to the three RGB channels and re-scaling to [0,255] interval does not seem to be better than using height values in the blue band, (d) HRVARIB, it seems that combining VARI and DSM does not improve the results of only using height in the blue channel.

Table 5.7. Influence of data fusion

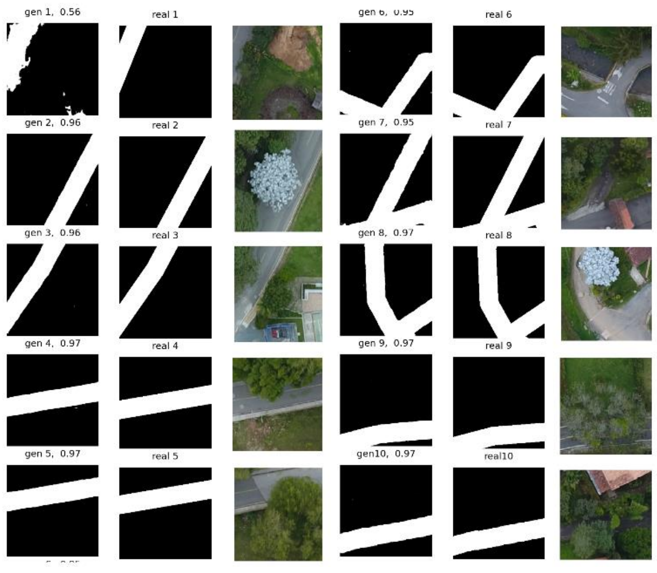
Fusion Method	**mIoU**
RG-DSM	0.725
RVARIB	0.549
HRGB	0.621
HRVARIB	0.508

5.2.11 Image to Mask Translation Model Generalization and Occlusions

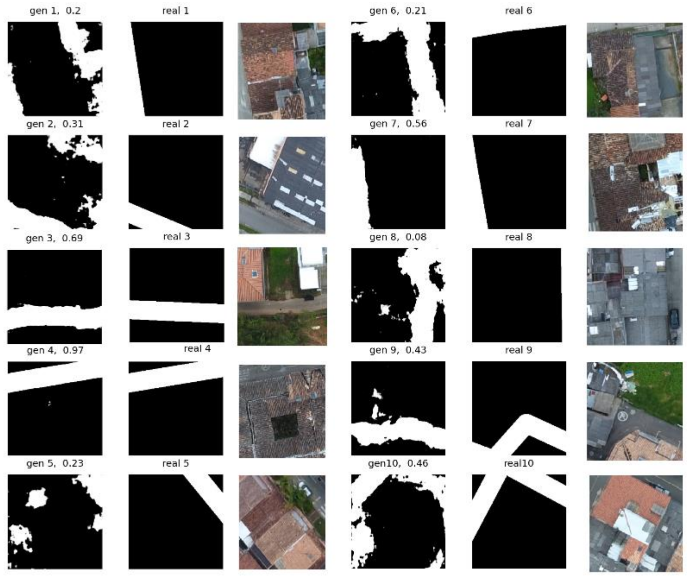
Model generalization is tested via inferencing roads using images from other settlement (La Ceja, Ant.) with a higher GSD (5.5 cm/px), these are different to the ones of the training dataset (El Retiro, Ant. GSD 7.09 cm/px). Also, model is tested on challenge situations like different type of occlusions, and positive class scarcity. Figure 5.27 shows examples when model is used on a different settlement, with other light conditions and road characteristics, and when tree and building occlusions are present. Table 5.8 summarizes model generalization results.



(a)



(b)



(c)

Figure 5.27. Image to Mask Translation Model Generalization. (a) Model generalization over different settlement, although resultant masks are relatively good, mIoU metric is around 60%, this is because the differences in the GSD of the orthomosaics which cause a difference in ground truth, (b) Tree Occlusions, (c) Building Occlusions. Model sorted out well tree occlusions. However, for road occlusion or when inferring

over imbalance images, model behaves deficiently, generating false positives especially in zones with similar spectral response.

Table 5.8. Model Generalization

Case	mIoU
Inferring on different settlements	0.595
Tree occlusions	0.929
Building occlusions	0.428

5.2.12 Color Encoding-Decoding for Attribute Extraction

Figure 5.28 shows the results of an example of the model trained using a dataset with two road types, paved, unpaved, and background. They are encoded using color, red (1), green (2), and black (0) respectively.

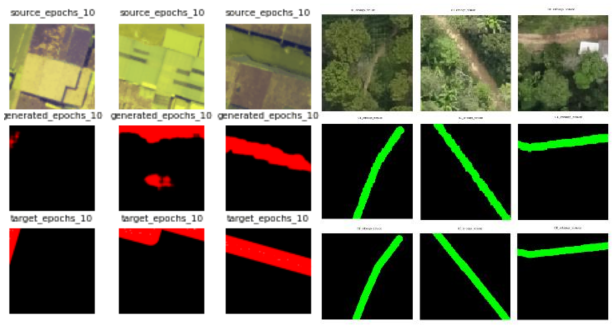
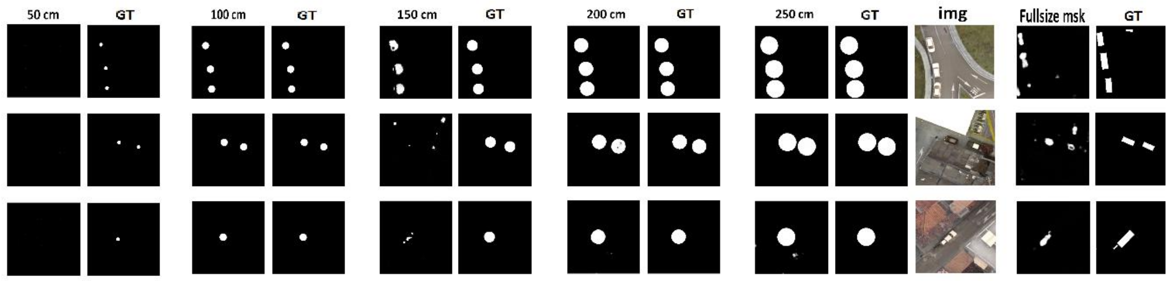


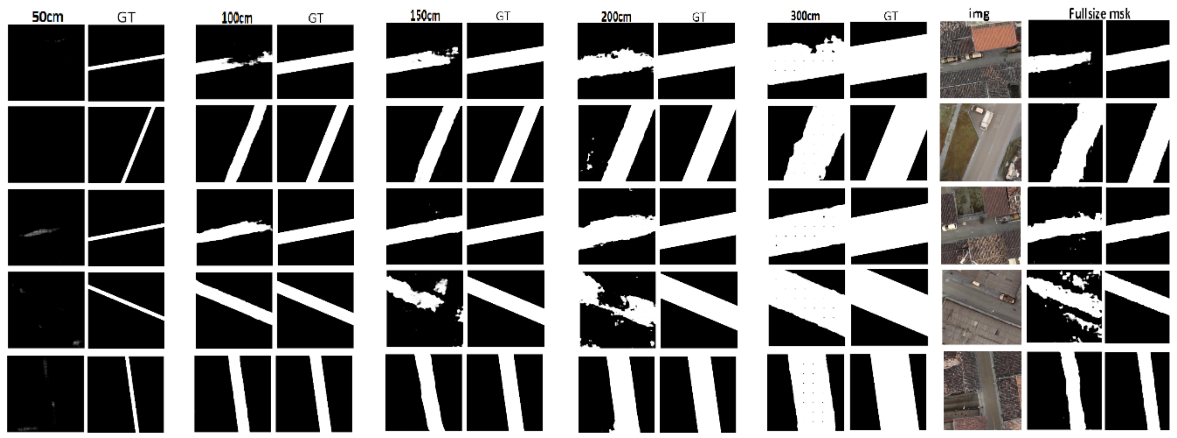
Figure 5.28. Encoding color for automatic attribute extraction. Model can generate and discriminate road types with an mIoU=0.798.

5.2.13 Results on point, line and polygon objects

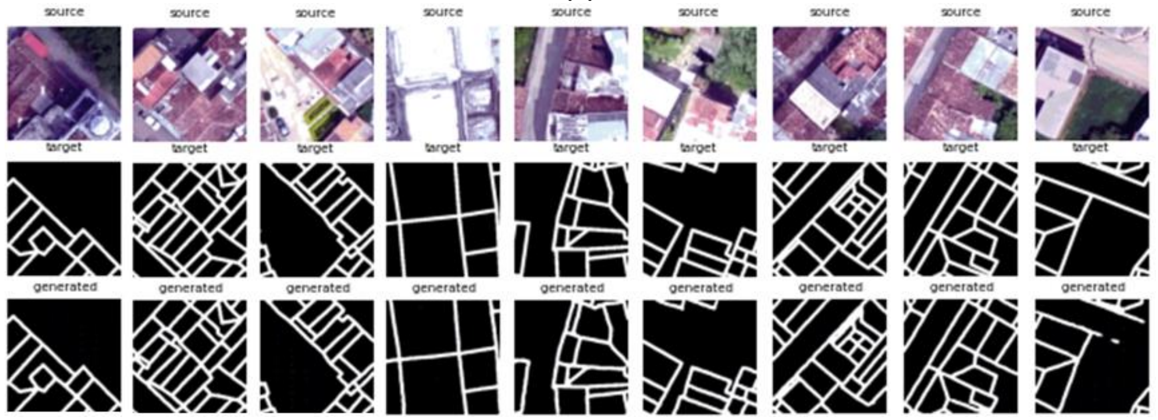
Figure 5.29 illustrates results on different object geometries.



(a)



(b)



(c)

Figure 5.29. Results on different sizes of masks for vehicles, roads and buildings. (a) vehicles, (b) roads, (c) buildings.

5.3 Conclusions and Future Work

Qualitative results show that Image to Mask Translation Models is agnostic to geometry of objects of interest in source imagery, and type or scale of input imagery. Results also show that conditional translation models work better than unconditional in terms of the need of a smaller number of examples, less training time, and number of epochs in the geography realm. Meanwhile, semantic segmentation and attention modules generated more round-shaped masks compared to image to mask translation approach. However, translation models are not completely free of producing round masks results for anthropic objects, like roads and buildings.

Although generation of masks started with around 250 image-mask pairs and increasing the number of training examples improved the results, it also increased the training time in Google Collab. It started in around 3 hours for 250 examples and went up to 14 hours for 2000 examples. Fortunately, it decreased almost in five times using a batch size of ten nor one as in the beginning. It slightly raised the mIoU metric as well. Other hyperparameters of the model like number of epochs, does not necessarily improved the results when moved above 100.

Transfer learning and multi-scale training did not seem to improve the mask translation results, on the other hand, imbalance datasets are indeed a ubiquitous complex problem that affected results in quality, number of examples and epochs needed for the model to learn. Imbalance checking by every image-mask pair, not by the whole orthomosaic, helped in obtaining results closer to the ground truth. Geometric augmentation, especially the overlapping method, improved results more than the spectral augmentation. Image fusion performed by the inclusion of height and VARI index made up the learning process less variable. Inclusion of height data reduced the variance of the generator and discriminator loss.

Model generalizes from similar region roads style and composition, nonetheless, mIoU metric did not measured well when masks are discontinuous or when orthomosaics with different GSD were used. Model also worked fine on tree occlusion, but mIoU highly decreased when challenged to situations of scarce or occlusion of roads.

The use of other color spaces different to RGB, such as HSV, is suggested as an additional work to test if image to mask translation models would perform better.

Chapter 6

6 A Post-processing Method

GIS vectorization methods rely on uniform width, and continuity of the input masks, to produce point, line, and polygons objects. While semantic segmentation methods employ full-size masks, that also contribute to have less imbalance classes. Vectorization of full-size, irregular width or discontinuous masks commonly create an erroneous vector representation, for instance, road centerline or building footprints. So, the objective becomes the obtention of continuous and uniform width translated masks from images, since they produce a better vector representation. Experiments with our model-integrated concepts of Primitive Masks and Double Image to Mask Translation for the enhancement of masks are presented next.

6.1 Primitive Masks

A Primitive Mask is the simplest and uniform-width target mask for objects of a certain geometry present in source images, that is obtained in a semi-automatic way using a chosen buffer distance. It is produced to have the less possible pixels from other classes and it is imbalance checked to a threshold t. To obtain a raster mask for an object from vector ground truth in a semi-automatic way, a GIS user may choose a value for the buffer distance. However, setting an optimum value is not easy, since for instance, a vehicle can be geo-located at the central point of a ball of radius 10 cm, 50 cm, 100 cm, etc. A road can be depicted by its centerline unitary pixels, or by a X meters-width mask. Choosing a small X value will cause an imbalance class, on the other hand, picking a large buffer distance would create a high variability distribution of misclassified pixels. So, an equilibrium between width and gaussian distribution variance is searched via an optimization process as shown in a future section. So, the purpose of using primitive masks is to find a gaussian-like statistical distribution of pixel values of the specific objects of interest that have enough number of pixels. This will improve the uniform shape of resultant masks and then the geometry and geographic coordinates of extracted objects.

A primitive mask can be colored to represent multi-classes or attributes like type of vehicle, road class, speed, roof material, and many others. Figure 6.1. illustrates a primitive mask for roads.

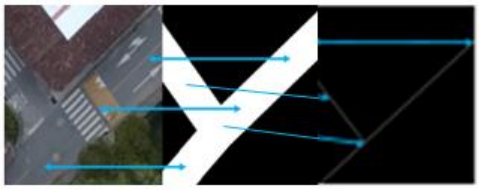
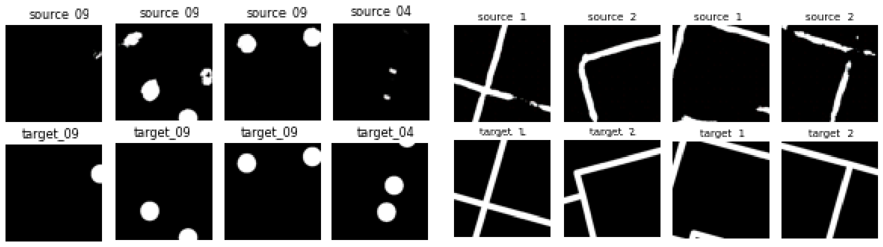


Figure 6.1. Example of a Primitive Mask for Roads. It consists of setting an optimum value for a buffer distance.

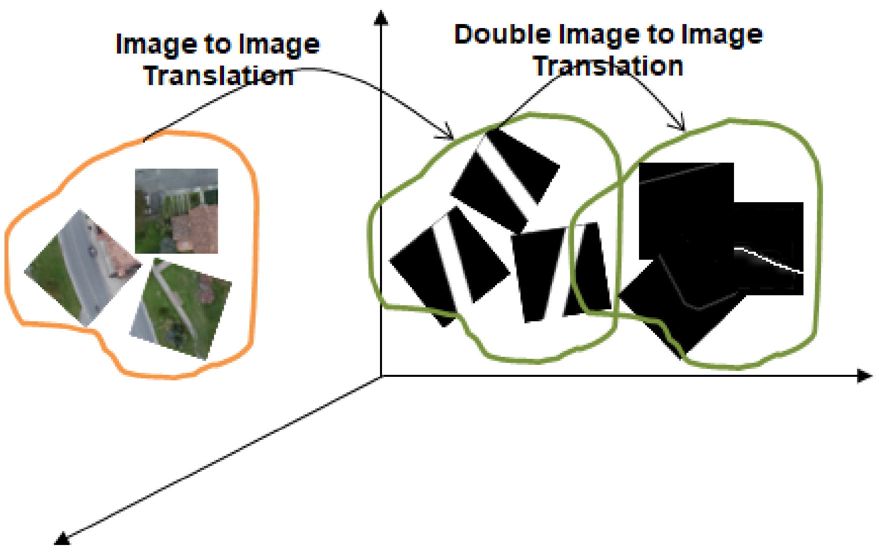
In certain cases, primitive masks may be imbalance, so their use by any deep learning model, including a generative model could be challenging. To deal with this problem, we propose the Double Conditional Image to Mask Translation (DCIT) and the double paired datasets as explained next.

6.2 Double Image to Mask Translation and Double Paired Datasets

As mentioned earlier, post-processing of resultant masks from deep learning models is overloaded with heuristics and hand engineering, depending on the type of masks obtained by a model, their quality, and the object to extract (Ng & Hofmann, 2018). Conditional translation of pixels between two semantic latent spaces may not produce perfect results in the case of highly distant spaces, or diverse, scarce and imbalance datasets, which produces spurious masks. To clean the resultant masks, we propose The Double Conditional Image Translation (DCIT), which consists in the use of a second generative model for the translation between spurious source masks and corrected target masks. The second generative model is trained on a double paired dataset as exhibited in Figure 6.2.



(a)



(b)

Figure 6.2. Double Pair Dataset Example and Double conditional image translation (DCIT). (a) Double Paired Dataset, it consists of pairs of (spurious masks – corrected masks), in which spurious are generated by a deep generative model. (b) DCIT, as it is shown in illustration, the first translation generated pixels in a full-size masks latent space, then the DCIT generates pixels in a primitive masks latent space.

In theory, the translation from full-size, irregular or discontinuous masks to corrected masks using DCIT is feasible, since the semantic distance between binary masks is smaller than the existing between the RGB images and binary masks. Figure 6.3. illustrates the training diagram for the DCIT.

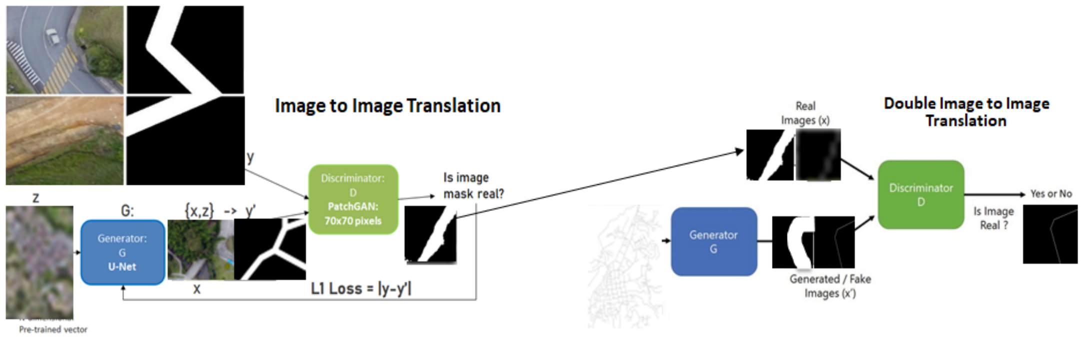


Figure 6.3. DCIT training diagram. First, a Pix2Pix translates source images to full or regular size masks, a second Pix2Pix architecture performs image translation from regular masks with errors, to primitive masks. A complete example would be, first a translation between source images (orthomosaic chips) to target (full-size masks) using a dataset with an imbalance ratio > 20% for the best results. Resultant masks (irregular and discontinuous) will be taken by a second translation, that translates them to corrected masks. Masks produced by DCIT are in principle easier to vectorize and create better point, line, or polygon geometries.

Figure 6.4. shows examples of full-size training mask, the resultant irregular and discontinuous masks obtained by deep learning models, and primitive masks used to clean masks using a DCIT. An example of a double paired dataset to train a DCIT model to correct roads is presented too.

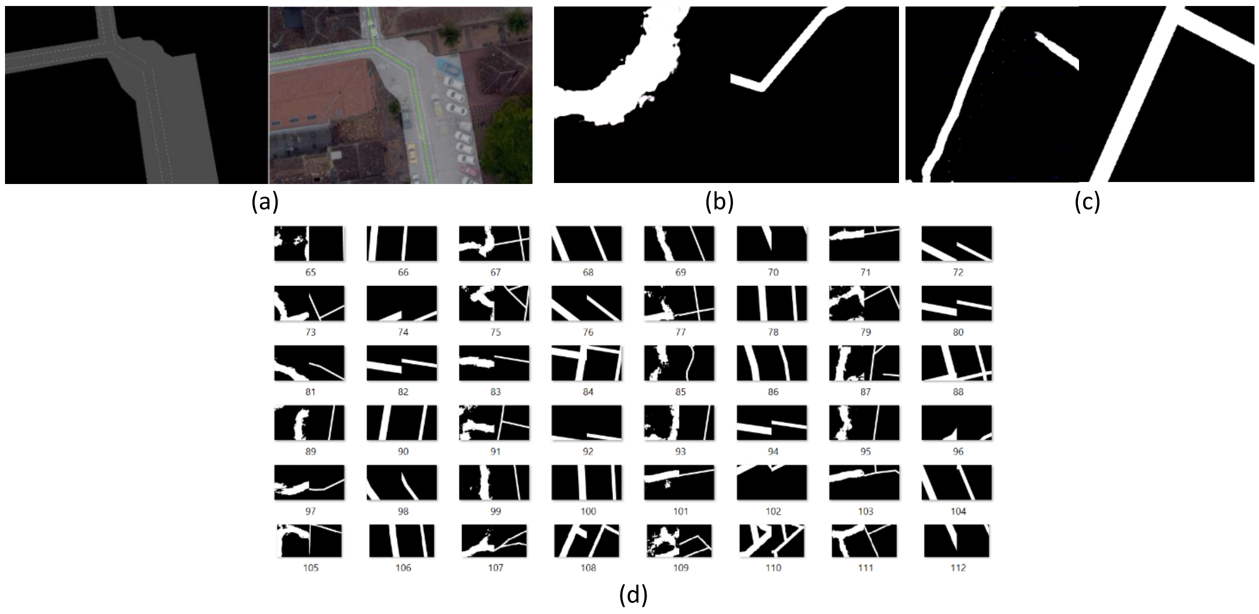


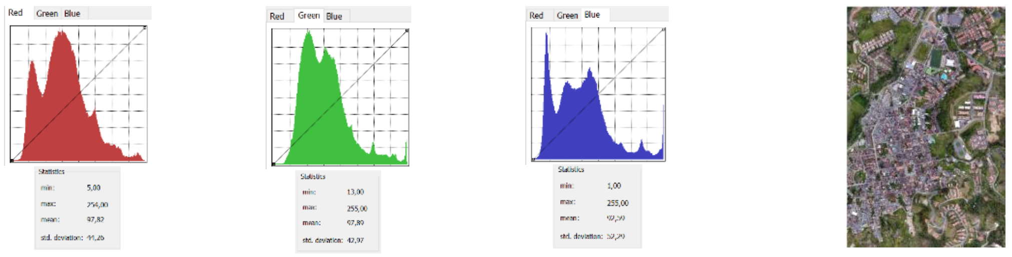
Figure 6.4. Examples of masks and double image translation road dataset. (a) Full-size and primitive masks, (b) Irregular mask, (c) Discontinuous mask, (d) DCIT dataset for roads. Picture (a) shows that full-size masks are manually digitized and difficult to acquire, primitive mask on the other hand are equal width and produced automatically by setting a buffer distance parameter, they produce a road centerline (green line) after vectorization. Pictures (b) and (d) show examples of resultant mask by deep learning models, our DCIT approach uses a dataset of irregular and discontinuous masks paired with primitive masks to be trained and output clean masks as shown in (d).

6.3 Experiments and Results

As found in previous chapter, imbalance makes a great impact on mask generation, obtaining the best results when at least 20% of pixels belong to the positive class. So, using equal-size masks with a wrong width will cause a low performance model. Next section describes a method to obtain primitive masks.

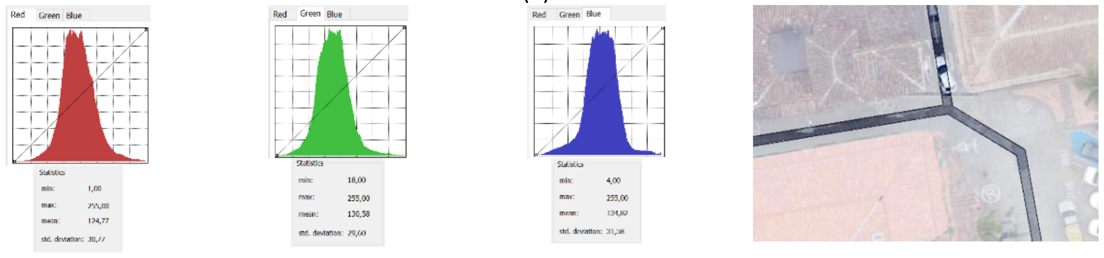
6.3.1 Primitive Masks and Buffer Distance

Primitive masks are equal-size masks obtained automatically to semi-automatically using a buffer distance parameter. This is one of the advantages over the traditional and manual digitized segmentation masks. However, a small buffer distance will produce low variance gaussian distribution but imbalance dataset. On the other hand, large buffer distances will produce high variance gaussian distributions but a larger value of the imbalance ratios. Primitive masks are obtained when the optimum value of buffer distance is set. Figure 6.5. shows how buffer distance parameter affects the distribution of pixels RGB values of roads.



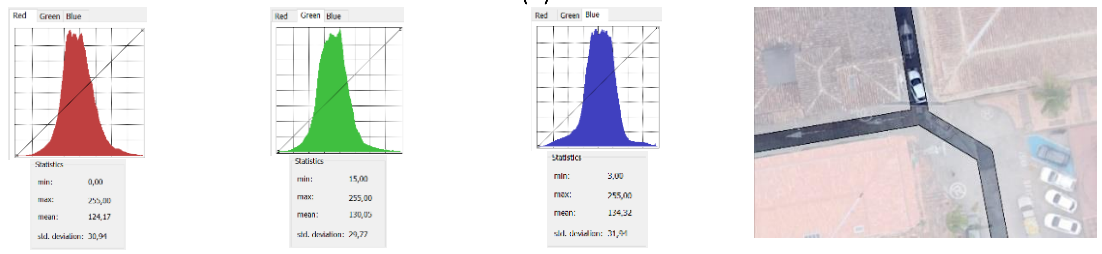
	Red	Green	Blue
min:	5,00	13,00	1,00
max:	254,00	255,00	255,00
mean:	97,82	97,89	92,59
std. deviation:	44,26	42,97	52,29

(a)



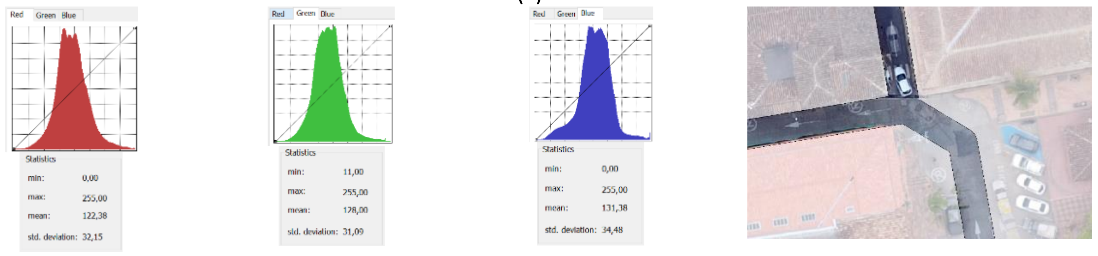
	Red	Green	Blue
min:	1,00	18,00	4,00
max:	255,00	255,00	255,00
mean:	124,77	130,58	134,82
std. deviation:	30,77	29,60	31,58

(b)



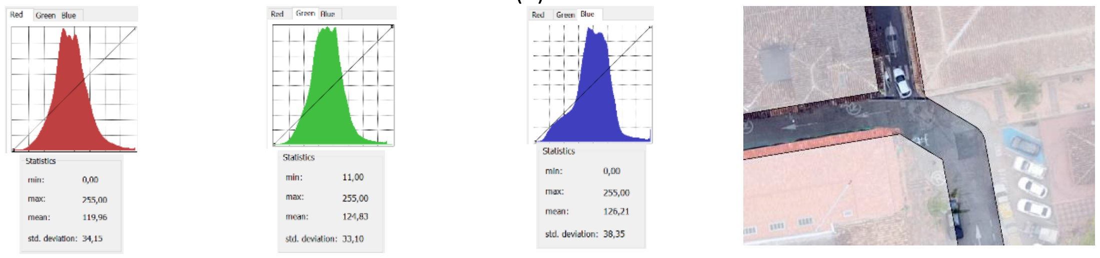
	Red	Green	Blue
min:	0,00	15,00	3,00
max:	255,00	255,00	255,00
mean:	124,17	130,05	134,32
std. deviation:	30,94	29,77	31,94

(c)



	Red	Green	Blue
min:	0,00	11,00	0,00
max:	255,00	255,00	255,00
mean:	122,38	128,00	131,38
std. deviation:	32,15	31,09	34,48

(d)



	Red	Green	Blue
min:	0,00	11,00	0,00
max:	255,00	255,00	255,00
mean:	119,96	124,83	126,21
std. deviation:	34,15	33,10	38,35

(e)

Figure 6.5. Primitive mask and buffer distance parameter. (a) RGB distribution for the whole orthomosaic, (b) primitive mask with 50cm buffer distance, imbalance ratio =2.13%, (c) primitive mask with 1m buffer distance, imbalance ratio =4.35%, (d) primitive mask with 2m buffer distance, imbalance ratio=7.86, (e) primitive mask with 3m buffer distance, imbalance ratio=13.56. The same experiment was performed over other 10 orthomosaics, and the values are similar due perhaps to the fact that roads have a similar size and materials in the mapping zone.

Graph of standard deviation of pixel values per band vs buffer distance of Figure 6.6. illustrates a method to find the optimum buffer distance, this is, to obtain primitive masks for the roads and the results.

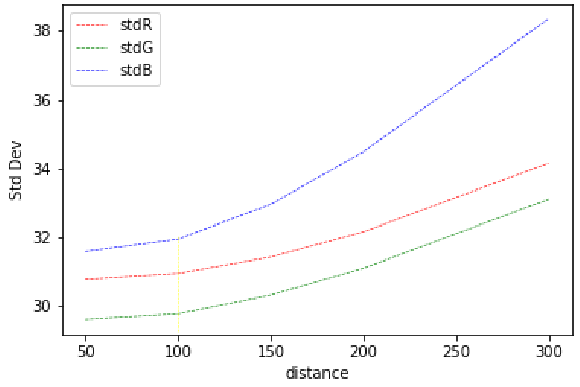
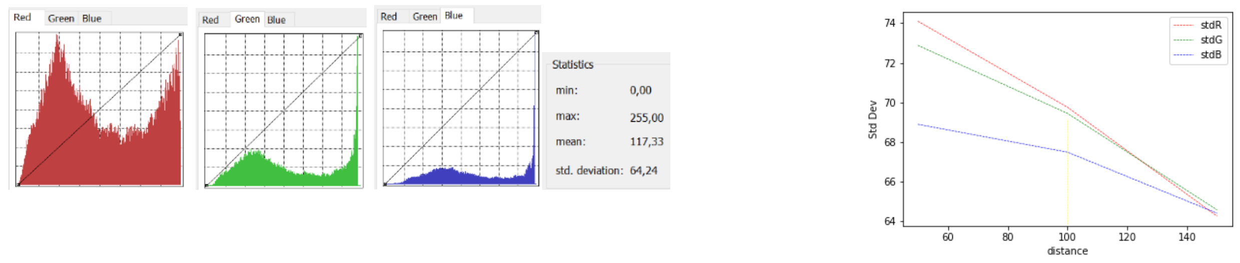


Figure 6.6. Primitive masks and buffer distance parameter. For a buffer distance of 100 cm (yellow vertical line), there is practically no change in standard deviation of RGB values distribution, which indicates that 1 meter is the best gaussian distribution of values, and so this is the buffer distance to set for obtaining primitive masks. Blue graph seems to confirm why replacing blue channel by DSM values does not seem to work as good as expected, and instead summing DSM values to every RGB channel is a better option.

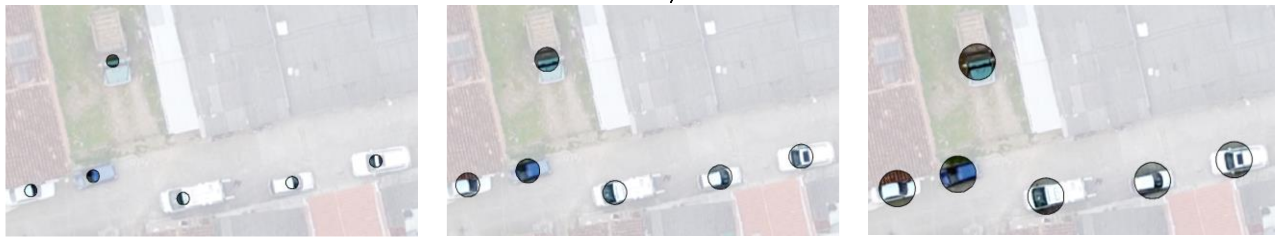
Figure 6.7. shows the results when obtaining primitive masks for point objects, in this case, vehicles.



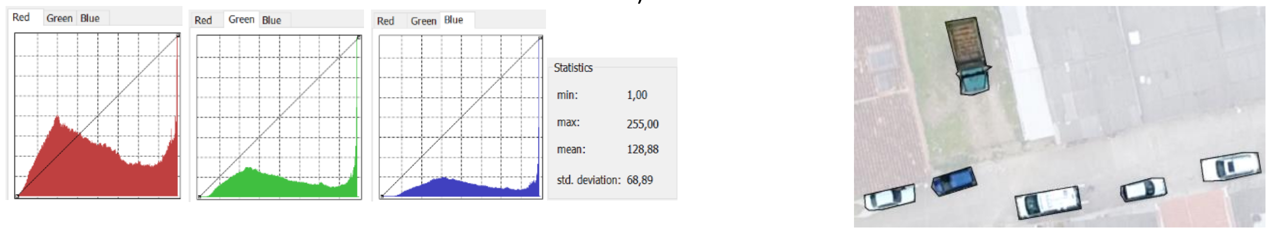
Statistics

min:	0,00
max:	255,00
mean:	117,33
std. deviation:	64,24

a)



b)



Statistics

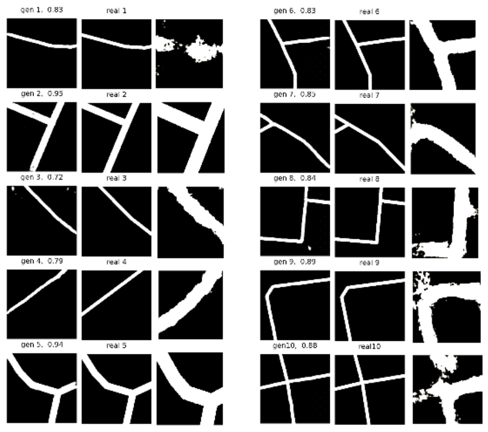
min:	1,00
max:	255,00
mean:	128,88
std. deviation:	68,89

(c)

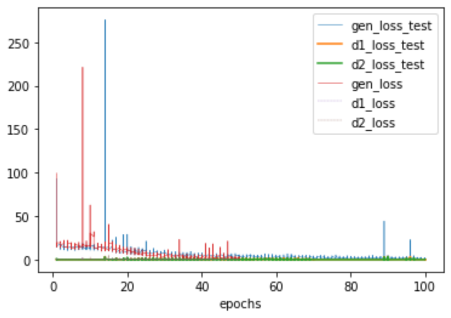
Figure 6.7. Primitive masks for point objects. (a) Gaussian distributions (50cm, 100 and 150cm) are not perfectly shaped because vehicles are not uniformly colored. This also explains the higher standard deviation compared to the previous road masks example. The graph of standard deviation of RGB pixel values vs distance (upper right image) shows that 100 cm seems to be the appropriate buffer distance for producing primitive masks. (b) In GIS, vehicles are modeled as points, so main feature is position and the attribute type, image in the middle shows masks with 50 cm (left), 100 cm (middle), and 150 cm (right) buffer distance of point vehicle locations. (c) Image in the bottom illustrates full size semantic segmentation masks for vehicles. However, although they consist of more pixels, the RBG distribution is very similar to the primitive masks exposed in image (a).

6.3.2 Double Image to Mask Translation for Masks and Vector Improvement

We applied double image to mask translation to the road dataset to clean resultant masks of the generative model. Figure 6.8. shows the results in terms of model performance and learning graphs of testing vs training road dataset.



(a)



(b)

Figure 6.8. DCIT model performance and learning graphs. (a) DCIT Model performance mIoU=0.892, As can be observed model cleans effectively discontinuous and irregular masks, right is input spurious mask, center is the GT and left is the obtained mask by DCIT, (b) Discriminator and generator losses of the training vs the testing dataset.

Figure 6.9. exhibit the improvement of resultant masks by the application on DCIT in the case of a dataset formed only by discontinuous and irregular masks.

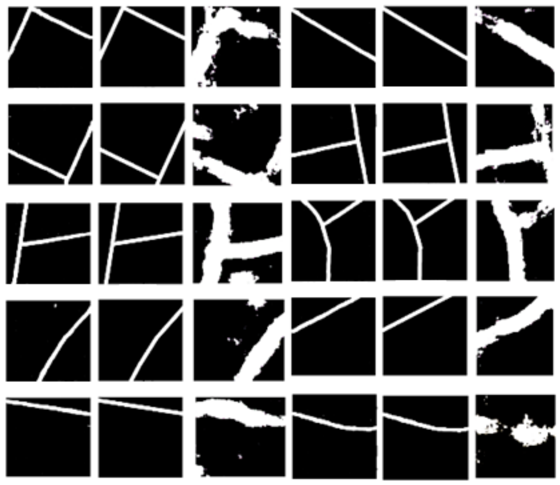
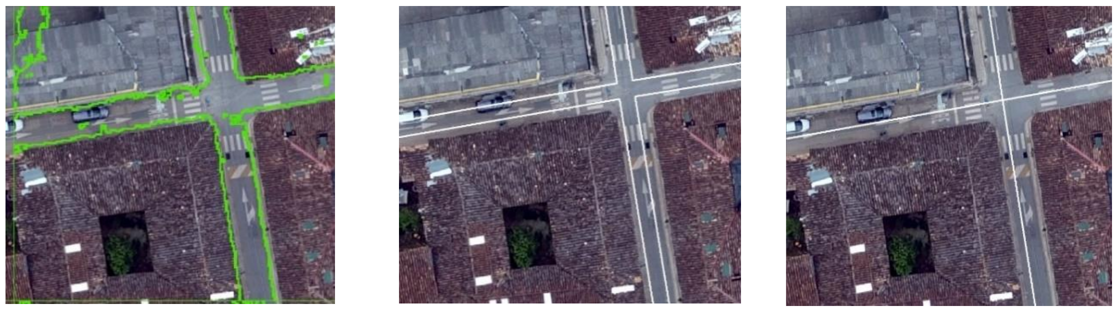


Figure 6.9. Improvement of discontinuous and irregular masks by the application of DCIT. Center mask is the GT, right is the input image, and left is the generated by DCIT. When DCIT is applied only on spurious masks, it effectively tackles discontinuity (false negative pixels) and irregularity (clusters of false positive pixels) in masks. However, mIoU decreases from 0.892 obtained for a mixed dataset to 0.802 for only spurious.

Finally, Figure 6.10 illustrates vectorization results on an irregular mask obtained by the generative model versus those after DCIT is applied.



(a) (b) (c)

Figure 6.10. Post-processing. (a) Vector of a road irregular mask obtained by a semantic segmentation model using a U-Net, (b) Vector representation of a primitive mask generated by our approach, (c) Vectorization into a centerline of the previous primitive mask.

6.4 Conclusions and Future Work

The concepts of primitive masks and double image translation are introduced and developed. Primitive masks for training image translation and double image translation models bring several benefits. It starts with allowing a more uniform gauss distribution of RGB pixel values that improves model training and performance. In the second place, it supports the enhancing of masks, correcting discontinuity and irregularity of masks using a DCIT integrated model. Primitive masks depend on the optimization of the standard deviation as a function of the buffer distance, this process can be automated following the method proposed in the text for a road dataset. Primitive masks obtained were 1 meter width a standard deviation of around 30 for the orthomosaic of a settlement. However, the value was a constant for the majority of the other ten settlements acquired with drone. A double paired dataset for roads is introduced for training a DCIT model. DCIT is an integrated model to perform image to mask and then mask to primitive mask translation. It is used as a measure to clean masks that are translated deficiently with an image to mask translation model. DCIT showed good results in a road-testing dataset for solving discontinuities, generating continuous masks, and making uniform masks when initial model generated irregular masks. The results obtained by the DCIT model are vectorized and visually compared with the resultant mask obtained by a U-Net segmentation model.

Future work is proposed in reducing the size of the DCIT network, it has the same size as the first one, but we consider that a smaller network might be work better since the distance between the irregular and discontinue masks latent spaces is closer than the image to mask latent spaces. Second would be to build a web application that integrates the whole approach proposed here and brings people an opportunity to play with pixel distributions, imbalance, primitive masks and double image translation to solve local mapping problems.

Chapter 7

7 Validation of the Proposed Methodology

Vehicle, road, and building localization from overhead imagery has often been treated as an image segmentation problem, i.e., identifying which pixels in an image belong to which class. However, metrics like F1-Score and mIoU do not consider mask discontinuities, and comparison to Vector GT. These two aspects make the difference of a metric to be used in GIS vector layer obtention. The proposed evaluation metric is discussed below.

7.1 Proposed Performance Metric

Road predictions are affected by occlusions and shadows, as stated in (Van Etten et al., 2019), legacy metrics as IoU does not optimally evaluate road continuity (discontinuities), and errors in width are less penalized than brief breaks in inferred roads. These situations led authors in (Van Etten et al., 2019) to propose metric based upon Dijkstra's shortest path algorithm. The Average Path Length Similarity (APLS) sums the differences in optimal paths between GeoJSON vector ground truth and GeoJSON proposal graphs. Missing paths in the graph are assigned the maximum proportional difference of 1.0. The APLS metric scales from 0 (poor) to 1 (perfect), Equation 7.1.

$$\mathrm{APLS} = 1 - \frac{1}{N}\sum min\left\{1, \frac{|\,\mathrm{L(a,b)} - \mathrm{L(a',b')}\,|}{\mathrm{L(a,b)}}\right\} \qquad 7.1$$

Where N = number of unique paths, while $\mathrm{L(a,b)}$ = length of path(a, b). The sum is taken over all possible source (a) and target (b) nodes in the ground truth graph. The node $\mathrm{a'}$ denotes the node in the proposal graph closest to the location of ground truth node a. If path $(\mathrm{a'}, \mathrm{b'})$ does not exist, the maximum contribution of 1.0 is used, thereby ensuring that missing routes are highly penalized. Equation 7.1. is easy to apply if the proposal graph G' has nodes coincident with the ground truth graph G. In practice, however, proposals will not align perfectly with ground truth graphs, so a number of post-processing snapping, and additional considerations should be done to calculate the APLS metric, which at the same time is only applicable to roads.

Additional to the flaws of the IoU reported by (Van Etten et al., 2019), Figure 5.6. showed that when evaluating model generalization with images of a different settlement, mIoU decreased although resultant masks were qualitatively good. This situation was caused by a difference in GSDs of the orthomosaics used for training and for inferencing, which resulted in a different size between the GT and the resultant masks. This cannot be corrected, and then, evaluating mask geometry after vectorized is particularly meaningful, because a simple and accurate vector data representation is crucial to the map production. For these reasons, we propose the Average Geometry Similarity (AGS), as APLS, a metric based on the ground truth vector objects, calculating a difference in number, distance, and area between vector objects generated by model against the original vectors, but also the time taken to generate a number of objects, length, or area. These two values are calculated at least within n (three as the minimum) different areas of interest (AOIs), 100m, 200m, and 400m. This metric better reflects not only the fidelity of geometry and thus the required manual editing workload when converting a model inference mask to real map products. Equations 7.2., 7.3, and 7.4 are designed for points (ex. vehicles), lines (ex. roads), and polygons (ex. buildings) respectively.

$$\text{AGS\_Points} = \frac{1}{n}\sum_1^n\left(1 - \min\left\{1, \frac{|\#\text{V(a,b)} - \#V'\text{(a,b)}|}{\#\text{V(a,b)}}\right\}\right), \text{ points/s} \quad 7.2$$

$$\text{AGS\_Lines} = \frac{1}{n}\sum_1^n\left(1 - \min\left\{1, \frac{|\text{L(a,b)} - L'\text{(a,b)}|}{\text{L(a,b)}}\right\}\right), \text{ m/s} \quad 7.3$$

$$\text{AGS\_Polygons} = \frac{1}{n}\sum_1^n\left(1 - \min\left\{1, \frac{|\text{A(a,b)} - A'\text{(a,b)}|}{\text{A(a,b)}}\right\}\right), \text{ m2/s} \quad 7.4$$

Where, $\#\text{V}$ and $\#V'$ are the number of, for instance vehicles, within an area defined by points a and b, for both the ground truth, and model respectively. L(a, b) and $L'\text{(a, b)}$ are the length of roads for ground truth and model within points (a, b), and A(a, b), $A'\text{(a, b)}$ are the area of buildings for the GT and the model within points (a, b). Proposed metric scales from 0 (poor) to 1 (perfect). The time would serve as a benchmark in the future, (H. L. Yang et al., 2018) obtained building binary masks, using a SegNet model, with an average processing time of less than one minute for an area of ~56 km2. However, they don´t mention the hardware employed. AGS is non dimensional, the second term is meter of automatic generation per second. AGS is used only for vector objects, so it is not applicable to imagery. Figure 7.1 illustrates the proposed AGS metric for the roads example. Center point of increasing area is arbitrary, but it is suggested to be in the middle of the orthomosaic of mapping interest. A script to obtain the three squares can be found in Annex 1.

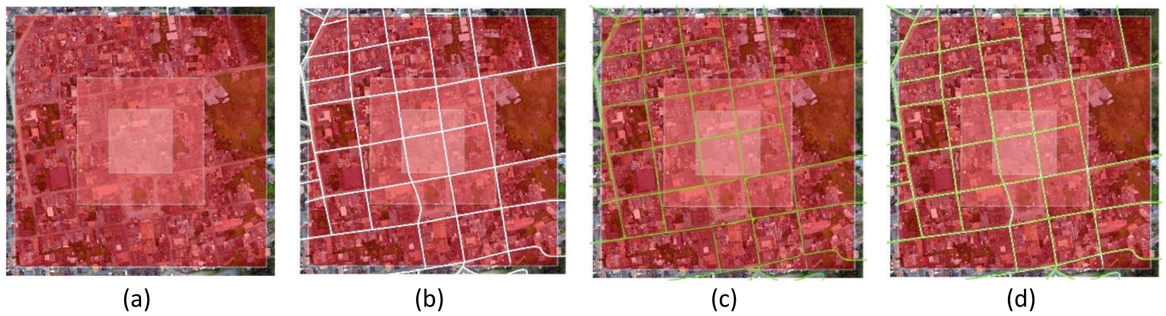
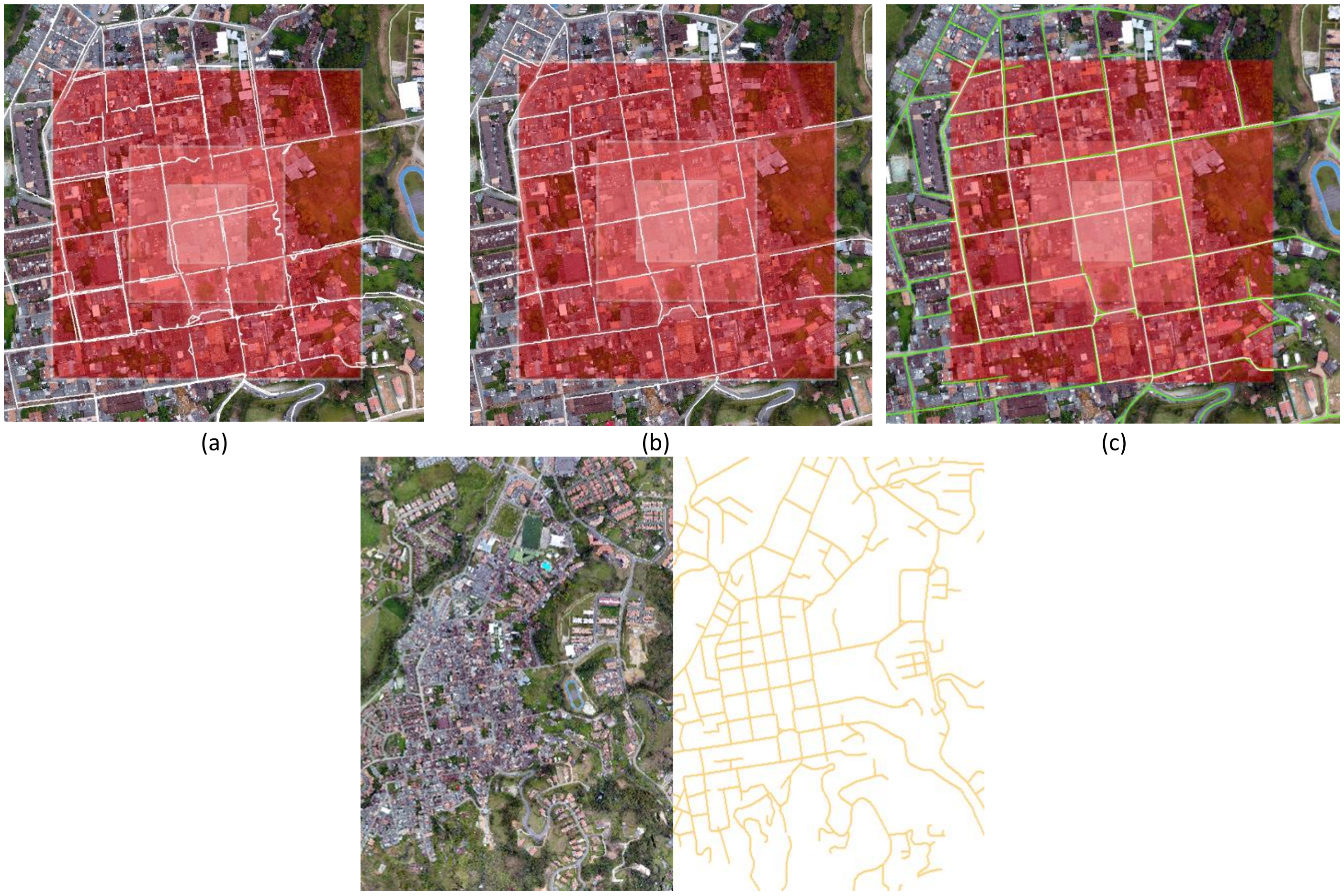


Figure 7.1. AGS Proposed Methodology Performance Metric. (a) 100m, 200m, and 400m AOIs, (b) GT roads, (c) Generated roads, (d) GT and generated roads.

7.2 Experiments and Results

AGS Metric was used to evaluate performance of the method using different generative model configurations. AGS_Lines for roads of settlement El Retiro, (Ant.) was calculated. Python script for calculating AGS Metric for Lines can be found in Annex 1. Figure 7.2. shows resultant road network after vectorization and Table 7.1. summarizes the AGS results.



(d)

Figure 7.2. Application of AGS Metric. (a) Image to mask translation model and vectorization without primitive masks, (b) Image to mask translation model with primitive masks and vectorization, (c) Model including double image to mask translation and vectorization, green line is the GT, (d) Complete area of inference.

Table 7.1 Results of estimating AGS_Lines for metric roads vectorization

mosaic	Application of AGS Metric - Roads	AGS_Lines
El Retiro, (Ant.)	Image to mask translation model and vectorization without primitive masks	0.801 at 12.87 m/s
	Image to mask translation model with primitive masks and vectorization	0.903 at 12.39 m/s
	Model including double image to mask translation and vectorization	0.940 at 12.03 m/s

7.3 Conclusions and Future Work

A metric called AGS is proposed based on the weakness of the mIoU for the mapping field and related work. At least roads of 4 settlements were mapped automatically and AGS metric evaluated. Results show that proposed metric is useful to measure the difference between maps created by a person and automatic mapping. This metric, accompanied of the produced datasets, may lead to a benchmark for standardizing map automation. The developed methodology created urban road network of El Retiro (Ant.) in 36 minutes using a windows PC CPU 2.8 HGz, core i7, with 4 Gb RAM. We obtained AGS_Lines of 0.94 and 12 m/sec, and La Ceja (Ant.) in 16 minutes with an AGS_Lines of 0.96 at 15 m/sec.

Space for future work is on the development of a metric that has into account differences in coordinates, also there is an opportunity to build a cloud platform that integrates the three methods documented in this thesis.

Chapter 8

8 Conclusions

This thesis presents a pipeline for extraction of point, line, and polygon vector features from orthomosaics via a generative model. It improves data creation and its effectiveness on model performance to produce better quality masks in terms of continuity and uniform size. These are the most relevant aspects in the vector layers production. Our pipeline integrates post-processing in three methods to pass from drone imagery to vector layers.

In Chapter 3, we showed the three components in which the pipeline is based on, and the specific details for everyone. Different data transformations that should be performed to improve model training and behavior are highlighted. Image to mask translation model is described to replace the traditional segmentation models in extracting geographic objects. Description of introduced concepts, primitive masks and double image translation, that help integrating post-processing of masks into the generative model to create better vector objects.

Chapter 4 presents the data pairing workflow, buffer distance proved to be the key parameter for allowing the method to produce datasets for point and line objects in a semi-automatic way compared to alternative manual approaches. The data imbalance check is keen to produce less imbalance image-mask pairs, with a suggested imbalance ratio threshold above 10% to guarantee generative model to learn. The proposed method developed an easy way to augment dataset with height values, and fusion of VARI Index into RGB images that may be discriminative for some objects. A vehicles dataset as an expression of point objects, and a road dataset as an instance of line objects dataset, are examples of the application of this method.

Chapter 5 showed that conditional image to mask translation showed better qualitative and quantitative results in production of cleaner and uniform masks compared to the unconditional or the semantic segmentation approach. Also, it exhibited a higher mIoU value even including state of the art attention modules. Generation of point, line or polygon masks, conditionally and supervised, could solve some of the problems of semantic segmentation commonly described by many scientists. Model hyperparameters like batch size equal to 10 and 100 epochs showed the best training results. Around 1000 training examples of NRG-DSM paired images with a size of 512x512 px exhibited the best model performance for production of road masks. This overcomes the performance of VARI index and the combination of VARI with the DSM.

Chapter 6 describes how to obtain primitive masks with the aim of producing gaussian-like distribution of masks that guarantees convergence of chosen generative model. Proposed graph of standard deviation vs mask buffer distance to obtain a primitive mask proved that the best size for vehicle and road datasets was 1 meter for orthomosaics of different settlements. The double image to mask translation also introduced in this chapter, solved the discontinuous, and irregular width masks obtained by semantic segmentation and previous generative models.

Finally, in chapter 7 the proposed AGS metric measures performance of the 3-method pipeline to extract vector layers, comparing objects automatically obtained vs man-made. Results of AGS show that double image translation enhanced vectorization in the urban road network of the settlement El Retiro, which is verified by a small value of AGS when model did not include it.

Recommendation for future research may be centered on the mechanisms for directly encoding geographic objects coordinates to produce vector outputs using geometric deep learning. There is also room to fine-tune the double image to mask translation model in a way that can be integrated in other deep learning architectures for improving resultant masks. More research is needed to establish how double image translation may replace all or part of the methods commonly used to postprocess masks and produce vector objects.

Bibliography

Abdollahi, A., Pradhan, B., & Alamri, A. (2021). RoadVecNet: A new approach for simultaneous road network segmentation and vectorization from aerial and google earth imagery in a complex urban set-up. *GIScience & Remote Sensing*, *58*(7), 1151–1174. https://doi.org/10.1080/15481603.2021.1972713

Abdollahi, A., Pradhan, B., & Shukla, N. (2019). Extraction of road features from UAV images using a novel level set segmentation approach. *International Journal of Urban Sciences*. https://doi.org/10.1080/12265934.2019.1596040

Abdollahi, A., Pradhan, B., Shukla, N., Chakraborty, S., & Alamri, A. (2020). Deep Learning Approaches Applied to Remote Sensing Datasets for Road Extraction: A State-Of-The-Art Review. *Remote Sensing*, *12*(9), 1444. https://doi.org/10.3390/rs12091444

Aldana Rodriguez, D., Ávila Granados, D. L., & Villalba-Vidales, J. A. (2021). Use of Unmanned Aircraft Systems for Bridge Inspection: A Review. *DYNA*, *88*(217), 32–41. https://doi.org/10.15446/dyna.v88n217.91879

Al-Najjar, H. A. H., Kalantar, B., Pradhan, B., Saeidi, V., Halin, A. A., Ueda, N., & Mansor, S. (2019). Land Cover Classification from fused DSM and UAV Images Using Convolutional Neural Networks. *Remote Sensing*, *11*(12), 1461. https://doi.org/10.3390/rs11121461

Avola, D., & Pannone, D. (2021). MAGI: Multistream Aerial Segmentation of Ground Images with Small-Scale Drones. *Drones*, *5*(4), 111. https://doi.org/10.3390/drones5040111

Badrinarayanan, V., Kendall, A., & Cipolla, R. (2017). SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *39*(12), 2481–2495. https://doi.org/10.1109/TPAMI.2016.2644615

Ballesteros, J. R., Sanchez-Torres, G., & Branch, J. W. (2021). Automatic road extraction in small urban areas of developing countries using drone imagery and Image Translation. *2021 2nd Sustainable Cities Latin America Conference (SCLA)*, 1–6. https://doi.org/10.1109/SCLA53004.2021.9540111

Ballesteros, J. R., Sanchez-Torres, G., & Branch-Bedoya, J. W. (2022). HAGDAVS: Height-Augmented Geo-Located Dataset for Detection and Semantic Segmentation of Vehicles in Drone Aerial Orthomosaics. *Data*, *7*(4), 50. https://doi.org/10.3390/data7040050

Batra, A., Singh, S., Pang, G., Basu, S., Jawahar, C. V., & Paluri, M. (2019). Improved Road Connectivity by Joint Learning of Orientation and Segmentation. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 10377–10385. https://doi.org/10.1109/CVPR.2019.01063

Bhatnagar, S., Gill, L., & Ghosh, B. (2020). Drone Image Segmentation Using Machine and Deep Learning for Mapping Raised Bog Vegetation Communities. *Remote Sensing*, *12*(16), 2602. https://doi.org/10.3390/rs12162602

Bisio, I., Haleem, H., Garibotto, C., Lavagetto, F., & Sciarrone, A. (2021). Performance Evaluation and Analysis of Drone-based Vehicle Detection Techniques From Deep Learning Perspective. *IEEE Internet of Things Journal*, 1–1. https://doi.org/10.1109/JIOT.2021.3128065

Blaga, B.-C.-Z., & Nedevschi, S. (2020). A Critical Evaluation of Aerial Datasets for Semantic Segmentation. *2020 IEEE 16th International Conference on Intelligent Computer Communication and Processing (ICCP)*, 353–360. https://doi.org/10.1109/ICCP51029.2020.9266169

Bolstad, P. (2016). *GIS fundamentals: A first text on geographic information systems : 5th ed.* Eider (PressMinnesota). http://repository.ntt.edu.vn/jspui/handle/298300331/2885

Brooks, C. (2017). Drone-Enabled Remote Sensing for Transportation Infrastructure Assessment. *INSPIRE-University Transportation Center Webinars*. https://scholarsmine.mst.edu/inspire_webinars/2

Bulatov, D., Häufel, G., & Pohl, M. (2016). VECTORIZATION OF ROAD DATA EXTRACTED FROM AERIAL AND UAV IMAGERY. *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, *XLI-B3*, 567–574. https://doi.org/10.5194/isprsarchives-XLI-B3-567-2016

Buslaev, A., Iglovikov, V. I., Khvedchenya, E., Parinov, A., Druzhinin, M., & Kalinin, A. A. (2020). Albumentations: Fast and Flexible Image Augmentations. *Information*, *11*(2), 125. https://doi.org/10.3390/info11020125

Butler, H., Daly, M., Doyle, A., Gillies, S., Schaub, T., & Hagen, S. (2016). *The GeoJSON Format* (Request for Comments RFC 7946). Internet Engineering Task Force. https://doi.org/10.17487/RFC7946

Cheng, B., Collins, M. D., Zhu, Y., Liu, T., Huang, T. S., Adam, H., & Chen, L.-C. (2020). *Panoptic-DeepLab: A Simple, Strong, and Fast Baseline for Bottom-Up Panoptic Segmentation*. 12475–12485. https://openaccess.thecvf.com/content_CVPR_2020/html/Cheng_Panoptic-DeepLab_A_Simple_Strong_and_Fast_Baseline_for_Bottom-Up_Panoptic_CVPR_2020_paper.html

ChiangYao-Yi, LeykStefan, & A, K. (2014). A Survey of Digital Map Processing Techniques. *ACM Computing Surveys (CSUR)*. https://doi.org/10.1145/2557423

Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., Franke, U., Roth, S., & Schiele, B. (2016). *The Cityscapes Dataset for Semantic Urban Scene Understanding*. 3213–3223. https://openaccess.thecvf.com/content_cvpr_2016/html/Cordts_The_Cityscapes_Dataset_CVPR_2016_paper.html

Crommelinck, S., Bennett, R., Gerke, M., Koeva, M. N., Yang, M. Y., & Vosselman, G. (2017). SLIC SUPERPIXELS FOR OBJECT DELINEATION FROM UAV DATA. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, *IV-2/W3*, 9–16. https://doi.org/10.5194/isprs-annals-IV-2-W3-9-2017

Crommelinck, S., Bennett, R., Gerke, M., Nex, F., Yang, M. Y., & Vosselman, G. (2016). Review of Automatic Feature Extraction from High-Resolution Optical Sensor Data for UAV-Based Cadastral Mapping. *Remote Sensing*, *8*(8), 689. https://doi.org/10.3390/rs8080689

Deigele, W., Brandmeier, M., & Straub, C. (2020). A Hierarchical Deep-Learning Approach for Rapid Windthrow Detection on PlanetScope and High-Resolution Aerial Image Data. *Remote Sensing*, *12*(13), 2121. https://doi.org/10.3390/rs12132121

Demir, I., Koperski, K., Lindenbaum, D., Pang, G., Huang, J., Basu, S., Hughes, F., Tuia, D., & Raskar, R. (2018). *DeepGlobe 2018: A Challenge to Parse the Earth Through Satellite Images*. 172–181. https://openaccess.thecvf.com/content_cvpr_2018_workshops/w4/html/Demir_DeepGlobe_2018_A_CVPR_2018_paper.html

DrivenData. *Open Cities AI Challenge: Segmenting Buildings for Disaster Resilience*. DrivenData. Retrieved June 22, 2022, from https://www.drivendata.org/competitions/60/building-segmentation-disaster-resilience/

Eng, L. S., Ismail, R., Hashim, W., & Baharum, A. (2019). The Use of VARI, GLI, and VIgreen Formulas in Detecting Vegetation In aerial Images. *International Journal of Technology*, *10*(7), 1385. https://doi.org/10.14716/ijtech.v10i7.3275

Fan, Q., Brown, L., & Smith, J. (2016). A closer look at Faster R-CNN for vehicle detection. *2016 IEEE Intelligent Vehicles Symposium (IV)*, 124–129. https://doi.org/10.1109/IVS.2016.7535375

Filin, O., Zapara, A., & Panchenko, S. (2018). Road Detection with EOSResUNet and Post Vectorizing Algorithm. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 201–2014. https://doi.org/10.1109/CVPRW.2018.00036

Gao, X., Sun, X., Zhang, Y., Yan, M., Xu, G., Sun, H., Jiao, J., & Fu, K. (2018). An End-to-End Neural Network for Road Extraction From Remote Sensing Imagery by Multiple Feature Pyramid Network. *IEEE Access*, *6*, 39401–39414. https://doi.org/10.1109/ACCESS.2018.2856088

Gerke, M., Rottensteiner, F., Wegner, J., & Sohn, G. (2014). *ISPRS Semantic Labeling Contest*. https://doi.org/10.13140/2.1.3570.9445

Girard, N., & Tarabalka, Y. (2018). End-to-End Learning of Polygons for Remote Sensing Image Classification. *IGARSS 2018 - 2018 IEEE International Geoscience and Remote Sensing Symposium*, 2083–2086. https://doi.org/10.1109/IGARSS.2018.8518116

GIS Mapping Software, Location Intelligence & Spatial Analytics | Esri. (n.d.). Retrieved August 1, 2022, from https://www.esri.com/en-us/home

Gitelson, A. A., Stark, R., Grits, U., Rundquist, D., Kaufman, Y., & Derry, D. (2002). Vegetation and soil lines in visible spectral space: A concept and technique for remote estimation of vegetation fraction. *International Journal of Remote Sensing*, *23*(13), 2537–2562. https://doi.org/10.1080/01431160110107806

Gong, Z., Xu, L., Tian, Z., Bao, J., & Ming, D. (2020). Road network extraction and vectorization of remote sensing images based on deep learning. *2020 IEEE 5th Information Technology and Mechatronics Engineering Conference (ITOEC)*, 303–307. https://doi.org/10.1109/ITOEC49072.2020.9141903

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., & Bengio, Y. (2014). Generative Adversarial Nets. *Advances in Neural Information Processing Systems*, *27*. https://proceedings.neurips.cc/paper/2014/hash/5ca3e9b122f61f8f06494c97b1afccf3-Abstract.html

Gülch, E. (n.d.). DIGITAL SYSTEMS FOR AUTOMATED CARTOGRAPHIC FEATURE EXTRACTION. *Part B*, 16.

Hartmann, S., Weinmann, M., Wessel, R., & Klein, R. (2017, May 30). *StreetGAN: Towards Road Network Synthesis with Generative Adversarial Networks*.

Heffels, M., & Vanschoren, J. (2020). *Aerial Imagery Pixel-level Segmentation*.

Introducing DroneDeploys Aerial Segmentation Benchmark | DroneDeploy. (n.d.). Retrieved December 31, 2021, from https://www.dronedeploy.com/blog/dronedeploy-segmentation-benchmark-challenge/

Isola, P., Zhu, J.-Y., Zhou, T., & Efros, A. A. (2017). Image-to-Image Translation with Conditional Adversarial Networks. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 5967–5976. https://doi.org/10.1109/CVPR.2017.632

Kameyama, S., & Sugiura, K. (2021). Effects of Differences in Structure from Motion Software on Image Processing of Unmanned Aerial Vehicle Photography and Estimation of Crown Area and Tree Height in Forests. *Remote Sensing*, *13*(4), 626. https://doi.org/10.3390/rs13040626

Kearney, S. P., Coops, N. C., Sethi, S., & Stenhouse, G. B. (2020). Maintaining accurate, current, rural road network data: An extraction and updating routine using RapidEye, participatory GIS and deep learning. *International Journal of Applied Earth Observation and Geoinformation*, *87*, 102031. https://doi.org/10.1016/j.jag.2019.102031

Li, Z., Xin, Q., Sun, Y., & Cao, M. (2021). A Deep Learning-Based Framework for Automated Extraction of Building Footprint Polygons from Very High-Resolution Aerial Imagery. *Remote Sensing*, *13*(18), 3630. https://doi.org/10.3390/rs13183630

Long, J., Shelhamer, E., & Darrell, T. (2015). *Fully Convolutional Networks for Semantic Segmentation*. 3431–3440.

https://openaccess.thecvf.com/content_cvpr_2015/html/Long_Fully_Convolutional_Networks_2015_CVPR_paper.html

Long, Y., Xia, G.-S., Li, S., Yang, W., Yang, M. Y., Zhu, X. X., Zhang, L., & Li, D. (2021). On Creating Benchmark Dataset for Aerial Image Interpretation: Reviews, Guidances, and Million-AID. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, *14*, 4205–4230. https://doi.org/10.1109/JSTARS.2021.3070368

López-Tapia, S., Ruiz, P., Smith, M., Matthews, J., Zercher, B., Sydorenko, L., Varia, N., Jin, Y., Wang, M., Dunn, J. B., & Katsaggelos, A. K. (2021). Machine learning with high-resolution aerial imagery and data fusion to improve and automate the detection of wetlands. *International Journal of Applied Earth Observation and Geoinformation*, *105*, 102581. https://doi.org/10.1016/j.jag.2021.102581

Machine Learning Mastery. (n.d.). Machine Learning Mastery. Retrieved August 1, 2022, from https://machinelearningmastery.com/

Maggiori, E., Tarabalka, Y., Charpiat, G., & Alliez, P. (2017). Can semantic labeling methods generalize to any city? The inria aerial image labeling benchmark. *2017 IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*, 3226–3229. https://doi.org/10.1109/IGARSS.2017.8127684

Mahammad, S. S., & Ramakrishnan, R. (2003). Map India 2003 Image Processing & Interpretation. *Image Processing*, 10.

Marmanis, D., Wegner, J. D., Galliani, S., Schindler, K., Datcu, M., & Stilla, U. (n.d.). *SEMANTIC SEGMENTATION OF AERIAL IMAGES WITH AN ENSEMBLE OF CNNS*. 8.

Mnih, V., & Hinton, G. E. (2010). Learning to Detect Roads in High-Resolution Aerial Images. In K. Daniilidis, P. Maragos, & N. Paragios (Eds.), *Computer Vision – ECCV 2010* (pp. 210–223). Springer. https://doi.org/10.1007/978-3-642-15567-3_16

Murtiyoso, A., Veriandi, M., Suwardhi, D., Soeksmantono, B., & Harto, A. B. (2020). Automatic Workflow for Roof Extraction and Generation of 3D CityGML Models from Low-Cost UAV Image-Derived Point Clouds. *ISPRS International Journal of Geo-Information*, *9*(12), 743. https://doi.org/10.3390/ijgi9120743

Ng, V., & Hofmann, D. (2018). *Scalable Feature Extraction with Aerial and Satellite Imagery*. 145–151. https://doi.org/10.25080/Majora-4af1f417-015

OpenStreetMap. (n.d.). OpenStreetMap. Retrieved August 1, 2022, from https://www.openstreetmap.org/

Osco, L. P., Marcato Junior, J., Marques Ramos, A. P., de Castro Jorge, L. A., Fatholahi, S. N., de Andrade Silva, J., Matsubara, E. T., Pistori, H., Gonçalves, W. N., & Li, J. (2021). A review on deep learning in UAV remote sensing. *International Journal of Applied Earth Observation and Geoinformation*, *102*, 102456. https://doi.org/10.1016/j.jag.2021.102456

Pan, X., Yang, F., Gao, L., Chen, Z., Zhang, B., Fan, H., & Ren, J. (2019). Building Extraction from High-Resolution Aerial Imagery Using a Generative Adversarial Network with Spatial and Channel Attention Mechanisms. *Remote Sensing*, *11*(8), 917. https://doi.org/10.3390/rs11080917

Pashaei, M., Kamangir, H., Starek, M. J., & Tissot, P. (2020). Review and Evaluation of Deep Learning Architectures for Efficient Land Cover Mapping with UAS Hyper-Spatial Imagery: A Case Study Over a Wetland. *Remote Sensing*, *12*(6), 959. https://doi.org/10.3390/rs12060959

Pillow. (n.d.). Retrieved August 1, 2022, from https://pillow.readthedocs.io/en/stable/index.html

Pinto, L., Bianchini, F., Nova, V., & Passoni, D. (2020). LOW-COST UAS PHOTOGRAMMETRY FOR ROAD INFRASTRUCTURES' INSPECTION. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, *XLIII-B2-2020*, 1145–1150. https://doi.org/10.5194/isprs-archives-XLIII-B2-2020-1145-2020

Pote, R. (2021, August). *Polygonal delineation of greenhouses using a deep learning strategy* [Info:eu-repo/semantics/masterThesis]. University of Twente. http://essay.utwente.nl/89006/

Professional photogrammetry and drone mapping software. (n.d.). Pix4D. Retrieved August 1, 2022, from https://www.pix4d.com/

Radford, A., Metz, L., & Chintala, S. (2016). *Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks* (arXiv:1511.06434). arXiv. https://doi.org/10.48550/arXiv.1511.06434

Ren, J., & Xu, L. (2015). On Vectorization of Deep Convolutional Neural Networks for Vision Tasks. *Proceedings of the AAAI Conference on Artificial Intelligence*, *29*(1), Article 1. https://ojs.aaai.org/index.php/AAAI/article/view/9488

Ronneberger, O., Fischer, P., & Brox, T. (2015). U-Net: Convolutional Networks for Biomedical Image Segmentation. In N. Navab, J. Hornegger, W. M. Wells, & A. F. Frangi (Eds.), *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015* (pp. 234–241). Springer International Publishing. https://doi.org/10.1007/978-3-319-24574-4_28

Saeedimoghaddam, M., & Stepinski, T. F. (2020). Automatic extraction of road intersection points from USGS historical map series using deep convolutional neural networks. *International Journal of Geographical Information Science*, *34*(5), 947–968. https://doi.org/10.1080/13658816.2019.1696968

Sahu, M., & Ohri, A. (2019a). VECTOR MAP GENERATION FROM AERIAL IMAGERY USING DEEP LEARNING. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, *IV-2/W5*, 157–162. https://doi.org/10.5194/isprs-annals-IV-2-W5-157-2019

Sahu, M., & Ohri, A. (2019b). *VECTOR MAP GENERATION FROM AERIAL IMAGERY USING DEEP LEARNING*. http://localhost:8080/xmlui/handle/123456789/520

Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., Chen, X., & Chen, X. (2016). Improved Techniques for Training GANs. *Advances in Neural Information Processing Systems*, *29*. https://proceedings.neurips.cc/paper/2016/hash/8a3363abe792db2d8761d6403605aeb7-Abstract.html

Sester, M., Feng, Y., & Thiemann, F. (2018). Building generalization using deep learning. *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences XLII-4 (2018)*, *XLII–4*, 565–572. https://doi.org/10.15488/5169

Shermeyer, J., & Etten, A. (2019). *The Effects of Super-Resolution on Object Detection Performance in Satellite Imagery*. 1432–1441. https://doi.org/10.1109/CVPRW.2019.00184

Shermeyer, J., & Van Etten, A. (2019). The Effects of Super-Resolution on Object Detection Performance in Satellite Imagery. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 1432–1441. https://doi.org/10.1109/CVPRW.2019.00184

Song, A., & Kim, Y. (2020). Semantic Segmentation of Remote-Sensing Imagery Using Heterogeneous Big Data: International Society for Photogrammetry and Remote Sensing Potsdam and Cityscape Datasets. *ISPRS International Journal of Geo-Information*, *9*(10), 601. https://doi.org/10.3390/ijgi9100601

Sun, W., & Wang, R. (2018). Fully Convolutional Networks for Semantic Segmentation of Very High Resolution Remotely Sensed Images Combined With DSM. *IEEE Geoscience and Remote Sensing Letters*, *15*(3), 474–478. https://doi.org/10.1109/LGRS.2018.2795531

The Home of Location Technology Innovation and Collaboration | OGC. (n.d.). Retrieved August 1, 2022, from https://www.ogc.org/

Touya, G., Zhang, X., & Lokhat, I. (2019). Is deep learning the new agent for map generalization? *International Journal of Cartography*, *5*(2–3), 142–157. https://doi.org/10.1080/23729333.2019.1613071

Van Etten, A. (2019). Satellite Imagery Multiscale Rapid Detection with Windowed Networks. *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, 735–743. https://doi.org/10.1109/WACV.2019.00083

Van Etten, A., Lindenbaum, D., & Bacastow, T. M. (2019). *SpaceNet: A Remote Sensing Dataset and Challenge Series* (arXiv:1807.01232). arXiv. http://arxiv.org/abs/1807.01232

Vinyals, O., Toshev, A., Bengio, S., & Erhan, D. (2015). *Show and Tell: A Neural Image Caption Generator*. 3156–3164. https://www.cv-foundation.org/openaccess/content_cvpr_2015/html/Vinyals_Show_and_Tell_2015_CVPR_paper.html

Wang, S., Bai, M., Mattyus, G., Chu, H., Luo, W., Yang, B., Liang, J., Cheverie, J., Fidler, S., & Urtasun, R. (2017). TorontoCity: Seeing the World with a Million Eyes. *2017 IEEE International Conference on Computer Vision (ICCV)*, 3028–3036. https://doi.org/10.1109/ICCV.2017.327

Wang, S., Liu, W., Wu, J., Cao, L., Meng, Q., & Kennedy, P. J. (2016). Training deep neural networks on imbalanced data sets. *2016 International Joint Conference on Neural Networks (IJCNN)*, 4368–4374. https://doi.org/10.1109/IJCNN.2016.7727770

Wang, W., Yang, N., Zhang, Y., Wang, F., Cao, T., & Eklund, P. (2016). A review of road extraction from remote sensing images. *Journal of Traffic and Transportation Engineering (English Edition)*, *3*(3), 271–282. https://doi.org/10.1016/j.jtte.2016.05.005

Weir, N., Lindenbaum, D., Bastidas, A., Etten, A., Kumar, V., Mcpherson, S., Shermeyer, J., & Tang, H. (2019). *SpaceNet MVOI: A Multi-View Overhead Imagery Dataset*. 992–1001. https://doi.org/10.1109/ICCV.2019.00108

Wu, H., Xiao, B., Codella, N., Liu, M., Dai, X., Yuan, L., & Zhang, L. (2021). *CvT: Introducing Convolutions to Vision Transformers*. 22–31. https://openaccess.thecvf.com/content/ICCV2021/html/Wu_CvT_Introducing_Convolutions_to_Vision_Transformers_ICCV_2021_paper.html

Xie, Y., Zhu, J., Cao, Y., Feng, D., Hu, M., Li, W., Zhang, Y., & Fu, L. (2020). Refined Extraction Of Building Outlines From High-Resolution Remote Sensing Imagery Based on a Multifeature Convolutional Neural Network and Morphological Filtering. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, *13*, 1842–1855. https://doi.org/10.1109/JSTARS.2020.2991391

Xu, Y., Wu, L., Xie, Z., & Chen, Z. (2018). Building Extraction in Very High Resolution Remote Sensing Imagery Using Deep Learning and Guided Filters. *Remote Sensing*, *10*(1), 144. https://doi.org/10.3390/rs10010144

Yan, H. (2019). *Description Approaches and Automated Generalization Algorithms for Groups of Map Objects*. Springer Singapore. https://doi.org/10.1007/978-981-13-3678-2

Yang, H. L., Yuan, J., Lunga, D., Laverdiere, M., Rose, A., & Bhaduri, B. (2018). Building Extraction at Scale Using Convolutional Neural Network: Mapping of the United States. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*. https://doi.org/10.1109/JSTARS.2018.2835377

Yang, W., Gao, X., Zhang, C., Tong, F., Chen, G., & Xiao, Z. (2021). Bridge Extraction Algorithm Based on Deep Learning and High-Resolution Satellite Image. *Scientific Programming*, *2021*, e9961963. https://doi.org/10.1155/2021/9961963

Zhang, Q., Qin, R., Huang, X., Fang, Y., & Liu, L. (2015). Classification of Ultra-High Resolution Orthophotos Combined with DSM Using a Dual Morphological Top Hat Profile. *Remote Sensing*, *7*(12), 16422–16440. https://doi.org/10.3390/rs71215840

Zhao, H., Shi, J., Qi, X., Wang, X., & Jia, J. (2017). *Pyramid Scene Parsing Network*. 2881–2890. https://openaccess.thecvf.com/content_cvpr_2017/html/Zhao_Pyramid_Scene_Parsing_CVPR_2017_paper.html

Zhao, S., Liu, Z., Lin, J., Zhu, J.-Y., & Han, S. (2020). Differentiable Augmentation for Data-Efficient GAN Training. *Advances in Neural Information Processing Systems*, *33*, 7559–7570. https://proceedings.neurips.cc/paper/2020/hash/55479c55ebd1efd3ff125f1337100388-Abstract.html

Zhu, J.-Y., Park, T., Isola, P., & Efros, A. A. (2017). Unpaired Image-to-Image Translation Using Cycle-Consistent Adversarial Networks. *2017 IEEE International Conference on Computer Vision (ICCV)*, 2242–2251. https://doi.org/10.1109/ICCV.2017.244

Annexes

Annex 1. Python Scripts

This Annex documents the scripts used within the proposed pipeline. Scripts are run in Google Collab or locally using Jupyter Notebooks. They are grouped in the three methods of the proposed methodology.

Paired Data

Data entry has two inputs, one is the raster format composed by orthomosaics and DSM. The other is the vector point, line, or polygon layers that form the ground truth.

Raster Layers

For the raster layers stream three block of scripts are developed, these are:

- Geometric augmentation: mirroring (horizontal and vertical), rotations (10 degrees increments), overlapping (10%, 20%).

```
# This script performs geometric augmentation over orthomosaics and GT masks, flip top to bottom, left to right, 10 deg

import os
import sys
from PIL import Image
Image.MAX_IMAGE_PIXELS = None

mypath = "C:/data/"

imagenes = 1
for f in os.listdir(mypath):
   i = Image.open(os.path.join(mypath, f))
   print(f, i.size, i.mode)
   fname, fext = os.path.splitext(f)
   i.rotate(10).convert('RGB').save(os.path.join(mypath,'n{}.tif'.format(fname)))
   i.transpose(Image.FLIP_TOP_BOTTOM).convert('RGB').save(os.path.join(mypath,'v{}.tif'.format(fname)))
   i.transpose(Image.FLIP_LEFT_RIGHT).convert('RGB').save(os.path.join(mypath,'h{}.tif'.format(fname)))
   imagenes = imagenes + 1
print("processed orthomosaics: "+ str((imagenes-1)*3))
```

```
# This script makes overlapping images from an orthomosaic (10%, 20%) with a specific chip size.
import os
import cv2

path_to_img = r"C:\data\NameOrthomosaic.tif"
path_to_msk = r"C:\data\NameMask.tif"
savedir = "C:/data/"

img = cv2.imread(path_to_img)
img_h, img_w, _ = img.shape
split_width = 256
split_height = 256

def start_points(size, split_size, overlap=0.2):
    points = [0]
    stride = int(split_size * (1-overlap))
    counter = 1
    while True:
        pt = stride * counter
        if pt + split_size >= size:
            points.append(size - split_size)
            break
        else:
            points.append(pt)
        counter += 1
    return points

X_points = start_points(img_w, split_width, overlap = 0.2)
Y_points = start_points(img_h, split_height, overlap = 0.2)

count = 0
name = 's'
frmt = 'png'

for i in Y_points:
    for j in X_points:
        split = img[i:i+split_height, j:j+split_width]
        cv2.imwrite(os.path.join(savedir + '{}_{}.{}'.format(name, count, frmt)), split)
        count += 1
print("Se han producido ", count, "imagenes de ",  overlap, "de overlap")
```

- Spectral augmentation: brightness, contrast and gamma value (intensity).

```
# This script makes spectral augmentation over orthomosaics changing brightness and contrast to 20% more

import os
import sys
from PIL import Image, ImageEnhance
Image.MAX_IMAGE_PIXELS = None

mypath = "C:/data/"
```

```
myorthomosaic = "Name.tif"

def change_brightness(path, ortomosaic, brightness = 1.2):
   img = Image.open(os.path.join(path, ortomosaic))
   print(img.size, img.mode)
   enhancer = ImageEnhance.Brightness(img)
   print("Se produjo una ortofoto con iluminacion de: "+ str(brightness))
   return enhancer.enhance(brightness).save(os.path.join(path,'Ilum_{}'.format(ortomosaic)))

def change_contrast(path, ortomosaic, contrast = 1.2):
   img = Image.open(os.path.join(path, ortomosaic))
   print(img.size, img.mode)
   enhancer = ImageEnhance.Contrast(img)
   print("Se produjo una ortofoto con contraste de: "+ str(contrast))
   return enhancer.enhance(contrast).save(os.path.join(path,'Cont_{}'.format(ortomosaic)))

change_brightness(mypath,myorthomosaic)
change_contrast(mypath,myorthomosaic)
```

- Data fusion: production of fusion of images into three channels, and indexes like VARI.

Vector Layers information from OSM

For the vector layers stream, a workflow is developed in python as well as in Model Builder-ARCGIS. OSM is a good, probably the default, source of point, line, and polygon information.

```
# This script downloads point, line or polygon objects of interest from OSM within coordinates (Xmin, Ymin), (Xmax, Ymax)

!pip install overpass
!pip install geopandas

from google.colab import drive
drive.mount('/content/drive', force_remount=True)
root_dir = "/content/drive/My Drive/"
base_dir = root_dir + 'automap/'
print("base_dir is: ", base_dir)

import os
import overpass
import geopandas as gpd
import time
import json

mypath = base_dir + 'ortofotos/vias/'
myname = "roadsGirardota.geojson"
mynameout = "roadsGirardota.shp"
mytime = time.time()

api = overpass.API()
data = api.get('way["highway"](6.367,-75.458,6.391,-75.433);', verbosity='geom', responseformat="geojson")
```

```
[f for f in data.features  if f.geometry['type'] == "LineString"]
with open(os.path.join(mypath, myname), 'w', encoding='utf-8') as myfile:
    json.dump(data, myfile, ensure_ascii=False, indent=4)
print("Se creo un mapa: ", myname, 'en', mypath)
gdf = gpd.read_file(os.path.join(mypath, myname))
gdf.to_file(os.path.join(mypath, mynameout), driver='Shapefile')
mytimef = time.time()
print("Se ha creado con éxito:", mynameout,'a partir de un geojson en:', round((mytimef-mytime),2), 'segundos')
```

Area and extension of Raster Drone Imagery

```
raster_path = r'C:\Users\john\Downloads\AutoMap\datos\El_Retiro600.tif'
ras = arcpy.Raster(raster_path)
area = ((ras.width*ras.height)*(ras.meanCellWidth*ras.meanCellHeight)*(111**2)*100)
print(round(area,1))
print(ras.extent)
print(ras.meanCellWidth)
```

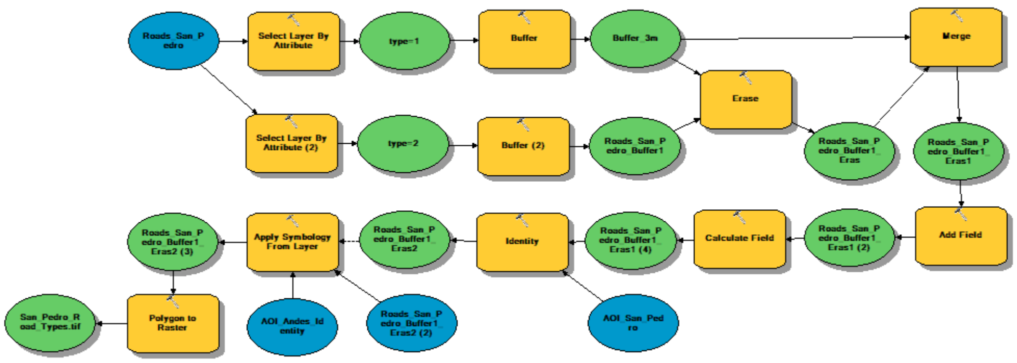
Production of binary or attributed mask from point, line or polygon

```
# This script creates a point, line or polygon dataset from ground truth shapefiles
import arcpy

# Local variables:
Roads_San_Pedro_Buffer1_Eras2__2_ = "Roads_San_Pedro_Buffer1_Eras2"
Roads_San_Pedro_Buffer1_Eras2__3_ = Roads_San_Pedro_Buffer1_Eras2__2_
AOI_Andes_Identity = "AOI_Andes_Identity"
AOI_San_Pedro = "AOI_San_Pedro"
Roads_San_Pedro = "Roads_San_Pedro"
type_1 = Roads_San_Pedro
type_2 = Roads_San_Pedro
Roads_San_Pedro_Buffer1 = "C:\\Users\\PC-ONE\\Documents\\ArcGIS\\Default.gdb\\Roads_San_Pedro_Buffer1"
Buffer_3m = "C:\\Users\\PC-ONE\\Documents\\ArcGIS\\Default.gdb\\Roads_San_Pedro_Buffer"
Roads_San_Pedro_Buffer1_Eras = "C:\\Users\\PC-
ONE\\Documents\\ArcGIS\\Default.gdb\\Roads_San_Pedro_Buffer1_Eras"
Roads_San_Pedro_Buffer1_Eras1 = "C:\\Users\\PC-
ONE\\Documents\\ArcGIS\\Default.gdb\\Roads_San_Pedro_Buffer1_Eras1"
Roads_San_Pedro_Buffer1_Eras1__2_ = Roads_San_Pedro_Buffer1_Eras1
Roads_San_Pedro_Buffer1_Eras1__4_ = Roads_San_Pedro_Buffer1_Eras1__2_
Roads_San_Pedro_Buffer1_Eras2 = "C:\\Users\\PC-
ONE\\Documents\\ArcGIS\\Default.gdb\\Roads_San_Pedro_Buffer1_Eras2"
Feature_Road1 = "C:\\Users\\PC-ONE\\Documents\\ArcGIS\\Default.gdb\\Feature_Road1"
ortofotos = "C:\\Users\\PC-ONE\\Downloads\\ortofotos"
ortofotos__2_ = ortofotos

# Process: Select Layer By Attribute (2)
arcpy.SelectLayerByAttribute_management(Roads_San_Pedro, "NEW_SELECTION", "\"type\" = 2")

# Process: Buffer (2)
arcpy.Buffer_analysis(type_2, Roads_San_Pedro_Buffer1, "150 Centimeters", "FULL", "ROUND", "ALL", "", "PLANAR")
```

```
# Process: Select Layer By Attribute
arcpy.SelectLayerByAttribute_management(Roads_San_Pedro, "NEW_SELECTION", "\"type\" = 1")

# Process: Buffer
arcpy.Buffer_analysis(type_1, Buffer_3m, "3 Meters", "FULL", "ROUND", "ALL", "", "PLANAR")

# Process: Erase
arcpy.Erase_analysis(Roads_San_Pedro_Buffer1, Buffer_3m, Roads_San_Pedro_Buffer1_Eras, "")

# Process: Merge
arcpy.Merge_management("C:\\Users\\PC-
ONE\\Documents\\ArcGIS\\Default.gdb\\Roads_San_Pedro_Buffer1_Eras;C:\\Users\\PC-
ONE\\Documents\\ArcGIS\\Default.gdb\\Roads_San_Pedro_Buffer", Roads_San_Pedro_Buffer1_Eras1, "surface
\"surface\" true false false 80 Text 0 0 ,First,#,C:\\Users\\PC-
ONE\\Documents\\ArcGIS\\Default.gdb\\Roads_San_Pedro_Buffer,surface,-1,-1;max_speed \"max_speed\" true false
false 10 Long 0 10 ,First,#,C:\\Users\\PC-
ONE\\Documents\\ArcGIS\\Default.gdb\\Roads_San_Pedro_Buffer,max_speed,-1,-1;type \"type\" true false false 10
Long 0 10 ,First,#,C:\\Users\\PC-ONE\\Documents\\ArcGIS\\Default.gdb\\Roads_San_Pedro_Buffer,type,-1,-1;quality
\"quality\" true false false 30 Text 0 0 ,First,#,C:\\Users\\PC-
ONE\\Documents\\ArcGIS\\Default.gdb\\Roads_San_Pedro_Buffer,quality,-1,-1;class \"class\" true false false 50 Text 0
0 ,First,#,C:\\Users\\PC-ONE\\Documents\\ArcGIS\\Default.gdb\\Roads_San_Pedro_Buffer,class,-1,-1;BUFF_DIST
\"BUFF_DIST\" true true false 0 Double 0 0 ,First,#,C:\\Users\\PC-
ONE\\Documents\\ArcGIS\\Default.gdb\\Roads_San_Pedro_Buffer,BUFF_DIST,-1,-1;ORIG_FID \"ORIG_FID\" true true
false 0 Long 0 0 ,First,#,C:\\Users\\PC-ONE\\Documents\\ArcGIS\\Default.gdb\\Roads_San_Pedro_Buffer,ORIG_FID,-1,-
1;Shape_Length \"Shape_Length\" true true true 8 Double 0 0 ,First,#,C:\\Users\\PC-
ONE\\Documents\\ArcGIS\\Default.gdb\\Roads_San_Pedro_Buffer1_Eras,Shape_Length,-1,-1,C:\\Users\\PC-
ONE\\Documents\\ArcGIS\\Default.gdb\\Roads_San_Pedro_Buffer1_Eras,Shape_length,-1,-1;Shape_Area
\"Shape_Area\" true true true 8 Double 0 0 ,First,#,C:\\Users\\PC-
ONE\\Documents\\ArcGIS\\Default.gdb\\Roads_San_Pedro_Buffer1_Eras,Shape_Area,-1,-1,C:\\Users\\PC-
ONE\\Documents\\ArcGIS\\Default.gdb\\Roads_San_Pedro_Buffer1_Eras,Shape_area,-1,-1")

# Process: Add Field
arcpy.AddField_management(Roads_San_Pedro_Buffer1_Eras1, "type", "SHORT", "", "", "", "", "NULLABLE",
"NON_REQUIRED", "")

# Process: Calculate Field
arcpy.CalculateField_management(Roads_San_Pedro_Buffer1_Eras1__2_, "type", "reclass(!OBJECTID!)", "PYTHON_9.3",
"def reclass(OBJECTID):\\n    if (OBJECTID == 1):\\n        return 1\\n    else:\\n        return 2\\n")

# Process: Identity
arcpy.Identity_analysis(AOI_San_Pedro, Roads_San_Pedro_Buffer1_Eras1__4_, Roads_San_Pedro_Buffer1_Eras2, "ALL",
"", "NO_RELATIONSHIPS")

# Process: Apply Symbology From Layer
tempEnvironment0 = arcpy.env.cartographicPartitions
arcpy.env.cartographicPartitions = Roads_San_Pedro_Buffer1_Eras2
arcpy.ApplySymbologyFromLayer_management(Roads_San_Pedro_Buffer1_Eras2__2_, AOI_Andes_Identity)
arcpy.env.cartographicPartitions = tempEnvironment0

# Process: Feature to Raster
```

```
arcpy.FeatureToRaster_conversion(Roads_San_Pedro_Buffer1_Eras2__3_, "type", Feature_Road1, "C:\\Users\\PC-ONE\\Downloads\\ortofotos\\wgs84\\San_Pedro.tif")

# Process: Raster To Other Format (multiple)
arcpy.RasterToOtherFormat_conversion("C:\\Users\\PC-ONE\\Documents\\ArcGIS\\Default.gdb\\Feature_Road1", ortofotos, "TIFF")
```

It is also possible to develop the previous script using the Model Builder in ARCGIS, see Figure A1.2:



Roads_San_P edro
Select Layer By Attribute
type=1
Buffer
Buffer_3m
Merge
Erase
Select Layer By Attribute (2)
type=2
Buffer (2)
Roads_San_P edro_Buffer1
Roads_San_P edro_Buffer1_ Eras
Roads_San_P edro_Buffer1_ Eras1
Roads_San_P edro_Buffer1_ Eras2 (3)
Apply Symbology From Layer
Roads_San_P edro_Buffer1_ Eras2
Identity
Roads_San_P edro_Buffer1_ Eras1 (4)
Calculate Field
Roads_San_P edro_Buffer1_ Eras1 (2)
Add Field
San_Pedro_R oad_Types.tif
Polygon to Raster
AOI_Andes_Id entity
Roads_San_P edro_Buffer1_ Eras2 (2)
AOI_San_Ped ro

Figure A1.2. Script to create GT masks from point, line or polygon vector layers.

Image Captioning

A script in ArcGIS to create an image captioning dataset is presented next.

```
# This script creates a captioning dataset: (img, attribute list)
import arcpy

# Local variables:
AOI_Andes = "Roads\\AOI_Andes"
Output_Labels = ""
Roads_Andes = "Roads_Andes"
Network_Image_Caption = "C:\\Users\\PC-ONE\\Downloads\\ortofotos\\RSQS_Dataset.mdb\\Roads\\Network_Image_Caption"
Roads_Andes_Captioned = "C:\\Users\\PC-ONE\\Downloads\\ortofotos\\RSQS_Dataset.mdb\\Roads\\Roads_Andes_Captioned"

# Process: Create Fishnet
Height_size = 0,00021954578432
Base_size = 0,00021954578432
arcpy.CreateFishnet_management(Network_Image_Caption, "-75,889360382 5,64578355200007", "-75,889360382 15,6457835520001", " Height_size ", " Base_size ", "", "", "-75,871570313 5,67187862600002", "NO_LABELS", AOI_Andes, "POLYGON")

# Process: Intersect
```

```
arcpy.Intersect_analysis("Roads_Andes #;C:\\Users\\PC-
ONE\\Downloads\\ortofotos\\RSQS_Dataset.mdb\\Roads\\Network_Image_Caption #", Roads_Andes_Captioned,
"NO_FID", "", "INPUT")
```

Tessellation, Pairing, and Dataset Splitting

For the central stream, three scripts were developed:

- Tessellation: partition of image and mask to certain size.

```
# This script tessellates orthomosaics and corresponding masks into chips img,msk of a specific size

import os
import sys
from PIL import Image
Image.MAX_IMAGE_PIXELS = None

savedir = "C:/data/img/"
path = "C:/data/img_in"

start_pos = start_x, start_y = (0, 0)
cropped_image_size = w, h = (512,512)
frame_num = 1

for filename in sorted(os.listdir(path)):
   img = Image.open(path + '/' + filename)
   ##if img.endswith('.tiff'):
   print(path + '/' + filename)
   width, height = img.size
   for col_i in range(0, width, w):
         for row_i in range(0, height, h):
            crop = img.crop((col_i, row_i, col_i + w, row_i + h))
            save_to= os.path.join(savedir + "{:1}.png")
            crop.convert('RGB').save(save_to.format(frame_num))
            frame_num += 1
print('El numero de imagenes de ' + str(w) + 'x' + str(h) + ' obtenidas fue de:', frame_num-1)

savedir2 = "C:/data/msk/"
path2 = "C:/data/msk_in"

start_pos = start_x, start_y = (0, 0)
#cropped_image_size = w, h = (600,600)
#cropped_image_size = w, h = (256,256)
frame_num = 1

for filename in sorted(os.listdir(path2)):
   img = Image.open(path2 + '/' + filename)
   print(path2 + '/' + filename)
   ##if img.endswith('.tiff'):
   width, height = img.size
   for col_i in range(0, width, w):
         for row_i in range(0, height, h):
            crop = img.crop((col_i, row_i, col_i + w, row_i + h))
            save_to= os.path.join(savedir2 + "{:1}.png")
            crop.save(save_to.format(frame_num))
            frame_num += 1
print('El numero de mascaras de ' + str(w) + 'x' + str(h) + ' obtenidas fue de:', frame_num-1)
```

- Pairing: put together each image chip and corresponding mask into one sole image.

```
# This script put img and corresponding msk together in one image

import os
import sys
from PIL import Image, ImageEnhance
Image.MAX_IMAGE_PIXELS = None

myfolder1 = "C:/data/img/"
myfolder2 = "C:/data/msk/"
myoutfolder = "C:/data/dataset512/"

total_width = 1024
max_height = 512

lst1 = []
lst2 = []
#formato = '.jpg'
formato = '.png'

new_im = Image.new('RGB', (total_width, max_height), color=(0,0,0))

for count, filename in enumerate(sorted(os.listdir(myfolder1))):
   (lst1.append(filename))

for count2, filename2 in enumerate(sorted(os.listdir(myfolder2))):
   (lst2.append(filename2))

for i in range(0,len(lst1)):
   im=Image.open(myfolder1+str(lst1[i])).convert("RGB")
   #im=Image.open(myfolder1+str(lst1[i]))
   img=Image.open(myfolder2+str(lst2[i])).convert("RGB")
   #img=Image.open(myfolder2+str(lst2[i]))
   new_im.paste(im, (0,0))
   new_im.paste(img, (max_height,0))
   new_im.save(myoutfolder + str(i+1) + formato)
print("Se han emparejado " + str(i+1) + ' imagenes de ' + str(max_height) +' pixeles en el formato: ' + formato)
```

- Splitting: randomly split the dataset into training, validation and testing.

```
# This script randomly splits the dataset into 80% for training, and 20% for validation and testing, and 50% of the last partition for test.
```

```
import os
source = "C:/data/dataset512"
dest1 = "C:/data/dataset512/valid"
dest2 = "C:/data/dataset512/test"

p1 = 0.2 # this is the percentage split value

files = os.listdir(source)
import shutil
import numpy as np
for f in files:
   if np.random.rand(1) < p1:
      shutil.move(source + '/'+ f, dest1 + '/'+ f)

p2 = 0.5
files = os.listdir(dest1)
for f in files:
   if np.random.rand(1) < p2:
      shutil.move(dest1 + '/'+ f, dest2 + '/'+ f)
print("Split done",str(p1*100),"% validation",str(p2*100), "% test")
```

- Imbalance calculation: calculate the degree of imbalance of a dataset.

```
# This code calculates the percentage of white pixels over total pixels in image

import cv2
import numpy as np
import os
#from PIL import Image

path = os.path.join(base_dir + 'dataset256_3m_RGB/', '750/')

count = 0
tot_white_px = 0
tot_blk_px = 0

for filename in os.listdir(path):
   img = cv2.imread(path + '/' + filename, cv2.IMREAD_GRAYSCALE)
   n_white_pix = np.sum(img == 255)
   tot_white_px = n_white_pix + tot_white_px
   n_black_pix = np.sum(img == 0)
   tot_blk_px = tot_blk_px + n_black_pix
   #if (count == 0):
                        #    mylog = open(base_dir + 'dataset256_3m_RGB/' + "imbalance_log.log", "a")
                        #    mylog.write('dataset_size' + ',' + 'wh_px' + ',' + 'bl_px'+','+'tot_px'+','+'imbalance' + '\n')
                        #    mylog.close()
   if (count == len(os.listdir(path))-1):
                           mylog = open(base_dir + 'dataset256_3m_RGB/' + "imbalance_log.log", "a")

mylog.write(str(len(os.listdir(path)))+','+str(tot_white_px)+','+str(tot_blk_px)+','+str(tot_blk_px+tot_white_px)+','+str(ro
und((tot_white_px/(tot_blk_px+tot_white_px))*100,1))+'\n')
```

```
                    mylog.close()
    count += 1
print('Number of white pixels:', tot_white_px)
print('Number of black pixels:', tot_blk_px)
print('Total pixels in the dataset:', tot_blk_px+tot_white_px)
print('imbalance:', round((tot_white_px/(tot_blk_px+tot_white_px))*100,1), "%")
print('Number of processed images:', count)
print('> Log of dataset imbalance has been saved into:', base_dir + 'dataset256_3m_RGB/')
```

- Removing black masks: removes black no information masks in the dataset.

```
# This code deletes (removes) all masks and the corresponding image chips that are only black (only one color)

from PIL import Image
import os
path = "C:/Users/john/Downloads/AutoMap/datos/msk256_RGB_Roads_1m/"
path2 = "C:/Users/john/Downloads/AutoMap/datos/img256_RGB_Roads_1m/"

count = 0
for filename in os.listdir(path):
    img = Image.open(path + '\\' + filename)
    clrs = img.getcolors()
    if len(clrs) == 1:
        count = count + 1
        print(filename, len(clrs), count)
        os.remove(path + '\\' + filename)
        os.remove(path2 + '\\' + filename)
print("El numero total de imagenes borradas es de:", str(count))
print("El numero total de máscaras borradas es de:", str(count))
```

Models

CycleGAN:

```
from google.colab import drive
drive.mount('/content/drive', force_remount=True)
root_dir = "/content/drive/My Drive/automap/"
base_dir = root_dir + 'CycleGan/'
print(base_dir)

# example of preparing the imgs to roads
from os import listdir
from numpy import asarray
from numpy import vstack
from keras.preprocessing.image import img_to_array
from keras.preprocessing.image import load_img
from numpy import savez_compressed
import os
from random import random
from numpy import load
from numpy import zeros
from numpy import ones
from numpy import asarray
from numpy.random import randint
from tensorflow.keras.optimizers import Adam
from keras.initializers import RandomNormal
from keras.models import Model
from keras.models import Input
from keras.layers import Conv2D
from keras.layers import Conv2DTranspose
from keras.layers import LeakyReLU
from keras.layers import Activation
from keras.layers import Concatenate
from keras_contrib.layers.normalization.instancenormalization import InstanceNormalization

# load all images in a directory into memory
def load_images(path, size=(256,256)):
        data_list = list()
        # enumerate filenames in directory, assume all are images
        for file_name in listdir(path):
                # load and resize the image
                pixels = load_img(path + file_name, target_size=size)
                # convert to numpy array
                pixels = img_to_array(pixels)
                # store
                data_list.append(pixels)
        return asarray(data_list)

# dataset path
#path = base_dir + 'img2road/'
path = base_dir + 'img2build/'
# load dataset A
dataA1 = load_images(path + 'trainA/')
dataAB = load_images(path + 'testA/')
#print(len(dataA1))
#print(len(dataAB))
dataA = vstack((dataA1, dataAB))
print('Loaded dataA: ', dataA.shape)
# load dataset B
dataB1 = load_images(path + 'trainB/')
dataB2 = load_images(path + 'testB/')
dataB = vstack((dataB1, dataB2))
print('Loaded dataB: ', dataB.shape)
# save as compressed numpy array
#filename = os.path.join(base_dir, 'img2road_256.npz')
filename = os.path.join(base_dir, 'img2build_256.npz')
savez_compressed(filename, dataA, dataB)
print('Saved dataset: ', filename)

# load and plot the prepared dataset
from numpy import load
from matplotlib import pyplot
# load the face dataset
```

```
data = load(os.path.join(base_dir, 'img2build_256.npz'))
dataA, dataB = data['arr_0'], data['arr_1']
print('Loaded: ', dataA.shape, dataB.shape)
# plot source images
n_samples = 3
for i in range(n_samples):
	pyplot.subplot(2, n_samples, 1 + i)
	pyplot.axis('off')
	pyplot.imshow(dataA[i].astype('uint8'))
# plot target image
for i in range(n_samples):
	pyplot.subplot(2, n_samples, 1 + n_samples + i)
	pyplot.axis('off')
	pyplot.imshow(dataB[i].astype('uint8'))
pyplot.show()

# define the discriminator model
def define_discriminator(image_shape):
	# weight initialization
	init = RandomNormal(stddev=0.02)
	# source image input
	in_image = Input(shape=image_shape)
	# C64
	d = Conv2D(64, (4,4), strides=(2,2), padding='same', kernel_initializer=init)(in_image)
	d = LeakyReLU(alpha=0.2)(d)
	# C128
	d = Conv2D(128, (4,4), strides=(2,2), padding='same', kernel_initializer=init)(d)
	d = InstanceNormalization(axis=-1)(d)
	d = LeakyReLU(alpha=0.2)(d)
	# C256
	d = Conv2D(256, (4,4), strides=(2,2), padding='same', kernel_initializer=init)(d)
	d = InstanceNormalization(axis=-1)(d)
	d = LeakyReLU(alpha=0.2)(d)
	# C512
	d = Conv2D(512, (4,4), strides=(2,2), padding='same', kernel_initializer=init)(d)
	d = InstanceNormalization(axis=-1)(d)
	d = LeakyReLU(alpha=0.2)(d)
	# second last output layer
	d = Conv2D(512, (4,4), padding='same', kernel_initializer=init)(d)
	d = InstanceNormalization(axis=-1)(d)
	d = LeakyReLU(alpha=0.2)(d)
	# patch output
	patch_out = Conv2D(1, (4,4), padding='same', kernel_initializer=init)(d)
	# define model
	model = Model(in_image, patch_out)
	# compile model
	model.compile(loss='mse', optimizer=Adam(lr=0.0002, beta_1=0.5), loss_weights=[0.5])
	return model

# generator a resnet block
def resnet_block(n_filters, input_layer):
	# weight initialization
	init = RandomNormal(stddev=0.02)
	# first layer convolutional layer
	g = Conv2D(n_filters, (3,3), padding='same', kernel_initializer=init)(input_layer)
	g = InstanceNormalization(axis=-1)(g)
	g = Activation('relu')(g)
	# second convolutional layer
	g = Conv2D(n_filters, (3,3), padding='same', kernel_initializer=init)(g)
	g = InstanceNormalization(axis=-1)(g)
	# concatenate merge channel-wise with input layer
	g = Concatenate()([g, input_layer])
	return g

# define the standalone generator model
def define_generator(image_shape, n_resnet=9):
	# weight initialization
	init = RandomNormal(stddev=0.02)
	# image input
	in_image = Input(shape=image_shape)
	# c7s1-64
	g = Conv2D(64, (7,7), padding='same', kernel_initializer=init)(in_image)
	g = InstanceNormalization(axis=-1)(g)
	g = Activation('relu')(g)
	# d128
	g = Conv2D(128, (3,3), strides=(2,2), padding='same', kernel_initializer=init)(g)
	g = InstanceNormalization(axis=-1)(g)
	g = Activation('relu')(g)
	# d256
	g = Conv2D(256, (3,3), strides=(2,2), padding='same', kernel_initializer=init)(g)
	g = InstanceNormalization(axis=-1)(g)
	g = Activation('relu')(g)
	# R256
	for _ in range(n_resnet):
		g = resnet_block(256, g)
	# u128
	g = Conv2DTranspose(128, (3,3), strides=(2,2), padding='same', kernel_initializer=init)(g)
	g = InstanceNormalization(axis=-1)(g)
	g = Activation('relu')(g)
	# u64
	g = Conv2DTranspose(64, (3,3), strides=(2,2), padding='same', kernel_initializer=init)(g)
	g = InstanceNormalization(axis=-1)(g)
	g = Activation('relu')(g)
	# c7s1-3
	g = Conv2D(3, (7,7), padding='same', kernel_initializer=init)(g)
	g = InstanceNormalization(axis=-1)(g)
	out_image = Activation('tanh')(g)
	# define model
	model = Model(in_image, out_image)
	return model

def define_composite_model(g_model_1, d_model, g_model_2, image_shape):
	# ensure the model we're updating is trainable
	g_model_1.trainable = True
	# mark discriminator as not trainable
	d_model.trainable = False
	# mark other generator model as not trainable
	g_model_2.trainable = False
	# discriminator element
	input_gen = Input(shape=image_shape)
	gen1_out = g_model_1(input_gen)
	output_d = d_model(gen1_out)
	# identity element
	input_id = Input(shape=image_shape)
	output_id = g_model_1(input_id)
	# forward cycle
	output_f = g_model_2(gen1_out)
	# backward cycle
	gen2_out = g_model_2(input_id)
	output_b = g_model_1(gen2_out)
	# define model graph
	model = Model([input_gen, input_id], [output_d, output_id, output_f, output_b])
	# define optimization algorithm configuration
	opt = Adam(lr=0.0002, beta_1=0.5)
	# compile model with weighting of least squares loss and L1 loss
	model.compile(loss=['mse', 'mae', 'mae', 'mae'], loss_weights=[1, 5, 10, 10], optimizer=opt)
	return model

# define a composite model for updating generators by adversarial and cycle loss
# load and prepare training images
def load_real_samples(filename):
	# load the dataset
	data = load(os.path.join(base_dir, filename))
	# unpack arrays
	X1, X2 = data['arr_0'], data['arr_1']
	# scale from [0,255] to [-1,1]
	X1 = (X1 - 127.5) / 127.5
	X2 = (X2 - 127.5) / 127.5
	return [X1, X2]

# select a batch of random samples, returns images and target
def generate_real_samples(dataset, n_samples, patch_shape):
	# choose random instances
	ix = randint(0, dataset.shape[0], n_samples)
	# retrieve selected images
	X = dataset[ix]
	# generate 'real' class labels (1)
	y = ones((n_samples, patch_shape, patch_shape, 1))
	return X, y

# generate a batch of images, returns images and targets
def generate_fake_samples(g_model, dataset, patch_shape):
	# generate fake instance
	X = g_model.predict(dataset)
	# create 'fake' class labels (0)
	y = zeros((len(X), patch_shape, patch_shape, 1))
	return X, y

# save the generator models to file
def save_models(step, g_model_AtoB, g_model_BtoA):
	# save the first generator model
	filename1 = os.path.join(base_dir, 'g_model_AtoB_%06d.h5' % (step+1))
	g_model_AtoB.save(filename1)
	# save the second generator model
	filename2 = os.path.join(base_dir, 'g_model_BtoA_%06d.h5' % (step+1))
	g_model_BtoA.save(filename2)
	print('>Saved: %s and %s' % (filename1, filename2))

# generate samples and save as a plot and save the model
def summarize_performance(step, g_model, trainX, name, n_samples=3):
	# select a sample of input images
	X_in, _ = generate_real_samples(trainX, n_samples, 0)
	# generate translated images
	X_out, _ = generate_fake_samples(g_model, X_in, 0)
	# scale all pixels from [-1,1] to [0,1]
	X_in = (X_in + 1) / 2.0
	X_out = (X_out + 1) / 2.0
	# plot real images
	for i in range(n_samples):
		pyplot.subplot(2, n_samples, 1 + i)
		pyplot.axis('off')
		pyplot.imshow(X_in[i])
	# plot translated image
	for i in range(n_samples):
		pyplot.subplot(2, n_samples, 1 + n_samples + i)
		pyplot.axis('off')
		pyplot.imshow(X_out[i])
	# save plot to file
	filename1 = os.path.join(base_dir, '%s_model_%06d.png' % (name, (step+1)))
	pyplot.savefig(filename1)
	pyplot.close()

# update image pool for fake images
def update_image_pool(pool, images, max_size=50):
	selected = list()
	for image in images:
		if len(pool) < max_size:
			# stock the pool
			pool.append(image)
			selected.append(image)
		elif random() < 0.5:
			# use image, but don't add it to the pool
			selected.append(image)
		else:
```

```
                # replace an existing image and use replaced image
                ix = randint(0, len(pool))
                selected.append(pool[ix])
                pool[ix] = image
    return asarray(selected)

# train cyclegan models
def train(d_model_A, d_model_B, g_model_AtoB, g_model_BtoA, c_model_AtoB, c_model_BtoA, dataset):
    # define properties of the training run
    n_epochs, n_batch, = 100, 1
    # determine the output square shape of the discriminator
    n_patch = d_model_A.output_shape[1]
    # unpack dataset
    trainA, trainB = dataset
    # prepare image pool for fakes
    poolA, poolB = list(), list()
    # calculate the number of batches per training epoch
    bat_per_epo = int(len(trainA) / n_batch)
    # calculate the number of training iterations
    n_steps = bat_per_epo * n_epochs
    # manually enumerate epochs
    for i in range(n_steps):
        # select a batch of real samples
        X_realA, y_realA = generate_real_samples(trainA, n_batch, n_patch)
        X_realB, y_realB = generate_real_samples(trainB, n_batch, n_patch)
        # generate a batch of fake samples
        X_fakeA, y_fakeA = generate_fake_samples(g_model_BtoA, X_realB, n_patch)
        X_fakeB, y_fakeB = generate_fake_samples(g_model_AtoB, X_realA, n_patch)
        # update fakes from pool
        X_fakeA = update_image_pool(poolA, X_fakeA)
        X_fakeB = update_image_pool(poolB, X_fakeB)
        # update generator B->A via adversarial and cycle loss
        g_loss2, _, _, _, _  = c_model_BtoA.train_on_batch([X_realB, X_realA], [y_realA, X_realA, X_realB,
X_realA])
        # update discriminator for A -> [real/fake]
        dA_loss1 = d_model_A.train_on_batch(X_realA, y_realA)
        dA_loss2 = d_model_A.train_on_batch(X_fakeA, y_fakeA)
        # update generator A->B via adversarial and cycle loss
        g_loss1, _, _, _, _ = c_model_AtoB.train_on_batch([X_realA, X_realB], [y_realB, X_realB, X_realA,
X_realB])
        # update discriminator for B -> [real/fake]
        dB_loss1 = d_model_B.train_on_batch(X_realB, y_realB)
        dB_loss2 = d_model_B.train_on_batch(X_fakeB, y_fakeB)
        # summarize performance
        print('>%d, dA[%.3f,%.3f] dB[%.3f,%.3f] g[%.3f,%.3f]' % (i+1, dA_loss1,dA_loss2, dB_loss1,dB_loss2,
g_loss1,g_loss2))
        # evaluate the model performance every so often
        if (i+1) % (bat_per_epo * 1) == 0:
            # plot A->B translation
            summarize_performance(i, g_model_AtoB, trainA, 'AtoB')
            # plot B->A translation
				summarize_performance(i, g_model_BtoA, trainB, 'BtoA')
			if (i+1) % (bat_per_epo * 5) == 0:
				# save the models
				save_models(i, g_model_AtoB, g_model_BtoA)

# load image data
dataset = load_real_samples(os.path.join(base_dir, 'img2build_256.npz'))
print('Loaded', dataset[0].shape, dataset[1].shape)
# define input shape based on the loaded dataset
image_shape = dataset[0].shape[1:]
# generator: A -> B
g_model_AtoB = define_generator(image_shape)
# generator: B -> A
g_model_BtoA = define_generator(image_shape)
# discriminator: A -> [real/fake]
d_model_A = define_discriminator(image_shape)
# discriminator: B -> [real/fake]
d_model_B = define_discriminator(image_shape)
# composite: A -> B -> [real/fake, A]
c_model_AtoB = define_composite_model(g_model_AtoB, d_model_B, g_model_BtoA, image_shape)
# composite: B -> A -> [real/fake, B]
c_model_BtoA = define_composite_model(g_model_BtoA, d_model_A, g_model_AtoB, image_shape)
# train models
train(d_model_A, d_model_B, g_model_AtoB, g_model_BtoA, c_model_AtoB, c_model_BtoA, dataset)

# load an image to the preferred size
def load_image(filename, size=(256,256)):
	# load and resize the image
	pixels = load_img(filename, target_size=size)
	# convert to numpy array
	pixels = img_to_array(pixels)
	# transform in a sample
	pixels = expand_dims(pixels, 0)
	# scale from [0,255] to [-1,1]
	pixels = (pixels - 127.5) / 127.5
	return pixels

# load the image
#image_src = load_image(os.path.join(base_dir, 'n02381460_541.jpg'))
image_src = load_image(os.path.join(base_dir+'img2build/test/', '1.tif'))
# load the model
cust = {'InstanceNormalization': InstanceNormalization}
#model_AtoB = load_model((os.path.join(base_dir, 'g_model_AtoB_011870.h5')), cust)
model_AtoB = load_model((os.path.join(base_dir+'img2build/ResultsAB/', 'g_model_AtoB_001765.h5')), cust)
# Make sure to use the following model to complete the task
#model_AtoB = load_model('g_model_AtoB_100895.h5', cust)
# translate image
image_tar = model_AtoB.predict(image_src)
# scale from [-1,1] to [0,1]
image_tar = (image_tar + 1) / 2.0
# plot the translated image
```

```
pyplot.axis('off')
pyplot.imshow(image_tar[0])
thresh = numpy.mean(image_tar[0])
myarray = numpy.where(image_tar[0] > thresh, 255, 0).astype(numpy.uint8)
pyplot.imsave(os.path.join(base_dir+'img2build/test/', '1_Gen2.png'),myarray, cmap=cm.gray,dpi=500)
pyplot.show()
```

AGS Metric

```
# create a polygon from a center(x,y) plus x=y distance for AGS metric evaluation
import geopandas as gpd
from shapely.geometry import Polygon

#filejson = r"C:\Users\john\Downloads\AutoMap\datos\area\AoI.geojson"
#fileshape = r"C:\Users\john\Downloads\AutoMap\datos\area\AoI_100m.shp"
fileshape = r"C:\Users\john\Downloads\Automap\datos\AoI_200m.shp"
# cx, cy son las coordenadas del centro del poligono
cx = -75.50263
cy = 6.0581
d = 50  # d es la mitad del lado del poligono de corte
x1 = cx - (d/111000.)
x2 = cx + (d/111000.)
y1 = cy - (d/111000.)
y2 = cy + (d/111000.)
#lat_points = [6.0584, 6.0547, 6.0547, 6.0584]
lat_points = [y2, y1, y1, y2]
#lon_points = [-75.5028, -75.5028, -75.4964, -75.4964]
lon_points = [x1, x1, x2, x2]
polygon_geom = Polygon(zip(lon_points, lat_points))
crs = {'init': 'epsg:4326'}
polygon = gpd.GeoDataFrame(index=[0], crs=crs, geometry=[polygon_geom])
print(polygon.geometry)
polygon.to_file(fileshape, driver="ESRI Shapefile")
#polygon.to_file(filejson, driver='GeoJSON')
print('xmin,ymin', x1,y1)
print('xmax,ymax', x2,y2)
import folium
m = folium.Map([cy, cx], zoom_start=15, tiles='cartodbpositron')
#folium.GeoJson(filejson).add_to(m)
folium.LatLngPopup().add_to(m)
m

# Calculates length of every shape file using ArcPy
geometries = arcpy.CopyFeatures_management(fc, arcpy.Geometry())
# Walk through each geometry, totaling the length
length = sum([g.length for g in geometries])
# Execute CalculateField
arcpy.CalculateField_management(in_table, field_name, length, "PYTHON_9.3")
```

Annex 2. Remote Sensing Datasets for Semantic Segmentation of Orthomosaics

High-quality satellite, aerial, and drone imagery datasets facilitate comparisons of existing methods and lead to increased interest in aerial imagery applications in the deep learning and computer vision communities.

Massachusetts buildings dataset

This dataset is derived from the work of (Mnih & Hinton, 2010). It consists of 151 aerial images of the Boston area, with each of the images being 1500 × 1500 pixels for an area of 2.25 square kilometers. Hence, the entire dataset covers roughly 340 square kilometers. The data is split into a training set of 137 images, a test set of 10 images and a validation set of 4 images. The target maps were obtained by rasterizing building footprints obtained from the OpenStreetMap project. The data was restricted to regions with an average omission noise level of roughly 5% or less. The large amount of high-quality building footprint data was possible to collect because the City of Boston contributed building footprints for the entire city to the OpenStreetMap project. The dataset covers mostly urban and suburban areas and buildings of all sizes, including individual houses and garages, are included in the labels. The datasets make use of imagery released by the state of Massachusetts. All imagery is rescaled to a spatial resolution of 1m/px. The target maps for the dataset were generated using data from OSM and represented as binary masks (0:background, 1:building). Target maps for the test and validation portions of the dataset were hand-corrected to make the evaluations more accurate.

Table A2.1. Metrics for Massachusetts Building Dataset

Author	mIoU	F1_Score	Pixel Accuracy	AUC	OA
Mnih, 2013	0.7138	0.8307	0.8311	0.9150	0.9356
Xie et al., 2020	0.7452	0.8613	0.8865	----	0.9563

Massachusetts roads dataset

This dataset is derived from the work of (Mnih & Hinton, 2010). It consists of 1171 aerial images of the state of Massachusetts. Each image is 1500×1500 pixels in size, covering an area of 2.25 square kilometers. We randomly split the data into a training set of 1108 images, a validation set of 14 images and a test set of 49 images. The dataset covers a wide variety of urban, suburban, and rural regions and covers an area of over 2600 square kilometers. The test set alone covers over 110 square kilometers. The target binary maps

(0:background, 1:road) were generated by rasterizing road centerlines obtained from OSM. A line thickness of 7 pixels and no smoothing was used in generating the labels. All imagery is rescaled to a spatial resolution of 1m/px.

Table A2.2. Metrics for Massachusetts Roads Dataset

Author	mIoU	F1_Score	Pixel Accuracy	AUC
Mnih & Hinton, 2010	0.6245	----	----	----
X. Gao et al, 2018	0.8210	----	----	0.8873
Abdollahi et al, 2021	0.8631	0.9251	----	----

Potsdam – Vaihingen Datasets

They are also called the Potsdam - Vaihingen 2D Semantic Labeling Contest, the dataset is created and maintained by the International Society of Photogrammetry and Remote Sensing (ISPRS) (Gerke et al., 2014). It contains 38 patches (of the same size) for the Germany city of Potsdam and 33 patches for the Germany city of Vaihingen respectively, each consisting of a true orthophoto, extracted from a larger orthomosaic, and a DSM. The ground sampling distance of both is 5 cm/px for the Potsdam and 9 cm/px for the Vaihingen. The orthomosaics come as TIFF files in different channel compositions, where each channel has a spectral resolution of 8bit:

- IRRG: 3 channels (IR-R-G)
- RGB: 3 channels (R-G-B)
- RGBIR: 4 channels (R-G-B-IR)

The DSM are TIFF files with one band; the grey levels (corresponding to the DSM heights) are encoded as 32-bit float values. Both products are defined on the same grid (UTM WGS84) and all images have dimension 6000 x 6000 pixels. The so-called normalized DSMs is also provided, that is, after ground filtering the ground height is removed for each pixel, leading to a representation of heights above the terrain. This data was produced using some fully automatic filtering workflow, without manual quality control. There is no error free data guarantee, purpose is to help researchers to use height data, other than the absolute DSM.

Since it is a contest, labelled ground truth is provided for only one part of the data. The ground truth of the remaining scenes will remain unreleased and stays with the benchmark test organizers to be used for evaluation of submitted results. Participants shall use all data with ground truth for training or internal evaluation of their method.

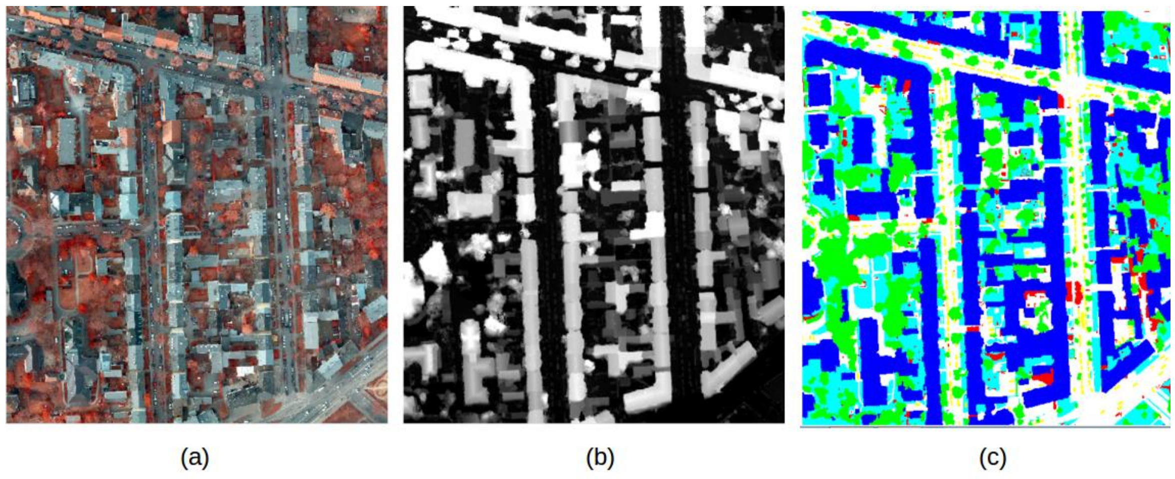


Figure A2.3. Potsdam-Vaihingen datasets. (a) RGBIR Image, (b) DSM, (c) Ground truth mask.

Table A2.3. Metrics for ISPRS Dataset

Author	mIoU	F1_Score	Pixel Accuracy
Song et al., 2020	----	0.7982	0.8721
Ziming Li et al, 2021	0.881	----	----
Marmanis et al., 2016	----	----	0.885

SpaceNet: A Remote Sensing Dataset and Challenge Series

CosmiQ Works, Radiant Solutions, and NVIDIA partnered to release SpaceNet as a Public Dataset on AWS and Challenge Series with the purpose of pursuing the innovation in the application of computer vision and deep learning to extract information from satellite imagery at scale (Van Etten et al., 2019). Today, map features such as roads, building footprints, and points of interest are primarily created through manual mapping techniques. SpaceNet partners believe that advancing automated feature extraction techniques will serve important downstream uses of map data, such as humanitarian and disaster response and other applications in both the public and private sectors. The first two SpaceNet challenges focused on building footprint extraction from satellite imagery. The third challenge addressed another foundational geospatial intelligence problem, road network extraction. The ability to create a road network is an important map feature (particularly if one can use this road network for routing purposes), while building footprint extraction serves as a useful proxy for population density.

The SpaceNet dataset was the first challenge that covered large areas including cities in Asia and Africa. SpaceNet is a large corpus of labeled diverse overhead satellite imagery organized in the following challenges:

- **Challenge 1 - Rio De Janeiro Building Footprints:** performed in 2016 aimed to extract building footprints from the DigitalGlobe WorldView 2 satellite imagery at

50cm resolution. Imagery consists of 8-band multispectral imagery at 1m resolution, as well as pan-sharpened red-green-blue (RGB) imagery at 50cm resolution. Released imagery was created from a mosaic of multiple images and covers 2544 square kilometers. The imagery was split into 400-meter tiles using utilities in python with 60% of the data released for training, 20% for testing, and 20% reserved for validation. For the Rio de Janeiro Area of Interest (AOI), over 300,000 building footprints were labeled semi-automatically and subsequently improved by hand. Any partially visible rooftops were approximated to represent the shape of the building. Adjoining buildings were marked individually as unique structures (i.e., each street address was marked as a unique building).

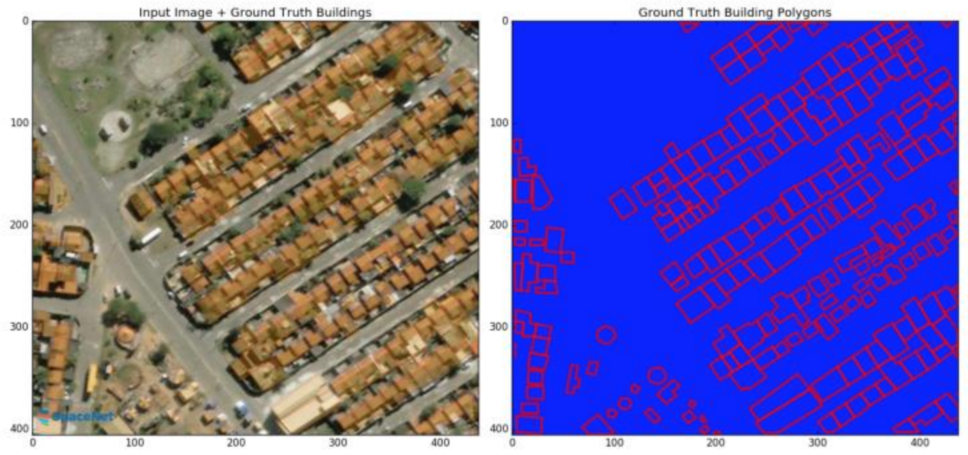

Input Image + Ground Truth Buildings
Ground Truth Building Polygons
SpaceNet
0
100
200
300
400

Figure A2.2. SpaceNet Challenge 1, Rio de Janeiro Building Footprints. Left RGB Satellite Image, Ground truth building masks.

Table A2.4 Metrics for Rio de Janeiro Buildings Footprint SpaceNet Dataset

Author	mIoU	F1_Score	Pixel Accuracy
Van Etten et al., 2019	----	0.26	----

- **Challenge 2 - Las Vegas, Paris, Shanghai, Khartoum Building Footprints:** It aimed to extract building footprints from the DigitalGlobe WorldView 3 satellite in a continuous image strip. The source imagery is distributed as a Level 2A standard product that has been radiometrically and sensor corrected and normalized to topographic relief using a coarse digital elevation model (DEM). It contains the original panchromatic band, the 1.24m resolution 8-band multi-spectral 11-bit geotiff, and a 30cm resolution Pan-Sharpened 3-band and 8-band 16-bit geotiff. The labeled dataset consists of 24,586 scenes of size 200 m × 200 m (650 px × 650 px) containing 302,701 building footprints across all four areas and are both urban and suburban in nature. The dataset was split 60%/20%/20% for train/test/validation. Each area is covered by a single image strip, which ensures that sun, satellite, and atmospheric conditions are consistent across the entire scene. Final product was then inspected against topology errors to ensure that building footprints were closed and polygons did not overlap.

Table A2.5. Metrics for Las Vegas building footprint SpaceNet Dataset

Author	mIoU	F1_Score	Pixel Accuracy
Van Etten et al., 2019	----	0.693	----

- **Challenge 3 - Las Vegas, Paris, Shanghai, Khartoum Road Extraction:** It used the imagery from Challenge 2, and because the competition was designed to enable the creation of routable road networks, a labeling schema based on OSM guidelines was established to ensure ground truth that would be usable by open-source routing tools. In addition to the digitizing of road- centerlines, four other attributes were recorded: 1) road type, 2) surface type, 3) bridge and 4) lane number. A GIS team at Radiant Solutions fully annotated each road centerline within 7 pixels, and all reported dangling roads or missed connections were corrected.

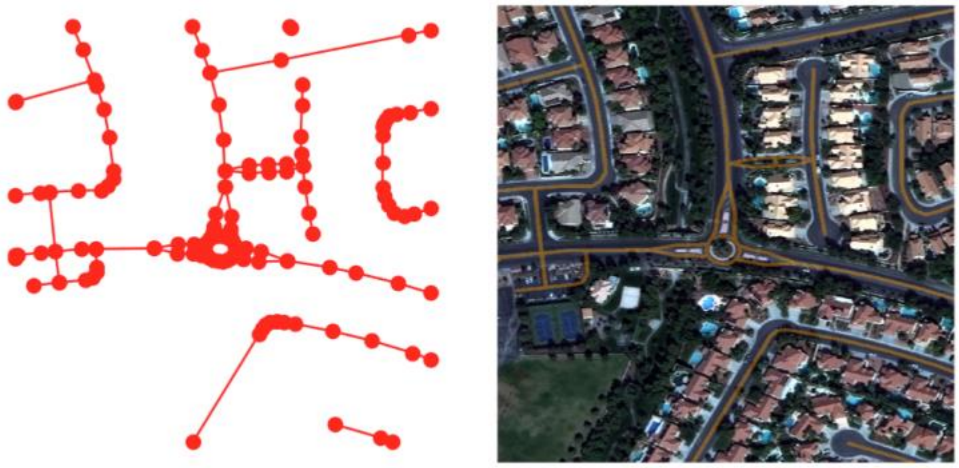


Figure A2.3. SpaceNet Challenge 3. Left: Las Vegas GeoJSON Road label. Right: RGB image overlaid with road centerlines (orange).

Table A2.6. Metrics for Road Extraction SpaceNet Dataset

Author	mIoU	F1_Score	Pixel Accuracy	APLS Metric
Van Etten et al., 2019	----	----	----	0.6663
Batra et al, 2019	0.6245	----	----	0.6077

DeepGlobe

Also presented as DeepGlobe Satellite Image Understanding Challenge. The datasets created and released for this competition may serve as reference benchmarks for future research in satellite image analysis (Demir et al., 2018). It is structured around three different satellite image understanding tasks:

- **Road extraction:** This is a binary segmentation problem to detect all the road pixels in each area. The evaluation is based on the accuracy of the road pixels.

The training data for Road Challenge contains 6226 satellite imagery in RGB, with size of 1024x1024. The imagery has 50cm pixel resolution, collected by Digital Globe’s satellite. The dataset contains 1243 validation and 1101 test images, but no masks. Each satellite image is paired with a mask image for road labels. The mask is a grayscale image, with white standing for road pixel, and black standing for background. File names for satellite images and the corresponding mask image are id _sat.jpg and id _mask.png, where id is a randomized integer. The labels are not perfect due to the cost for annotating segmentation mask, especially in rural regions. In addition, data providers intentionally didn't annotate small roads within farmlands.

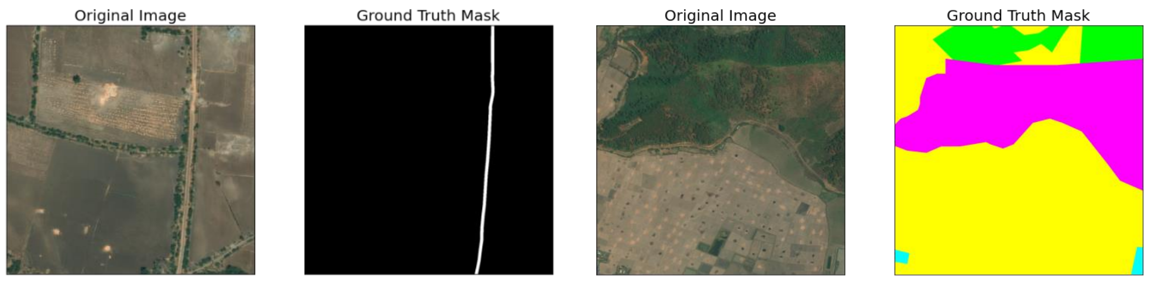
Table A2.7. Metrics for Road Extraction DeepGlobe Dataset

Author	mIoU	F1_Score	Pixel Accuracy	APLS Metric
Demir et al, 2018	0.5450	----	----	----
Batra et al, 2019	0.6472	----	----	0.6871

- **Building detection:** It is formulated as a binary segmentation problem to localize all building polygons in each area. The evaluation will be based on the overlap of detected polygons with the ground truth. It uses the SpaceNet Building Detection Dataset.

- **Land cover classification:** This problem is defined as a multi-class segmentation task to detect areas of urban, agriculture, rangeland, forest, water, barren, and unknown. The evaluation will be based on the accuracy of the class labels.

The training data for Land Cover Challenge contains 803 satellite imagery in RGB, size 2448x2448. The imagery has 50cm pixel resolution, collected by DigitalGlobe's satellite. The dataset contains 171 validation and 172 test images, but no masks. Each satellite image is paired with a mask image for land cover annotation. The mask is a RGB image with 7 classes of labels, using color-coding (R, G, B). File names for satellite images and the corresponding mask image are id _sat.jpg and id _mask.png, where the id is a randomized integer.



(a) (b)

Figure A2.4. DeepGlobe Satellite Image Understanding Challenge. (a) Road extraction. (b) Land cover classification.

Inria Aerial Image Labeling Dataset

The Inria Aerial Image Labeling covered 810 km2 (405 km² for training and 405 km² for testing), with orthorectified color imagery of 30 cm resolution in various European and American cities (Maggiori et al., 2017). The aerial images cover dissimilar urban settlements, ranging from densely populated areas, e.g., San Francisco's financial district to alpine towns, e.g., Lienz in Austrian Tyrol. Ground truth data has two semantic classes: building and not building, it is publicly disclosed only for the training subset. Inria addressed model portability between areas as some cities were included only in training data and some only in testing data. Instead of splitting adjacent portions of the same images into the training and test subsets, different cities are included in each of the subsets. For example, images over Chicago are included in the training set, and not on the test set, and images over San Francisco are included on the test set, and not on the training set. The ultimate goal of this dataset is to assess the generalization power of the techniques: while Chicago imagery may be used for training, the system should label aerial images over other regions, with varying illumination conditions, urban landscape and time of the year. The dataset was constructed by combining public domain imagery and public domain official building footprints in 2017.

The training set contains 180 color image tiles of size 5000×5000, covering a surface of 1500 m × 1500 m each. There are 36 tiles for each of the following regions: Austin, Chicago, Kitsap County, Western Tyrol, Vienna. The format is GeoTIFF (TIFF with georeferencing, but the images can be used as any other TIFF). Files are named by a prefix associated to the region (e.g., Austin or Vienna) followed by the tile number (1-36). The reference data is in a different folder and the file names correspond exactly to those of the color images. In the case of the reference data, the tiles are single-channel images with values 255 for the building class and 0 for the not building class. The test set contains the same number of tiles as the training set (but the reference data is not disclosed). There are 36 tiles for each of the following regions: Bellingham, WA, Bloomington, IN, Innsbruck, San Francisco, Eastern Tyrol.

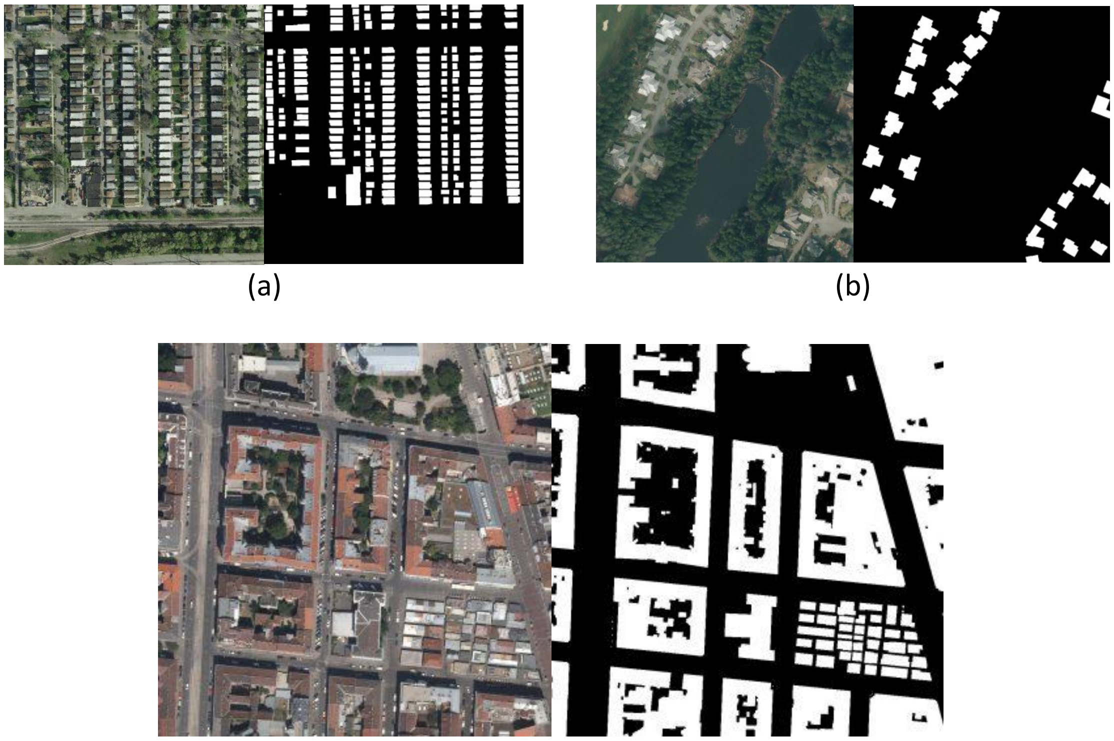


Figure A2.5. Inria Aerial Image Labeling Dataset. (a) Chicago. (b) Kitsap County, WA. (c) Vienna, Austria.

Table A2.8. Metric for Inria Dataset U-Net

Author	mIoU	F1_Score	Pixel Accuracy	OA
Maggiori et al., 2017	0.7339	0.8452	0.8716	0.9420
Pan et al., 2019	0.7752	----	----	0.9660
Xie et al., 2020	0.7943	0.8846	----	0.9639

DroneDeploy Dataset

The DroneDeploy Segmentation Benchmark challenge was released as partnering with Weights & Biases to evaluate and encourage state-of-the-art machine learning on aerial drone data. The dataset consists of aerial orthomosaics and elevation images (*Introducing DroneDeploys Aerial Segmentation Benchmark | DroneDeploy*, n.d.). These have been annotated into 6 different classes: Ground, Water, Vegetation, Cars, Clutter, and Buildings. The resolution of the images, captured from drones, is approximately 10cm per pixel which gives them a great level of detail. The images are RGB TIFFs, the elevations are single channel floating point TIFFs, where each pixel value represents elevation in meters, and finally the labels are PNGs with 7 colors representing the 7 classes. Python scripts to clip images, elevation, and labels into chips of 300x300 pixels are provided. Color code used is BGR (Blue, Green, Red), (075, 025, 230) : BUILDING, (180, 030, 145) : CLUTTER, (075, 180,

060) : VEGETATION, (048, 130, 245) : WATER, (255, 255, 255) : GROUND, (200, 130, 000) CAR, (255, 000, 255) : IGNORE. U-Net architecture is proposed as the base implementation, and dataset providers suggest that there is immediate opportunity to experiment with: Data augmentation, Hyperparameters, Post-processing, Chip size, Model architecture, and the use of elevation tiles.

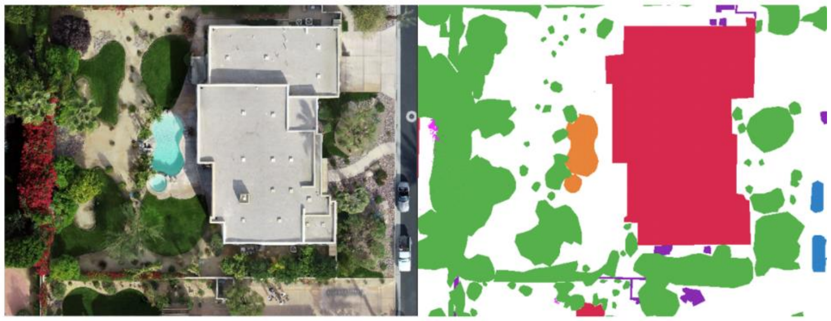


Figure A2.6. DroneDeploy Dataset. Left: Image chip. Right: Labels.

Table A2.9. Metric for DroneDeploy Dataset U-Net

Author	mIoU	F1_Score	Pixel Accuracy
DroneDeploy Benchmark 2019	0.525	----	----
Blaga et al, 2020	0.6178	----	----

Toronto City dataset

The TorontoCity benchmark covers the full greater Toronto area (GTA) with 712.5km2 of land, 8439km of road and around 400, 000 buildings (S. Wang et al., 2017). The benchmark provides different perspectives of the world captured from airplanes, drones and cars driving around the city. We use aerial images with full coverage of the GTA taken in 2009, 2011, 2012 and 2013. They are orthorectified to 10cm/pixel resolution for 2009 and 2011, and 5 and 8cm/pixel for 2012 and 2013 respectively. This contrasts satellite images, which are at best 50cm/pixel. Our aerial images have four channels, i.e., RGB and Near infrared, and are 16-bit resolution for 2011 and 8 bit for the rest. As is common practice in remote sensing, projection of each image is the Universal Transverse Mercator (UTM) 17 zone in the WGS84 geodetic datum and tiled the area to 500 × 500m2 images without overlap. Images are not true orthophotos and thus facades are visible. The dataset also exploits airborne LIDAR data captured in 2008 with a Leica ALS sensor with a resolution of 6.8 points per m2. The total coverage is 22 km2. All the points are also geo-referenced and projected to the UTM17 Zone in WGS84 geodetic datum. Most of the dataset's ground truth maps were created by the City of Toronto. Buildings have height estimates, the tallest building with 443m of elevation, the mean height of each building is 4.7m, and the mean building area is 148m2. In contrast, the largest building has an area of 120, 000m2. Dataset contains very accurate polylines representing streets, sidewalks, rivers and railways within the GTA. Each line segment is described with a series of attributes such as name, road category and address number range. Road intersections are explicitly encoded as intersecting points between polylines. Center lines define the connectivity (adjacency) in the street network. Road curbs define the road boundaries are also available and describe the shape of roads.

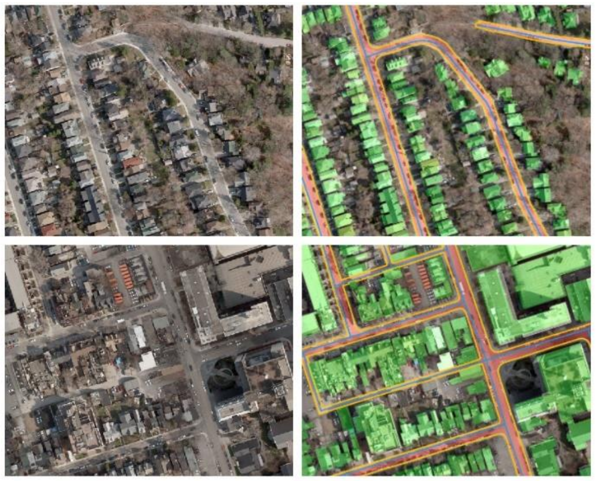


Figure A2.7. TorontoCity Dataset. Left: Images. Right: Labels for roads center lines and curbs, and buildings.

Table A2.10. Metric for TorontoCity Dataset U-Net

Author	mIoU Road	mIoU Building	Pixel Accuracy	F1_Score
S. Wang et al., 2017	82.72%	78.80%	----	----

Open Cities dataset

Open Cities AI Challenge dataset has the purpose of segment buildings in African cities from aerial imagery and advance responsible AI ideas for disaster risk management. This means addressing barriers to applying ML in African urban environments and adopting best practices in geospatial data preparation to enable easier ML usage. The competition dataset has over 400 square kilometers of high-resolution drone imagery and 790K building footprints (DrivenData, n.d.). It is sourced from locally validated, open-source community mapping efforts from 10+ urban areas across Africa. The dataset was released in 2020 and main prize is for best open-source semantic segmentation model of building footprints from drone imagery that can generalize across a diverse range of African urban environments, spatial resolutions, and imaging conditions. Imagery is collected by commercial drones at much higher resolution and under diverse environmental conditions. In this dataset, buildings are more densely situated, and diverse in shape, construction style, and size compared to the ones seen in Europe, America or Asia. Furthermore, since this dataset is crowdsourced and community-driven data labeled, it may differ greatly in what base imagery layers are used, workflow, data schema, and quality control, requiring models that are robust to more label noise. Figure A2.8 shows examples of errors in labels.

Dataset was manually labeled by different people by applying best practices in cloud-native geospatial data processing, i.e., using Cloud-Optimized GeoTIFFs [COG] and SpatioTemporal Asset Catalogs [STAC]. STAC Browser tool allows to rapidly visualize and access any training data asset in a web browser, despite individual image files being up to several GBs.

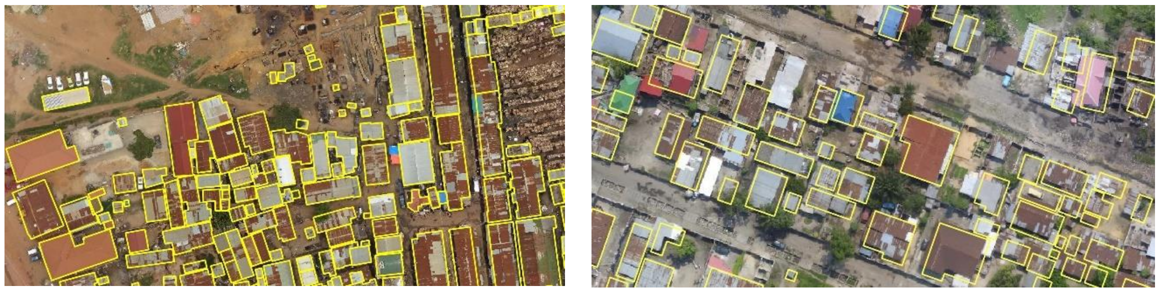


Figure A2.8. Open Cities Dataset. Quality of hand-drawn building footprint labels (alignment and completeness) can vary across or within image scenes. Examples from Challenge training dataset for Kampala, Uganda (left) and Kinshasa, Democratic Republic of the Congo (right).

Table A2.11. Metric for Open Cities Dataset U-Net

Author	mIoU	Precision	Recall	F1_Score
Drivendata.com	0.86	92%	93%	----