Universitä dell' Aquila

Segmentation of streets and buildings using U-Net from satellite images

Dipartimento di Ingegneria e Scienze dell'Informazione e Matematica
Corso di Laurea in Computer Science

Candidato

Jayasurya Arasur Subramanian
Matricola 267412

Relatore
Prof. Pasquale Caianiello

Correlatori
Dr. T.Senthil Kumar
Dr. L. Ravikumar

Anno Accademico 2020/2021

SEGMENTATION OF STREETS AND BUILDINGS USING U-NET FROM SATELLITE IMAGES.

A PROJECT REPORT

Submitted by

JAYASURYA A S

(CB.EN.P2CSE18007)

In partial fulfilment of the requirements for the degree of

MASTER OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

AMRITA SCHOOL OF ENGINEERING

AMRITA VISHWA VIDYAPEETHAM

COIMBATORE – 641 112

MARCH 2021

AMRITA VISHWA VIDYAPEETHAM

AMRITA SCHOOL OF ENGINEERING, COIMBATORE

ABSTRACT

Over a hundred thousand people are dying in road accidents in India every year among those 30% of accidents take place due to roads and buildings which are constructed without following any government regulations. To minimize road accidents we need to identify the buildings which are illegally constructed (without following any government regulations) on the roadside. The aim of this paper is divided into two parts. Firstly, segmenting the roads and buildings from the satellite image and secondly, finding out the illegal construction of roads and buildings by matching the current image with the timed image which is authorized by the government authorities. In order to carry out the objective on a larger scale, an innovative method is evolved which combines a semantic segmentation neural network with residual learning and U-net. In addition, it is proposed a method for road and building extraction from a satellite image. The convolution neural network is built using the U-net architecture. The two main advantages of this network are (1) U-net is symmetric and (2) the skip connections between the down-sampling path and the up-sampling apply a concatenation operation instead of an addition operation when compared to standard Fully Conventional Neural network (FCN)-8. In this application, we used the Massachusetts Benchmark dataset for roads and building extraction. Essentially, in this research, a comparison carried out with other methods recently available (RESNET, SEGNET), this method greater advantage providing result even with not fully connected layers. Moreover, this network was found to equally good in terms of computational time.

Key terms: Road extraction, Building extraction, Convolutional Neural Network, U-Net, Convolutional layer, Max pooling layer, Epoch.

ACKNOWLEDGEMENT

I would like to express my gratitude to our beloved **Satguru Sri Mata Amritanandamayi Devi** for providing the bright opportunity at Univaq as a dual degree student, which has made this entire task appreciable.

I express my sincere gratitude to **Brahmachari Abhayamrita Chaitanya**, Pro-Chancellor, **Dr. P. VenkatRangan**, Vice-Chancellor, and **Dr. Sasangan Ramanathan** Dean Engineering of Amrita Vishwa Vidyapeetham for providing the opportunity to undergo this program.

I express my thanks to **Dr. (Col.) P.N.Kumar**, Chairperson of Department of Computer Science Engineering, Amrita Vishwa Vidyapeetham, for his valuable help and support during our study. I express my gratitude to **Prof. Henry Muccini**, Department of Computer Science and Mathematics**,** University of L'Aquila for his guidance, support, and supervision.

I wish to record my deep sense of gratitude and profound thanks to my research supervisor and project coordinator **Prof. Pasquale Caianiello**, Associate Professor, Department of Computer Science and Engineering for his constant support, inspiring guidance, encouragement with my work during all stages, to bring this thesis into fruition.

I also thank my project correlator, **Dr. Senthil Kumar**, senior researcher, Department of computer science and mathematics, University of L'Aquila, Italy and **Dr. L. Ravikumar**, Scientist 'SG' Division Head, Hardware In Loop Simulation (HILS), UR Rao Satellite Center, Bangalore for their support and guidance in this thesis. I am also grateful to **Dr. Anantha Narayanan V.**, Associate Professor, Department of Computer Science and Engineering for his support and guidance as my class advisor and to all panel members for their feedback and encouragement which helped me to progress in the project. I also thank the faculty and non-teaching staff members of the Department of Computer Science and Engineering for their valuable support throughout the course of research work.

I would like to express my gratitude to all who have helped me directly or indirectly in doing the thesis. Last but not least, I thank the Almighty for all his blessings showered on us during the tenure of the thesis.

JAYASURYA A S

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATION

RBM	-	Restricted Boltzmann Machine
SVM	-	Support Vector Machine
SIFT	-	Scale Invariant Feature Transform.
KNN	-	k-Nearest Neighbor
CNN	-	Convolutional Neural Network
FCNN	-	Fully Convolutional Neural Network
SURF	-	Speed Up Robust Feature

CHAPTER I

INTRODUCTION

There are more than three hundred thousand accidents taken place in India every year, in that more than a hundred thousand people are dying [1] because of that. In those, 30\% of accidents take place due to faulty roads and by buildings around the roads which are illegally constructed. If all the illegally constructed buildings or detected early we can avoid the danger of accidents, and prevent human life from falling. To do that we are using the medium of the satellite to get the image of the urban and rural area to find any illegally constructed building is present in that area. The advantages of using satellite images (remote sensing) are its spatial resolution, spectral characteristics, Temporal characteristics, sensor sensitivity, program history, Image surface area, Multi-angle capability, Tasking, Price and licensing, Browse options, and processing options. To do so CNN (convolutional Neural Network) and U-net came to the rescue and help us a lot.

In current times CNN plays a major role in object detection and image classification [2]. In deep learning, we are having a lot of sub-branches, among those CNN and recurrent neural networks are one of the major things. This neural networks main motive is to classify the image into different kind of categories by learning from its training data. Apart from that, they learn from their layers which includes their own max-polling, convolutional, and fully connected layers.

Mnih and Hinton [3] are the first to apply deep learning techniques in road extraction process. They propose Restricted Boltzmann machine (RBMs) to detect road area from ariel image, Then they use a support vector machine (SVM) and K-Nearest Neighbor (KNN) for the extraction of roads, which gave the precision of 0.8-0.9. Initially, U-Net was introduced to segment medical images, but in recent years for gaining better accuracy U-Net was introduced to satellite image segmentation also.

It consumes more time and manpower to identify the buildings and roads are illegally constructed or not through the field visit process. So we are moving to the satellite images. In that also it's not that easy, sometimes we will get images with high noise and with low contrast which makes our process tougher. Not only that the artifact like vegetation and vehicles also should be taken care of to improve the object detection process. Motivated by deep learning

[4] and U-Net [5], we use deep residual U-Net, which takes advantage of both deep learning and U-Net architecture. Nowadays these classification techniques are improving with providing high accuracy on the results. And those techniques play a major role in the remote sensing industries also.

CHAPTER 2

RELATED WORK

Semantic segmentation literature is made of two categories, classical method, and deep learning method. The classical method or traditional method is the approach evolved before the Deep learning. The classical method includes classifiers like SVM, k-nn, etc. Semantic segmentation becomes a very difficult task with these conventional methods because their main focus is on SHIFT, HoG, SURF, forest classifiers [39],[40], and conditional random fields.[41]. While categorizing the patch type these methods clustered or group the pixels. Even though, to perform pixel-level classification they mostly depend upon hard crafter features than the structure of the data. In other case, experts predict that the conditional random field is the right approach for the structured prediction problem.

In upcoming days the sight of researchers is turning towards the automatic road and building detection using image segmentation, which is a quite modern field of research. Due to that only a limited papers and journals have been issued. Such work gave a clear understanding of my area of research and also helped to narrow down the options and choose a suitable model for the segmentation process. We can site [31],[32],[33]. SVM classification algorithms were used in the first two articles for the extraction of roads and buildings from VHR (very high resolution) satellite images. In each case, their accuracies were 74% and 83% respectively. One of the recent papers is released based on the INRIA Aerial Image dataset, which uses different architecture like FCN [34], and SegNet [35] to segment the building block. We just take note of that because as a counterpart INRIA data are made up of VHR images.

One of the developing networks is U-Net a specific type of fully conventional neural network, which received a lot of interest from the researchers for the biomedical image segmentation because of the usage of the reduced dataset. It has been proven that it also be very effective in the pixel-wise categorization of satellite images [36]. Our model is purely based on U-Net [37]. In that paper, the author developed a U-Net model specifically for biomedical images. As an extension of that paper another article [38] describes the usability of U-Net in the 3dimetional medical images. The accuracy of that is 75% in both cases. Anyhow 3D images are beyond our scope according to our problem statement.

In [39] they had altered the original U-Net architecture and used the altered one for the land cover maps. As a result, it gives the land cover maps which improves the pixel-based categorization generated by RF for the class where the pixel context is crucial. By doing that it produces a more geometrically accurate result than the original U-Net architecture model.

CHAPTER 3

REMOTE SENSING

Remote-sensing is used for the acquisition of data on the planet's surface without doing any interaction with that. It is one of the growing environment which includes a wide area from public policy to coerce to science. This field evolves from the interpretation of aerial images or photographs to the analysis of satellite images. It can be done from the local studio to the global analysis. Advanced sensors and digital computing are used for the achievement of that goal. Today remote sensing data also gives information on reflected, emitted, and transmitted energy from all parts of the electromagnetic spectrum.

For the process of monitoring and mapping biodiversity, remote sensing is considered the main tool across the globe. Remote sensing data provides information on landscape characteristics, ecosystem's structural properties, biodiversity spatial components, human-induced vegetation patterns, and impacts of various disturbances on them. Data obtained through remote sensing is used for human environment analysis and modeling purposes. We obtain a good knowledge of lifestyle and biodiversity when we integrate remote sensing data with the information on the human environment and lifestyle. For biodiversity studies range of remote sensing data is needed, which includes time series for acquiring hyperspectral data on 3D structures.

3.1 Different Platforms of Remote Sensing

The structure on which the instruments of remote sensing are mounted is called the different platforms of remote sensing. The number of attributes is determined by which a particular sensor is hosted. These attributes include sensors in form of an object of interest and image acquisition with the timing when it is acquired. The remote sensing platform can be categorised into three broad categories which are

1. Ground based
2. Airborne and
3. Satellite.

3.1.1 Ground Based Remote Sensing

A large collection of ground-based platforms are let to use in the remote-sensing process. Hand-held gadgets, trivets, towers, and cranes are the most common ones among them. The ground-based instruments are used to calculate the quality and the quantity of the light beam emanating from the sun or for the categorization of the close-range items. For the research, sensors calibration and quality control laboratory instruments are used. The main reason to learn from the laboratory work is for better utilization and the identification of different materials. One of the other largely used instruments for research purpose are field instruments. It is usually hand-held or mounted on a tripod. Constant ground platforms like cranes are mainly used for monitoring atmospheric changes. Apart from that, they are also used for the observation of continental features. These permanent ground platforms are used for research purpose, that needs long term stable platform. For example to monitor the forest or any landmass towers can be built on-site so that the range of measure can be taken from the forest ground.

3.1.2 Airborne Based Remote Sensing

In the early remote sensing process, airborne platform are the only non-ground platform which are used for the remote sensing process. A camera carried up in the air balloon captured the first aerial images. Although miniature helium balloon bearing expendable surveys are still used for some meteorological study, helium balloon are seldom used these days because they are not very stable and the course of flying is not always predictable. Airplanes are the most popular airborne platform now a days. For remote sensing application, nearly every type of civilian and military aircraft are used. Easy, low-cost airplane can be used as platforms when the sensor's altitude and stability requirements aren't too stringent. However, as the need for greater device stability or greater heights emerges, more advanced planes may be needed. Based on their altitude limitation, airplanes are classified into 3 groups (low, mid, and high). The higher an airplane climb, the more steady it is as a platform, but it is much more expensive to manage and retain.

The altitude with high oxygen or pressure is the paradise for the low-altitude aircraft. Those aircraft will fly below altitude of twelve thousand five hundred feet above sea level. They're useful for collecting high-level spatial resolution data that's restricted to specific

region. Helicopters are commonly used for low-altitude application that necessitate the capability to fly. Helicopters are costly to run, so they are normally only used when absolutely necessary. Ultralight aircraft are a rising segment of the aviation industry.

The height maximum for mid-altitude aircraft is less than thirty thousand feet above sea level. A amount of turboprop planes are included in this group. Since there is less noise at higher altitudes, stability is increased. When steadiness is more essential, and it is required or desirable to obtain metaphors from a larger distance than is possible with low-altitude planes, this class of plane is used. These aircraft can cover a wider area in less time than low-altitude platforms.

High-altitude aircraft can achieve heights of more than thirty thousand feet above sea level. This type of plane is operated by jet engines and is operated for specific missions like atmospheric studies, study to simulate satellite platforms, and other applications that involve a high elevation platform. High-altitude planes are useful for obtaining wide areas of coverage with lower spatial resolution. Remote control aircraft, also known as drones, are another type of aircraft that has been around for a long time. Remotely operated aircraft are commonly used in circumstances where flying is too risky. The military has made heavy use of them.

3.1.3 Satellite Remote Sensing

The most steady platform is a space-based satellite. More than a one hundred remote sensing satellites have now have being deployed, with more have being send each year. The Space Shuttle is a one-of-a-kind spacecraft that can be used for a range of missions while also acting as a remote sensing satellite. The orbital geometry and timing of satellites can be used to identify them. Geostationary, equatorial, and Sun synchronous orbits are the most common for remote sensing satellites. A geostationary satellite rotates at the same rate as the Earth (24 hours), so it still passes over the same spot on the globe. Geostationary orbits are used by many communications and weather satellites, with several of them positioned over the equator. A satellite in an equatorial orbit orbits Earth at a low angle among the orbital plane and the equatorial plane. The Space Shuttle flies in an equatorial orbit with a 57-degree predisposition. Every satellite has its own orbit, which is matched to the capability and purpose of the sensors it carries. The altitude of orbits, as well as their own orientation and rotation in relation to the earth, will vary.

The Sun's synchronic satellites have orbits with extreme angles of inclination passing almost on the poles. The satellite's paths are planned such that it crosses over the equator at the same local sun time every time. On all of its orbits, the satellites hold the same relative location with the earth. Several remote sensing satellites are Sun synchronic, maintaining constant solar illumination conditions throughout the year. Since a Sun synchronous orbit does not move directly on the poles, data for the severe polar zones is not always possible satellite to obtain. The frequency at which a satellite sensor can obtain information from the Planet as a whole depends on its sensor and orbital characteristics. For many remote-sensing satellites, the total coverage frequency ranges from twice daily to once every sixteen days.

Altitude is another orbital characteristic. The Space Shuttle has a low orbital altitude of 300 km, while other common remote sensing satellites typically have higher orbits ranging from 600 to 1000 km. Most remote sensing satellites have been designed to transmit data to ground receiving stations located around the world. In order to receive data directly from a satellite, the receiving station must have a satellite viewing line. If there are not enough designated receiving stations around the world, any given satellite may not easily have a direct view of the station, leading to potential data discontinuity problems. In order to deal with this problem, the data can be stored temporarily on the satellite and then downloaded after contact with the receiving station. Another alternative is to relay data via the TDRSS (Tracking and Data Relay Satellite System), a geosynchronous (geostationary) communication satellite network used to relay data from satellites to ground stations.

3.2 Different Platforms of Remote Sensing

The initial form of remote-sensing commenced in the 1860s, even prior to the Wright brothers, who initially flew the aircraft. Using balloons and kites, the geographers take pictures of the ground from sky to capture a large area from the ground. With the launch of aircraft, arial photography may possibly capture pictures from much higher upwards. These Days, the altitude of airborne photography extends from a short distance above the ground to a height of just over 60,000 feet. The lower altitude of the photograph, the more details it captures, which means that fine particulars will be covered at maximum height, but a broader region and the relationship among elements will be displayed. As the year passes, people began to move towards satellite remote sensing. The main reason for moving to satellite remote sensing is its

high speed, time resolution, level of detail, etc. The following table helps us to understand the differences and advantages of satellite remote sensing over Aerial photography.

SATELLITE REMOTE SENSING	AERIAL PHOTOGRAPHY
High speed, temporal resolution.	Slow in speed, need more time to cover an area.
Level of details:- 50 cm in Geo Eye 1.	Level of details:- May go up to 2.5cm.
Weather condition:- Cloud is big hindrance in optical data.	Weather condition :- May work in high cloud and thin cloud.
Type of Data:- Less as each new data type needs new satellite launch.	Type of data:- High as new data type needs new sensors to be mounted on the aircraft.
Location :- Cross boarder images and large swath image can be easily taken.	Location:- Cross boarder images cannot be taken without permission.
Post processing:- Easy because of signal image covering large area.	Post processing:- Tough due to large number of image covering small area.

Table 3.1. Satellite Remote Sensing vs Aerial Photography

We know that remote-sensing tools can be set up on a numerous platforms for viewing and targeting images. Although ground-based and air-based platforms may be used, satellites offer a large amount of remote-sensing images generally used these days. Satellites have numerous distinctive qualities that make them especially helpful for remote-sensing the surface of the planet. In this chapter, we will see how remote satellite sensing is done and what its advantages are. The elements for remote-sensing satellites may consist of photographic systems, electro-optical sensors, microwave or lidar systems. For applications profiting from the instantaneous coverage of various sensors, more than one sensing system can be installed on a single satellite. As an add-on to sensor systems, data recording, pre-processing and transmission equipment is often used.

3.3 Fundamental Sensors

There are numerous broad categories of essential sensor system types, such as passive vs. active, and imaging vs. non-imagery. Passive vs. active refers to the system's source of illumination; imagery vs. non-imagery refers to the data form. A variety of distinct sensors fit into these categories, which are not mutually exclusive.

3.3.1 Passive vs. Active Sensors

Passive sensors detect light that is reflected or spontaneously produced by surfaces of the objects. Solar energy is the greatest supplier of radiation lighting surfaces of the objects, and certain devices only detect it. Active sensors (such as radar and lidar systems) emit energy (provided by their own energy source) before assessing the energy restored after communicating with the surface. The accurate calculation of solar radiation striking the surface at the moment the studies are made is also needed when using data obtained by passive sensors. This data allows "atmospheric impacts" to be corrected, resulting in data or photographs that are more descriptive of the real surface characteristics..

Passive Sensor

Passive sensors are the most common kind of sensor used in vegetation remote sensing. This is due not only to the fact that passive sensor systems are typically simpler in design (built solely to absorb energy), but also to the fact that some wavelengths of the solar spectrum provide extremely useful information for plant observing and canopy properties. One of the most significant limitations of passive systems is that, in most situations, they need daylight to obtain valid and useful data. As a result, the deployment or acquisition of information from passive sensors is highly dependent on illumination (daytime, yeartime, latitude) and weather conditions, as cloud cover will disrupt the direction of solar radiation from the sun to the surface and then to the sensor.

Atmospheric effects can greatly alter signals detected by passive sensors, especially in the shorter wavelengths of the solar spectrum that are heavily scattered by the atmosphere. These effects can be mitigated (but not eliminated) by collecting data only in very bright and dry weather. To eliminate atmospheric effects from data collected by passive sensors, sophisticated atmospheric correction routines are now in operation. The mainly used passive sensors are

1. Photographic
2. Electro optic radiometer
3. Passive microwave system
4. Visible, infrared and thermal imaging system

Active Sensors

Active techniques deliver their own light energy that can be monitored. Some of the benefits of active systems over passive sensors are that they do not need solar surface lighting or perfect weather environments to collect valuable information. As a result, they can be installed at night-time or in environments of fog, clouds or drizzle (reliant on the wavelength of the method). The mainly used active sensors are

1. Radar
2. Lidar

3.3.1 Imaging vs. Non Imaging Sensors

A registered representation of the reflected or released radiation from an environment or object is referred to as remote sensing data. The reflected or released energy may be measured using imagery or non-imagery sensors. Data from imaging sensors can be analyzed to create an image of a region in which smaller portions of the sensor's entire vision are visually resolved (see discussion of pixels below). Non-image sensors are typically hand-held instruments that record only solo response value with no finer resolution than the sensor's entire field of view, allowing no image to be created from the input. Although a small area is usually involved based on the spatial resolution of the sensor, these single values are referred to as "point" data.

Each type of data, image and non-image, has its own set of applications. Nonimage data may be used to describe the reflectance of various materials in a wider scene and learn more about the interactions of electromagnetic energy and artifacts by providing information for a

particular (usually small) region or surface cover category. Based on spatial resolution and sampling images, image data may be used to examine spatial relationships, object shapes, and approximate physical sizes. Where spatial information (such as mapped output) is required, image data is preferable. The focus of this text is on image sensors and data.

Images created from remote sensing data may be analogue (like a photograph) or digital (like a video) (a multidimensional array or grid of numbers). Digital data can be interpreted by looking at values and doing computer-based equations, or it can be transformed into an image for visual representation. The detail in a scene is deciphered using image analysis. Image analysis used to be primarily based on subjective visual methods, but with the advent of computing technologies, numeric or digital processing has emerged as an efficient and popular interpretation method.

3.4 Sensors in use.

Satellite Remote sensing data shall be acquired by detecting and recoding the exposed or discharged energy and by managing, evaluating and directing that information to applications. In a large part of remote sensing, the procedure includes collaboration of incident radiation and targets of interest. This is characterized by the use of the imaging system, which involves the following elements. However, remote sensing involves the sensing of the energy emitted and the use of non-imaging sensors. The following image shows the seven elements and how these elements play a major role in remote sensing.

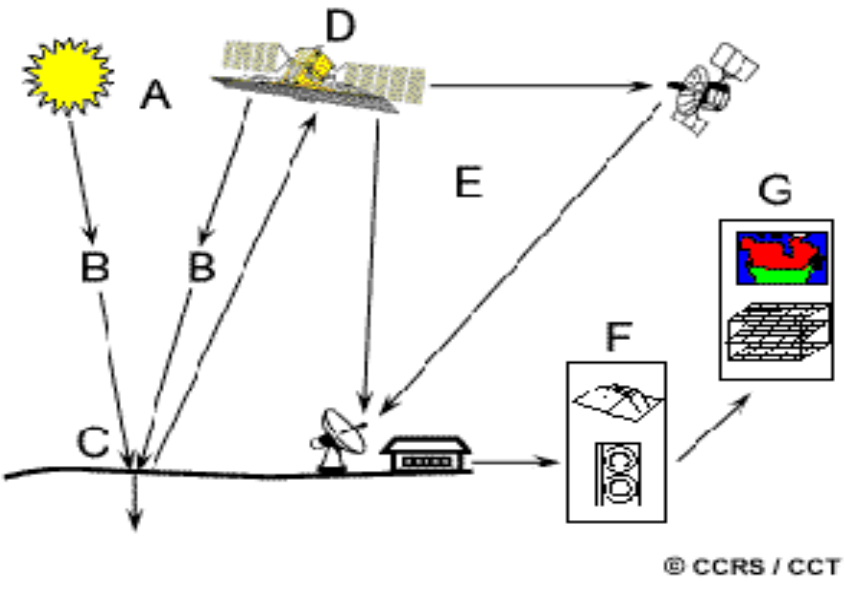


Figure 3.1 Components of Remote Sensing.

A	Energy source or illumination
B	Radiation and the atmosphere
C	Interaction with the target
D	Recording of energy by sensor
E	Transmission, reception and processing
F	Interpretation and analysis
G	Application.

Table 3.2 Components of Remote Sensing.

ENERGY SOURCE OR ILLUMINATION (A): The first prerequisite of remote sensing is to provide an energy supply that provides electromagnetic energy to the object of interest.

RADIATION AND THE ATMOSPHERE (B): As it flows from its source to its destination, the energy can come into communication with the environment. A second interaction can occur as the energy flows from the target to the sensor.

INTERACTION WITH THE TARGET (C): Energy deals with the target as it passes through the atmosphere and reaches it, depending on the properties of both the target and the radiations.

RECODING OF ENERGY BY THE SENSORS (D): We wanted a sensor (remote- not in contact with the target) to capture and record the electromagnetic radiation after it had dispersed from or released from the target.

TRANSMISSION, RECEPTION, AND PROCESSING (E): The energy from the sensor must be transmitted to a receiving and processing station, where the data is transformed into an image (hardcopy or digital).

INTERPRATION AND ANALYSIS (F): Visually, digitally, or electronically, the generated image is viewed to collect data about the illuminated target..

APPLICATION (G): The last step in the remote sensing phase is to use the information we've gleaned from the imagery to better interpret the objective, discover new information, or help solve a problem.

CHAPTER 4

QUANTUM GEOGRAPHIC INFORMATION SYSTEM

Quantum GIS or QGIS is an open-source desktop GIS that supports viewing, editing, and analysis of geospatial data. It allows users to edit and analyse the spatial data by composing and exporting geographical maps. Not only that it supports vector and raster data layers in which the vector data are stored as point, line, or polygon features. In our case, we use the line for the mapping of roads and polygon for the mapping of buildings. The software can georeference the image and supports multiple formats of the raster image. The definition of raster and vector data are briefly explained in the following chapters.

Personal geodatabase, MapInfo, shapefiles, dxf, PostGIS, coverages, and other formats are supported by QGIS. To axis the external data it uses Webservices including WMS (Web Map Service). It also combines with other open-source GIS packages like PostGIS, GRASS, GIS, and Map-Server. It also supports the plugins written in C++ or python which are geocoded using google geocoding API.

4.1 GIS

GIS stands for either Geographic Information system or Geographic Information Science, depending on what aspect of the term we are interested in. Geographic Information system typically refers to the software, like QGIS, we use to create spatial data and to investigate spatial relationships between that data. Geographic Information science is the framework we use to ask questions about the spatial relationship between data.

4.2 QGIS

From the QGIS website, "QGIS is a GNU General Public License-licensed, user-friendly Open Source Geographic Information System (GIS). The Open Source Geospatial Foundation has designated QGIS as an official project (OSGeo). It supports a wide range of vector, raster, and database formats and functions and runs on Linux, Unix, Mac OSX, Windows, and Android."

Let's unpack some of that.

QGIS is a desktop GIS. That means you get a program that opens up on our computer as a window with buttons we can click, forms we can fill out to do tasks, and it's generally a visual interactive experience (as opposed to command line programming in a terminal). Often this kind of interface is called a Graphical User Interface or GUI (often pronounced "gooey") for short.

QGIS is open source. That means the code is available for us to read or modify, should we choose to, but we don't have to. What's the advantage of this? It means anyone can make fixes if something is wrong or anyone can add new features. we don't have to wait for a paid developer to add something.

QGIS is an official project of the Open Source Geospatial Foundation (OSGeo). It is an open-source non-profit organization with the vision to foster global adaptation of open geographical technology. It is done by being an inclusive software foundation devoted to an open philosophy and community-driven development. It assists in public education, open-source geospatial projects, and organizing conferences.

4.3 Why QGIS?

QGIS is an open source, community-driven desktop GIS software that allows users to visualize and analyse spatial data in a variety of ways. There are many reasons to use QGIS, but here are a few:

- It's a robust, powerful desktop GIS
- Runs on all major platforms: Mac, Linux, & Windows
- Free of charge, all access (no paid add-ons or extensions)
- Frequent updates & bug fixes
- Responsive, enthusiastic community
- Integration with other geospatial tools & programming languages like R, Python, & PostGIS
- Access to analysis tools from other established software like GRASS and SAGA
- Native access to open data formats like geoJSON & GeoPackage

- Comes in a more than 40 languages, making it easier to work with a larger variety of collaborators
- Growing use by local, state, federal, and international governments

4.4 Data types- Vector vs. Raster vs. other type of Spatial data.

There are several data spatial data models that we may encounter as we work with spatial data. The two we will likely encounter most frequently are called vector and raster data. Figure 2 explains how the raster and vector data looks like.

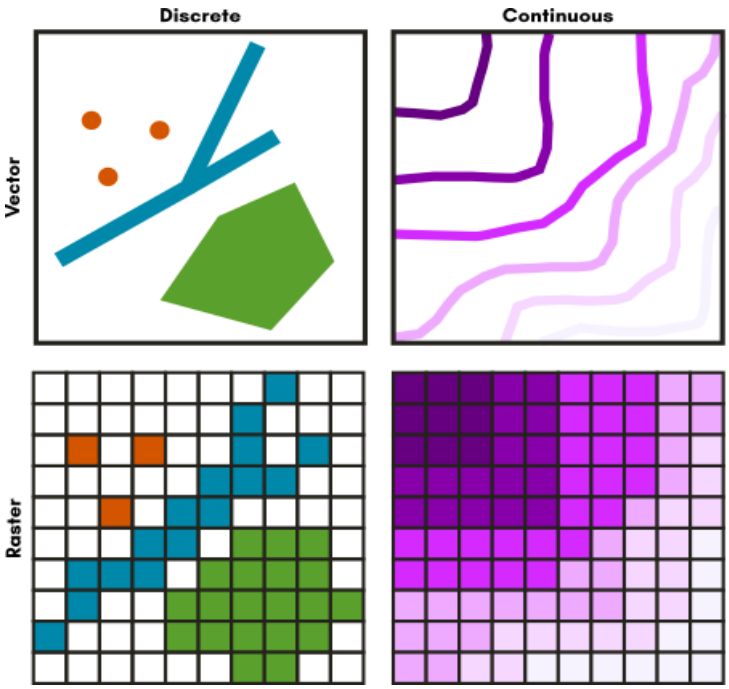


Figure 4.1 Vector and Raster data

4.5 Vector Data.

Vector data represents discrete objects in the real world with points, lines, and polygons in the dataset. If you were to draw a map to your house for a friend, you would typically use vector data - roads would be lines, a shopping centre included as an important landmark might be a rectangle of sorts, and your house might be a point (perhaps represented by a star or a house icon).

4.6 Raster Data.

Raster data represents continuous fields or discrete objects on a grid, storing measurements or category names in each cell of the grid. Digital photos are raster data you are already familiar with. If you zoom in far enough on a digital photo, you'll see that photo is made up of pixels, which appear as coloured squares. Pixels are cells in a regular grid and each contains the digital code that corresponds to the colour that should be displayed there. Satellite images are a very similar situation.

CHAPTER 5

U-NET ARCHITECTURE

In the last few years, the computer vision field had advanced rapidly because of deep learning. In that semantic segmentation is considered as the gem of all. Even though there is a lot of ways to solve this problem, U-Net got a spatial place. U-Net is a fully convolutional network that uses convolutional operation and max polling operation to solve the problem which gives an advantage over any other methods. Before going into U-Net we the application of semantic segmentation and where it is used.

5.1 Application of U-Net

5.1.1 Autonomous Vehicle

One of the complex robotic tasks is autonomous driving which requires the greatest perception, planning, and execution in a constantly changing environment. Since the Safety of the passengers involves in this process it should be presented with maximum precision. The role of semantic segmentation in an autonomous vehicle is to provide information on free space between vehicles, lane markings, and traffic signs.

5.1.2 Bio Medical Diagnosis

The role of semantic segmentation in biomedical image diagnosis is to reduce the time required for the diagnostic tests by making a radiologist to perform augment analysis through machines.

5.1.3 Geo Sensing

Since the classification and semantic segmentation problems are almost identical, we should also consider the semantic segmentation problem to be a classification problem. Each pixel is classified into a set of object classes in both cases. As a result, we can conclude that satellite imagery are used for land use planning. Land cover knowledge is more relevant in applications like studying areas of erosion and urbanization. Not only that for traffic management, monitoring of roads and city planning, roads and buildings detection is important which can be done through semantic segmentation. Other than this it is also used in a type of land recognition system and for the land cover classification system.

5.1.4 Precision Agriculture

The image vision techniques help agriculture by segmenting crops and weeds to trigger weeding actions. The amount of usage of herbicides can be reduced by using precision farming robots that sprays out in the field. These advantages will reduce the human monitoring of crops in the field.

5.2 Understanding Operations of U-Net

We need to know about the different operations of convolutional network before we go deep into the U-Net model. The upcoming chapters will help in the understanding of convolutional networks.

5.2.1 Convolutional Operation

CNN relies heavily on convolutional layers (Convolutional Neural Network). Convolution is the process of applying a filter to an input and inactivating it. The same filters applied to the input image produce a function map, also known as an activation map, which shows the intensity of the detected feature and its position in the input image. When training a dataset for a modeling query, convolutional neural networks automatically learn any number of filters palely. We can get extremely precise features that can be detected anywhere in the input images as a result of our work.

The two inputs to a convolutional operations are.

1. A 3D image of size (nin x nin x channels).
2. A set of kernels or feature extractor, each one of size (f x f x channels), where f is typically 3 or 5.

The convolution Arithmatic is shown in the Fig 5.1, A function map of size (nout x nout x k) is the product of a convolutional process . The following chapters explain the relationship between n-in and n-out.

$$n_{out} = \left\lfloor \frac{n_{in} + 2p - k}{s} \right\rfloor + 1$$

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

Figure 5.1 Convolutional Arithmetic

Visualization of the convolutional arithmetic can be seen in figure 5.2.

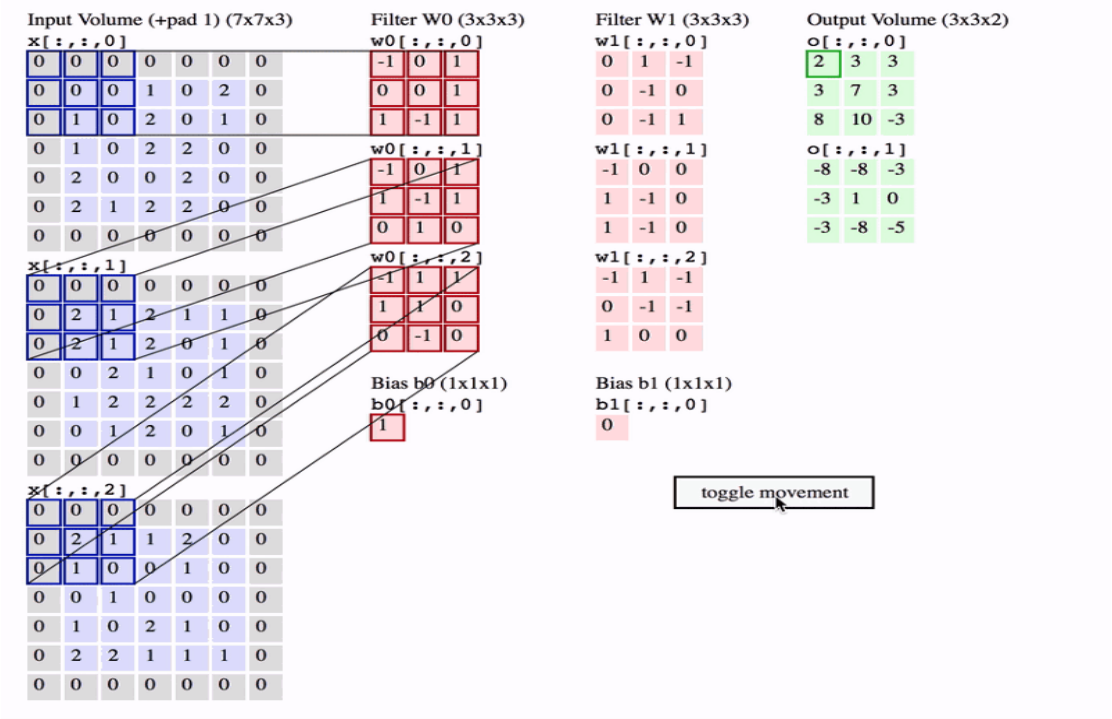


Figure 5.2 Convolutional Operation.

As seen in Figure 5.2, the convolution operation can be visualized. We have a 3x3x3 input with padding 0 and strides 2 here. As a result, the performance we're having is 3x3x2. The receptive field, which is an area in the input that the function extractor examines, is one of the most significant words used. The background, or receptive area, in Figure 3 is a 3x3 boxed region in the input that covers any given case.

5.2.2 Max Polling Operation

Pooling has the effect of lowering the scale of the feature diagram, resulting in fewer parameters in the network.

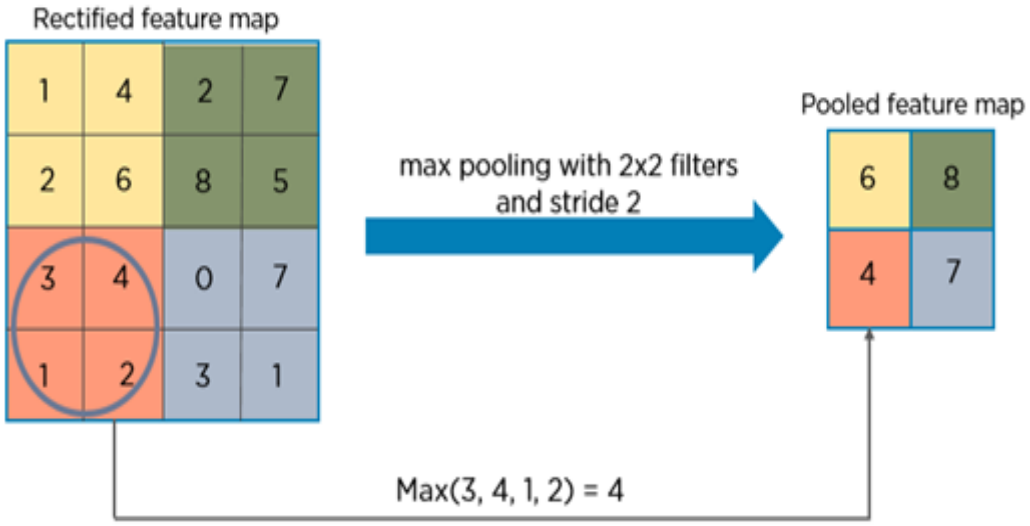


Figure 5.3 Max Polling Operation.

The feature map pooling operation is used to minimize the scale of the feature map. This is required to reduce the number of parameters in the network. It's achieved by picking the best pixel values from each 2x2 block of the input feature map, which will become our pooled feature map. Strides and the filter size are the two most critical parameters in max pooling operations. The aim of the maximum pooling operation is to exclude non-essential data and preserve only the most critical features from each field. Down-sampling refers to the process of reducing the size of an image using both pooling and convolutional operations. The max pooling process is depicted in Figure 5.3. The scale of the input image in Figure 5.3 is 4x4 before the pooling operation, and it is reduced to 2x2 after the pooling operation. The detail in the picture remains unchanged in both situations..

The convolutional operation helps to get the larger context of the image when it gets applied again. This explains that when we go deeper in the network the receptive field increases and the size of the image decreases. Consider the bellow LeNet5 architecture [6] which is shown in Figure 5.4 as an example. The height and width of the picture steadily decrease in a traditional convolutional network due to pooling. This smaller scale allows the filters to reflect on the deeper layers' broader contexts. As a result, as the number of filters or channels grows, it becomes easier to remove more complex features from the input image.

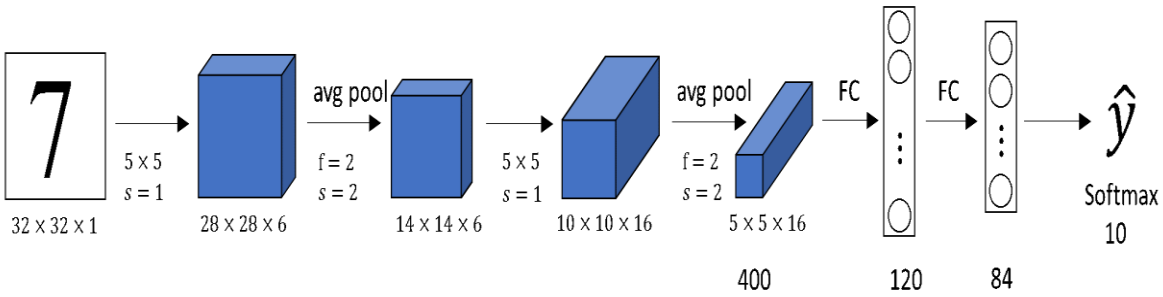


Figure 5.4 LeNet-5 Architecture.

5.2.3 Need for Up-Sampling

The semantic segmentation process produces a full high-resolution image with all pixels labelled. But, if we use a standard convolutional sheet, we'll lose "where" information and just keep "what" information. We do need the "where" detail in our situation, not the "when." In the case of segmentation, both "what" and "where" information is required. So, in order to obtain all pieces of material, we will up-sample. To put it another way, up-sampling is the process of transforming a low-resolution image into a high-resolution image in order to extract "where" data.

5.2.3 Transpose Convolutional

Transpose convolution or de-convolution is a procedure that performs up-sampling of an image. The up-sampling can be done by the following methods..

1. Nearest neighbour interpolation
2. Bi-linear interpolation
3. Bi-cubic interpolation

Figure 5.5 explains about how convolutional operation works. In the Figure 5.5 we have 4x4 matrix as our input and 3x3 as our kernel. We are applying convolution operation on 4x4 matrix with a 3x3 kernel, with the stride of 1 and no padding. As we do so we get the output of the 2x2 matrix.

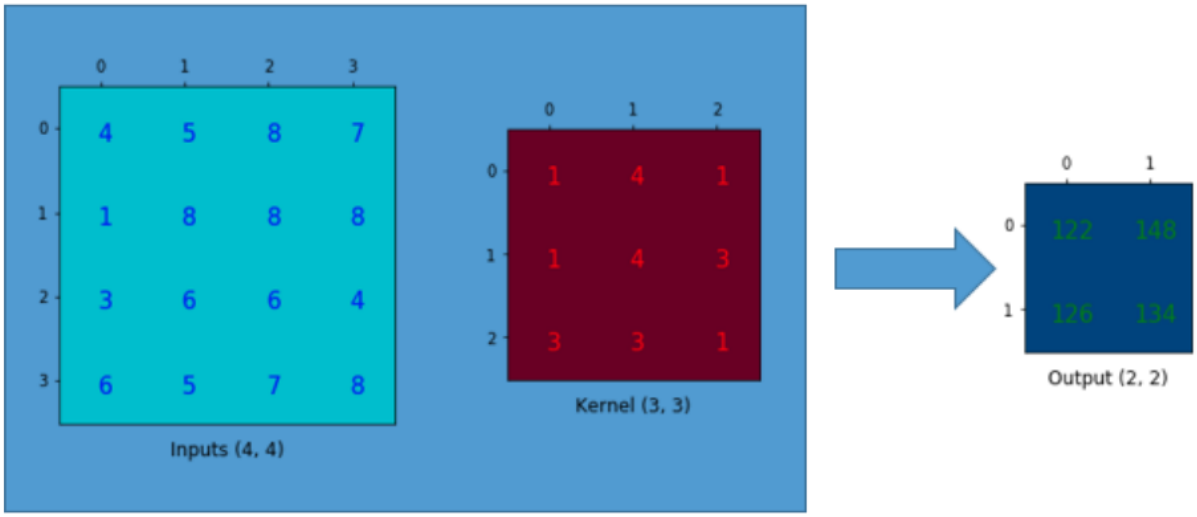


Figure 5.5 Transpose Convolutional Operation.

Between input matrix and kernel the sum of element-wise multiplication is calculated by the convolutional operation. We can calculate this only for 4 times because we have a stride of 1 and no padding. As the result, we get the output as a 2x2 matrix.

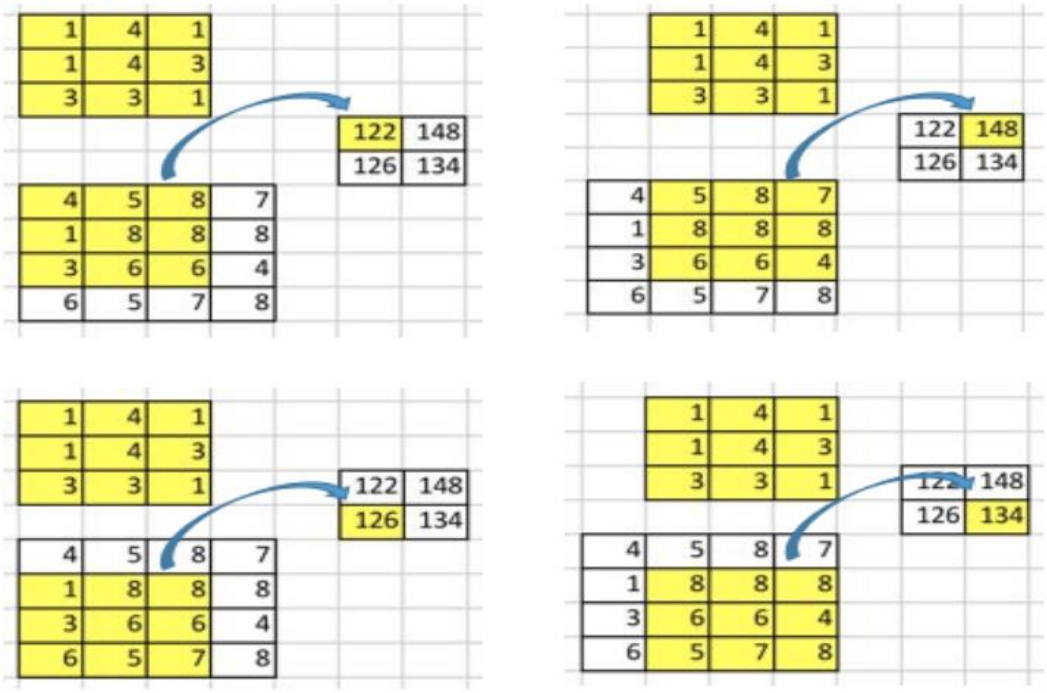


Figure 5.6 Sum of Element vs. Multiplication.

In such convolutional operations, there is positional connectivity between the input and output values. The input matrix's top left values have an effect on the output matrix's

top left values. The 3x3 kernel binds each of the input matrix's nine values to a value in the output matrix, forming a many-to-one relationship.

Now, if we want to do that in opposite direction, we need to associate a value of the output matrix to 9 values of the input matrix, which makes it a one-to-many relationship. Basically here we are going away towards the back of convolutional procedure, which is the heart of transpose convolution. In other words, we are up-sampling the 2x2 matrix to a 4x4 matrix. The process will be 1-to-9 relation.

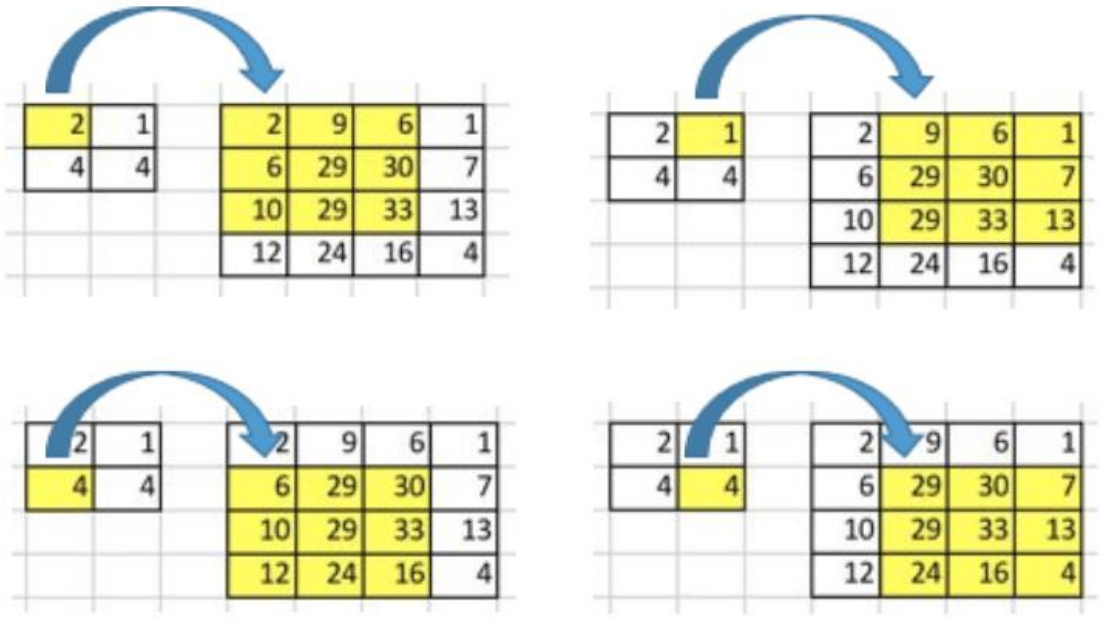


Figure 5.7 Backword of Convolutional.

There is no big difference between transposed convolutional operation and normal convolutional operation. The only difference are it forms connectivity in the opposite direction of the normal convolutional operation, that's why it is used in up-sampling operations. Other than that the weights of convolutional operations are not learnable like in transpose convolutional operations. So the transpose convolutional operation doesn't require a predefined interpolation process. It doesn't mean transposed convolution is taking a transposed version of the existing convolution matrix. The main idea is the way how the relationship between input and the output matrix is managed. It should be handled backward when compared with the normal convolutional matrix. This explains that transpose convolution is not a convolution.

5.3 U-Net Architecture

U-Net architecture was first developed by Olaf Ronnberger et al. [5]. He created it for the segmentation of biomedical images. Encoder or contraction path and decoder or symmetric expansion path make up the architecture. The encoder path captures the image background, while the decoder path allows for accurate localization using transposed convolutions. It just has convolutional layers, so we should call it a end-to-end fully convolutional network (FCN). It will tolerate any size image due to the lack of a thick layer. Figure 5.8 depicts the original architecture proposed by Olaf Ronnberger et al.

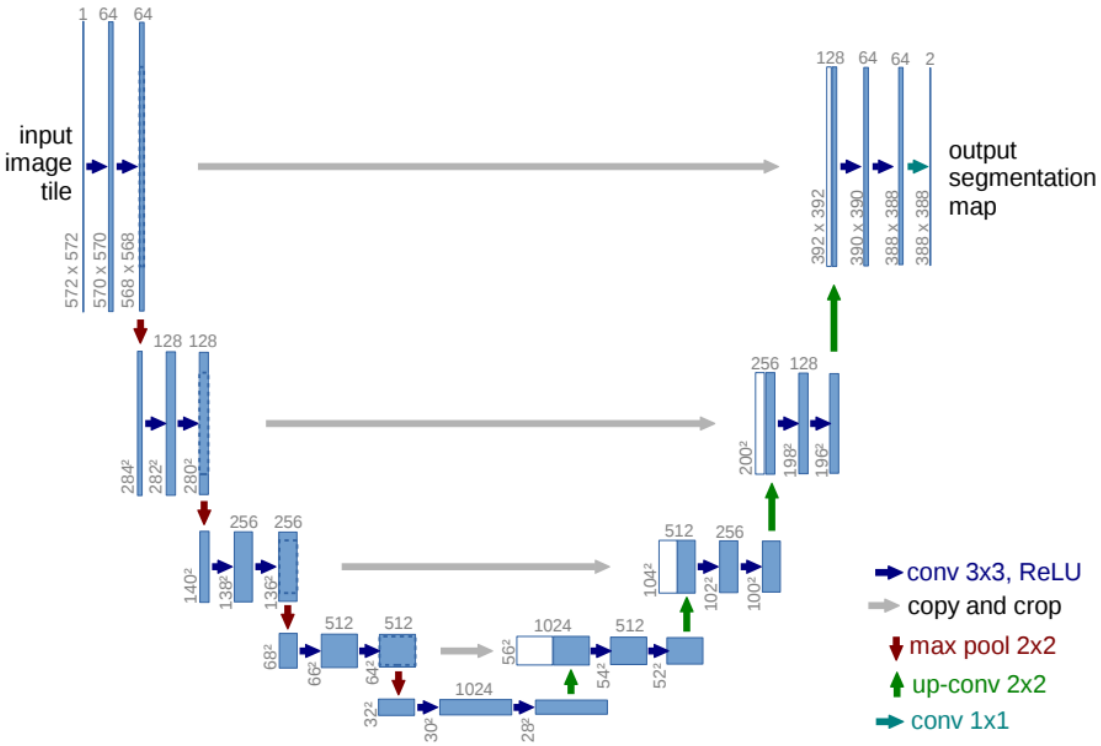


Figure 5.8 The Original U-Net Architecture.

Every blue box in the U-Net architecture represents a multi-channel function plot. On the top of the box, the number of channels is shown. The x-y-size is shown on the box's lower left lip. Copies of function maps are represented by white boxes. The various operations are shown by the arrow.

CHAPTER 6

Implementation

6.1 Dataset

We use the dataset of Massachusetts Benchmark dataset for Roads and Buildings extraction. In the image directory, there are 2300 satellite images representing rural and urban areas. In which we can find images of roads and buildings in the rural and urban areas. The training dataset contains 3 labeled images of 1300×1300 pixels. After the training, we run a prediction on 100 test images of 1300x1300 pixels. The predicted images are then cropped in patches of 256x256x3 pixels and transformed to a CSV submission file containing the predicted output (22 for a building, 11 for a road, 00 for background) for each patch.

6.2 Pre-Processing

Pre-processing is used to improving differentiation among the pictures, which includes space adjustment, shading standardization, expel picture relics. We do the geometric corrections, Radiometric corrections, Image transformation, and image enhancement. After that, we do other noise removal techniques, for better visualization of the image.

6.3 Preparing dataset

After the pre-processing step, we need to digitize our image, for digitizing process we are using QGIS. QGIS is a free and open-source cross-platform desktop geographic information system (GIS) application that supports viewing, editing, and analysis of geospatial data. As the first process of digitization, we need to vectorize and rasterize our data.

6.3.1 Loading Raster data

As the first step of our preparation we are going to rasterize our data. Rasterization of the data can be done by following process.

- Click on the *Open Datta Source Manager* button on QGIS toolbar. It looks like three cards (one red, one yellow, and one blue) fanned out.
- Click the *Raster* button (it looks like a checker board) on the left side of the *Data Source Manager* window.

- Click on the "..." button and then navigate to where we saved our data and select the image file.
- Click *Open*.
- Finally, click "Add" and we should see a black and white raster image appear in the map canvas below the dialog we are working in. Leave the *Data Source Manager* window open so we can add some more data. Next we move on to Vectorization of data.

6.3.2 Loading Vector data.

As the next step we are going to vectorize the data. Vectorization can be done by following process. Vectorization can be done by two ways in QGIS. First by shape file and next by .csv file.

6.3.2.1 Shapefiles

Shapefiles are a very popular vector data format. Let's load our shapefile data.

- In the *Data Source Manager*, click on the *Vector* tab on the left.
- In the *Source* section, click on the "..." and navigate to the folder containing your vector data.
- Holding down the Ctrl button on your keyboard while you click, select the *.shp* file you need to add. Then click *Open*.
- In the *Data Source Manager* click *Add*.

6.3.2.2 .csv files

It's pretty common to get point data in the "CSV" file, especially if the spatial data is represented by latitude and longitude coordinates. CSV stands for Comma Separated Value. Typically this is tabular data where the edge of each cell of the table is indicated by a comma. Sometimes people use a different character instead of the comma such as a semicolon, tab, or pipe. The character used to indicate the edge of the cells is called the "delimiter". If a file has tabs as the delimiter, for example, you would call that file a "tab-delimited" file.

To load our .csv file:

- In the *Data Source Manager*, click on the *Delimited Text* tab on the left. Notice that the icon is a comma.
- Next to the *File Name* text box, click the "..." button, then navigate to our *.csv* file and click *Open*.
- In the *File format* section, we'll leave the default selection of *CSV (comma separated values)*, but if we had a file with a different delimiter, we could change the delimiter by using *Custom delimiters*.
- In the *Records and fields options* section, make sure *First record has field names* is checked. If your data didn't have table headings, we would want to uncheck this box.
- In the *Geometry definition* section is where we indicate what kind of geometry we have. For ours, select *Point coordinates* and in the *X field*, pick "Longitude", and for the *Y field* pick "Latitude".
- Review the *Sample data* section to preview how the attribute table will look. This is a good way to find out if we picked the right delimiter or if our data has some formatting issues (such as someone put commas in a text field and use commas as the delimiter).
- If everything looks good, click *Add*. If the file is large, it will take some to load.
- Close the *Data Source Manager* window because we are done adding data.

6.3.3 Working with Raster Data

Let's start by looking at some Raster data. We'll work with a digital elevation model (DEM) for our image. A DEM is a raster in which each cell in the grid contains the elevation at that location. For now, let's turn off all of the layers in the Layers panel except for the DEM_SF layer by unchecking the boxes next to the layer names in the Layers Panel on the left side of our screen. Now we can see a greyscale image that roughly looks like the input image. This is a Digital Elevation Model (DEM). Each cell in the raster contains a number representing the elevation at that location.

6.3.4 Working with Vector Data

After the raster data now we concentrate on the vector data. In the Layers panel, turn off the DEM and turn on the streets layer. Notice that the streets are represented with lines. In

this process we are going to do single symbol styling, attribute tabling and select the data by attributes.

6.3.4.1 Single Symbol Styling

In single symbol styling, Street-layer is loaded by default with a randomly selected colour. Let's start our vector work by changing the styling of our streets to something more appropriate

.

- If *Layer Styling* panel isn't still open, reopen it from the *View* menu by selecting *Panels* and then checking the box next to *Layer Styling*.
- Make sure the drop-down to select the layers to work with is set to our data.
- Leave the drop-down for selecting the method of symbolizing the data on *Single symbol*.
- In the white box near the top, we can see the word *Line* and *Simple line*. Click on the words *Simple line*. This will let us access lots of options for how to symbolize this set of lines.
- In the *Colour* box, click on the coloured box to open the colour selection dialog.
- The colour selection dialog has multiple options for how you select our colours. Take a minute to get a feel for how each of these works. I find each of these has advantages for certain situations.
- Choose a colour that we think represents roads well. I used a dark grey. We can enter *#666666* into the *HTML notation* box to use the same colour if you'd like.
- Once we picked a colour, use the *Go Back* button to return to the main dialog.

6.3.4.2 Attribute Table

Before going to select the data by attribute process we should first understand the nature of vector data. Vector data is typically made up of two parts: (1) the points, lines, or polygons that represent real-world entities called the geometry and (2) information about those entities, typically in a table format, called attributes. Selecting streets and buildings by hand is helpful, but depending on what we want to do, we might want an automated way to select the streets and buildings that we want to highlight. Let's investigate the class code column..

- In the attribute table for the streets layer, click on the *Select features using an expression* to open the *Select by Expression* tool.
- We'll build an expression in the white box on the left side of the tool. In the centre panel, expand the *Fields and Values* list.
- Double click the *class code* field to add it to the expression box on the left.
- Then click the = button to add an equal sign to the equation.
- We can also click the *All Unique* button on the right to see all of the values that are found in the *class code* column. I don't recommend using this option on continuous data; it's best for categorical data with a relatively small number of unique values. Click *'1'* in the list. Note that while we think of the items in this list as numbers, this column was defined as text, so this is why the numbers are wrapped in quotes. If we tried to use the number without the quotes, the GIS would find no matches because it would be looking for the number 1, not the text 1. Your expression should look like *"class code" = '1'*.
- Click *Select features*. You'll notice that rows in the attribute table and lines on the map have been highlighted.
- In our data we assign the class code 00 for background which is other that roads and buildings, 11 for roads and 22 for buildings.
- After this process we can symbolize the layers by its attributes. Which means we can add the building name street name for each and every buildings and roads. Now our data is ready for the segmentation process using Deep learning models.

6.4 Rasterize

The rasterization process is done in two ways one is by using QGIS which is time-consuming, which is described earlier and the second method is by using python. In this, we will see how to rasterize the data using python. As a first process, we are going to create labels with shapefiles for the images. While doing that we need to take care of the projection of imagery and shapefile which should be the same. And another main constrain is the projection unit should be in meter if you want to buffer line features. It will take the input from the given file as the image and add the raster add vector data to those images and save those images in the destination file. While doing so we are adding buffer length (--buffer) for line features. The polygon features don’t need buffer length so we are ignoring that for those features. Then we are adding buffer length for the attributes (--buffer_atr) from the vector file. This attribute can

be buffer width and multiples with the buffer of line. And this is also the same as --buffer which is not needed for polygon features. After that the attributes form the vector file (labels_atr), pixels inside the polygons will be assigned by their attribute value. This is not required for the line and single class (polygon).

Example:

python rasterize.py --raster ../data/spacenet/raster/spacenet_chip0.tif --vector ../data/spacenet/vector/spacenet_chip0.shp --buffer 2 --buffer_atr lanes --output_file ../data/spacenet/binary/test.tif

Options	Description
--help	Print usage information
--raster	Raster/Image name with directory
--vector	Vector file name with directory
--output_file	Output file name with directory
--buffer	Buffer length for line feature. Not required for polygon
--buffer_atr	Attribute from the vector file, this attribute can be buffer width and multiplies with –buffer. Not required for polygon
--labels_atr	Attribute from the vector file, pixels inside the polygon will be assigned by its attribute value. Not required for line and single class (Polygon)

Table 6.1 rasterize.py explanation

6.5 Patch Generation

After the rasterization and vectorization of the data we need to generate patches for the input images. Patches are the process of generating multiple images from a single image and here we are making the patches for 256x256 pixel values. To generate the patches for train, test, and validate the command must be run three times. The main constrain is that the name of the image and the label name should be the same.. patch_gen.py is used for generating the patches for the image.

Example:

python patch_gen.py --image_folder ../data/mass_sample/test/image/ --image_format tiff --label_folder ../data/mass_sample/test/roads_and_buildings/ --label_format tif --patch_size 256 --output_folder ../data/mass_patches/

Options	Description
--image_folder	Folder of input images/tiles with directory
--image_format	Image format tiff/tif/jpg/png
--label_format	Label format tiff/tif/jpg/png
--label_folder	Folder of label images with directory
--patch_size	Patch size to feed network. Default size is 256
--overlap	Overlap between two patches on image/tile
--output_folder	Output folder to save patches

Table 6.2 patch_gen.py explanation

6.6 .csv path generation

As a next step we are going to save the directories of patches in .csv file. We are saving the patches in .csv because instead of reading patches from the folders directly it will be easy to read from the .csv file. It includes the image folder which consists of image patches with the directory, image format, label_folder where label patches are get stored, label format, patch size, and output folder where the output .csv file gets saved.

Example

python csv_paths.py --image_folder ../data/mass_patches/image/ --image_format tif --label_folder ../data/mass_patches/label/ --label_format tif --output_csv ../paths/data_rd.csv

Options	Description
--image_folder	Folder of image patches with directory
--image_format	Image format tif (patch_gen.py save patches in tif format)
--label_folder	Folder of label patches with directory
--label_format	Label format tif (patch_gen.py save patches in tif format)

--patch_sie	Patch size to feed network. Default size is 256
--output_csv	Csv filename with directory

Table 6.3 csv_path.py explanation

6.7 Training the model

After generating the patches now we are ready to train our model. It consists of training model name in our case it is U_net model, the csv file name with the directory, consist of directories of image and label patches of the training set, the csv file name with the directory, consist of directories of images and labels patches of the validation set, input shape of models to feed (patch size and patch size channels), batch size depends on GPU/CPU memory in our case we are using a batch size of 1, the number of classes ion label data, and the number of epochs. In our case, we are using the epoch size of 100..

Example

python train.py --model unet --train_csv ../paths/data_rd.csv --valid_csv ../paths/data_rd.csv --input_shape 256 256 3 --batch_size 1 --num_classes 3 --epochs 100

Option	Description
--model	Name of the model which is u-Net
--train_csv	.csv file name with directory, consists of directories of image and label patches of training set.
--valid_csv	.csv file name with directory, consists of directories of image and label patches of validation set.
--input_shape	Input shape of model to feed (patch_size, patch_size_channels)
--batch_size	Batch size, depends on GPU/CPU memory
--num_class	Number of classes in labels data
--epochs	Number of epoch

Table 6.4 train.py explanation

6.8 Predicting using Deep Learning Models

One form of deep learning model is the U-Net model, which is a completely convolutional network model. We use u-Net to train 18 layers in this experiment. Prior to training the files, pre-processing steps such as managing missed entries, inserting median values, segmenting, and resizing the images are completed. We get the accuracy as well as the failure after training. The predicting model is made up of the model's input form (patch size, patch size channels), a trained model with a directory, a folder of input images/tiles with a directory, and an output folder where the projected image/tiles are saved.

Example:

python tile_predict.py --model unet --input_shape 256 256 3 --weights ../trained_models/unet300_06_07_20.hdf5 --image_folder ../data/mass_sample/test/image/ --image_format tiff --output_folder ../data/

Options	Description
--input_shape	Input shape of the model
--weights	Trained model with directory
--image_folder	Folder of input images with directory
--image_format	Image format tiff/tif/jpg/png
--output_folder	Output folder to save predicted image

Table 6.5 tile_predict.py explanation

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	(None 256 256 3)	0	
conv2d (Conv2D)	(None 256 256 64)	1792	input_1[0][0]
conv2d_1 (Conv2D)	(None 256 256 64)	36928	conv2d[0][0]
max_pooling2d (MaxPooling2D)	(None 128 128 64)	0	conv2d_1[0][0]
conv2d_2 (Conv2D)	(None 128 128 128)	73856	max_pooling2d[0][0]
conv2d_3 (Conv2D)	(None 128 128 128)	147584	conv2d_2[0][0]
max_pooling2d_1 (MaxPooling2D)	(None 64 64 128)	0	conv2d_3[0][0]
conv2d_4 (Conv2D)	(None 64 64 256)	295168	max_pooling2d_1[0][0]
conv2d_5 (Conv2D)	(None 64 64 256)	590080	conv2d_4[0][0]
max_pooling2d_2 (MaxPooling2D)	(None 32 32 256)	0	conv2d_5[0][0]
conv2d_6 (Conv2D)	(None 32 32 512)	1180160	max_pooling2d_2[0][0]
conv2d_7 (Conv2D)	(None 32 32 512)	2359808	conv2d_6[0][0]
max_pooling2d_3 (MaxPooling2D)	(None 16 16 512)	0	conv2d_7[0][0]
conv2d_8 (Conv2D)	(None 16 16 1024)	4719616	max_pooling2d_3[0][0]
conv2d_9 (Conv2D)	(None 16 16 1024)	9438208	conv2d_8[0][0]
up_sampling2d (UpSampling2D)	(None 32 32 1024)	0	conv2d_9[0][0]
concatenate (Concatenate)	(None 32 32 1536)	0	up_sampling2d[0][0] conv2d_7[0][0]

conv2d_10 (Conv2D)	(None 32 32 512)	7078400	concatenate[0][0]
conv2d_11 (Conv2D)	(None 32 32 512)	2359808	conv2d_10[0][0]
up_sampling2d_1 (UpSampling2D)	(None 64 64 512)	0	conv2d_11[0][0]
concatenate_1 (Concatenate)	(None 64 64 768)	0	up_sampling2d_1[0][0]conv 2d_5[0][0]
conv2d_12 (Conv2D)	(None 64 64 256)	1769728	concatenate_1[0][0]
conv2d_13 (Conv2D)	(None 64 64 256)	590080	conv2d_12[0][0]
up_sampling2d_2 (UpSampling2D)	(None 128 128 256)	0	conv2d_13[0][0]
concatenate_2 (Concatenate)	(None 128 128 384)	0	up_sampling2d_2[0][0] conv2d_3[0][0]
conv2d_14 (Conv2D)	(None 128 128 128)	4424960	concatenate_2[0][0]
conv2d_15 (Conv2D)	(None 128 128 128)	1475840	conv2d_14[0][0]
up_sampling2d_3 (UpSampling2D)	(None 256 256 128)	0	conv2d_15[0][0]
concatenate_3 (Concatenate) (None 256 256 192)	0	up_sampling2d_3[0] [0] conv2d_1[0][0]	
conv2d_16 (Conv2D)	(None 256 256 64)	1106560	concatenate_3[0][0]
conv2d_17 (Conv2D)	(None 256 256 64)	36928	conv2d_16[0][0]
conv2d_18 (Conv2D)	(None 256 256 3)	195	conv2d_17[0][0]

Table 6.6 u-Net Architecture layer values

The accuracy of the model is 0.98, the total number of params are 31,379,075,000, the total number of trainable params are 31,379,075,000 and the total number of non-trainable params are 0. The output of the convolutional layer is provided by the input layer after convolution, with the image's height and breadth reduced. With respect to the input layer, the output size of the convolutional layer is reduced to two. The input is fed into the max-pooling layer after the convolutional layer. This layer reduces the image's dimension by downsampling the input. Finally, we have the encoding layer, which reconstructs the signal. Figure 7 depicts the input and output images. With the help of a segmented image, one can easily find the buildings which are illegally constructed along the roadside manually..

Figure 6.1 Input and output of road and building segmentation using u-Net

6.9 Accuracy

After the prediction process, we are calculating the accuracy of the model. We are calculating the accuracy of the model by using different accuracy metrics which are IoU score, F1-Measure, Precision, Recall, Mean IoU Score, and Mean F1-Measure with the confusion matrix. It is done by taking the input shape of the model, weight which is trained model with the directory, .csvfile name with the directory, consists of the directories of image and label patches of the test set. The overall accuracy we are getting is 0.98 which is very good in our case. Figure 6.2 shows the overall accuracy and the accuracy gained for each classes.

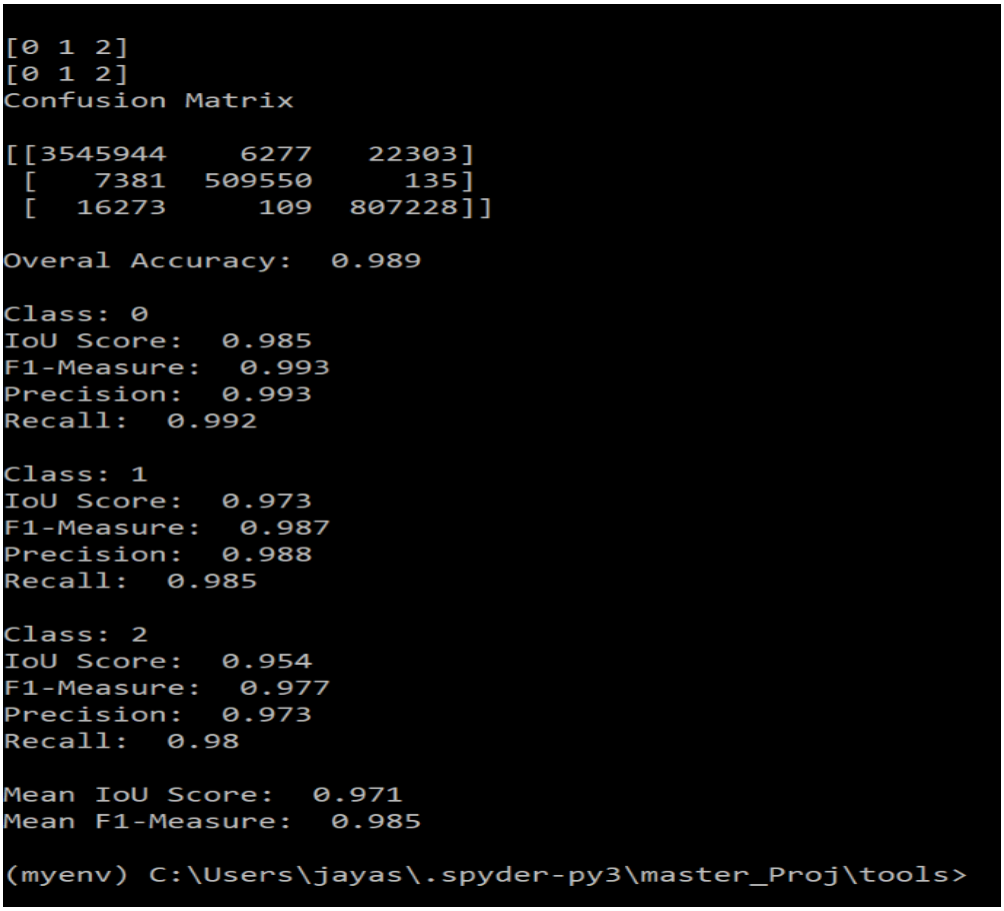


Figure 6.2: Accuracy of the model.

Example

python accuracy.py --model unet --input_shape 256 256 3 --weights ../trained_models/unet300_06_07_20.hdf5 --csv_paths ../paths/data_rd.csv --num_classes 3

Options	Description
--input_shape	Input shape of model
--weights	Trained model with director
--csv_paths	.csv file name with directory, consists of directories of image and label patches of test set.
--num_classes	Number of classes in labels data

Table 6.7 accuracy.py explanation

6.10 Summary of the model.

As the final step of segmentation, we are getting the summary of the model by using summary.py. It consists of the name of the model, input shape of the model to feed, number of classes to train. In our case, the number of classes is 3 which are buildings, roads, and background images. As a result, we will get the layers of the u-Net architecture and the total number of trainable parameters, and the total number of trained parameters.

Example

python summary.py –model unet –input_shape 256 256 3 –num_classes 3

Options	Description
--model	Name of the model which is u-Net
--input_shape	Input shape of the model to feed (patch_size, patch_size_channels)
--num_classes	Number of classes to train.

Table 6.8 summary.py explanation

6.11 Comparison of the image

As the next step, we need to compare the timed images for checking if there any illegally constructed buildings and roads are present in that particular area. For the comparison of the images, we are taking the image from the repository which is checked by the government officials manually and say there are no illegally constructed buildings or roads in that area, the timed image of the same area. And then we are converting both the images into NumPy arrays. After that we are converting all the 0's into 3's in both the arrays. We are doing that because the multiplicative inverse of zero will be zero itself, in that case, our total model will fail that's why we are converting the value 0's to 3's. There will be no change in the result by the 0's conversion. Now 1 will represent the roads, 2 will represent the buildings and 3 will represents the backgrounds (other than roads and buildings). After that, we are multiplying the first image array value (q) which is our checked image with '10', and add the result with the checking image array (w). The following equation explains the above-mentioned operation.

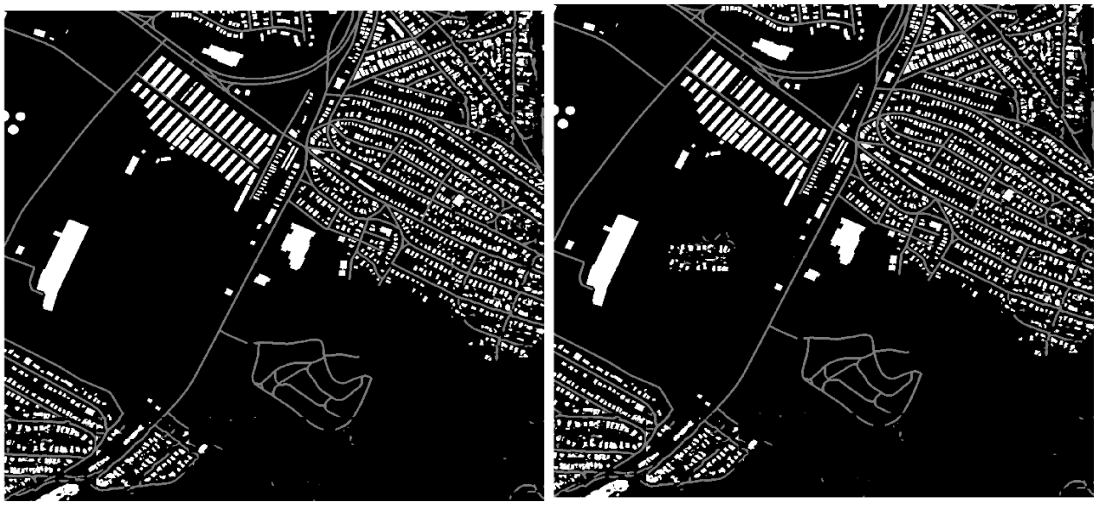
$$\mathbf{Rt=(q*10)+w}$$

Where, Rt = resultant array, q = checked image array value, w = comparing image array value. The result will be in the form of 11, 12, 13, 21, 22, 23, 31, 32, and 33. The following table gives a brief explanation of the output representation.

Value	Description
11	No changes in the road.
12	New buildings are constructed in the place of roads.
13	Roads had been demolished or removed.
21	New roads are constructed in the place of building.
22	No changes in the buildings.
23	Buildings had been demolished or removed.
31	New road is constructed in the empty area.
32	New buildings are constructed in the empty area.
33	No change in the empty area.

Table 6.9 Output array value description

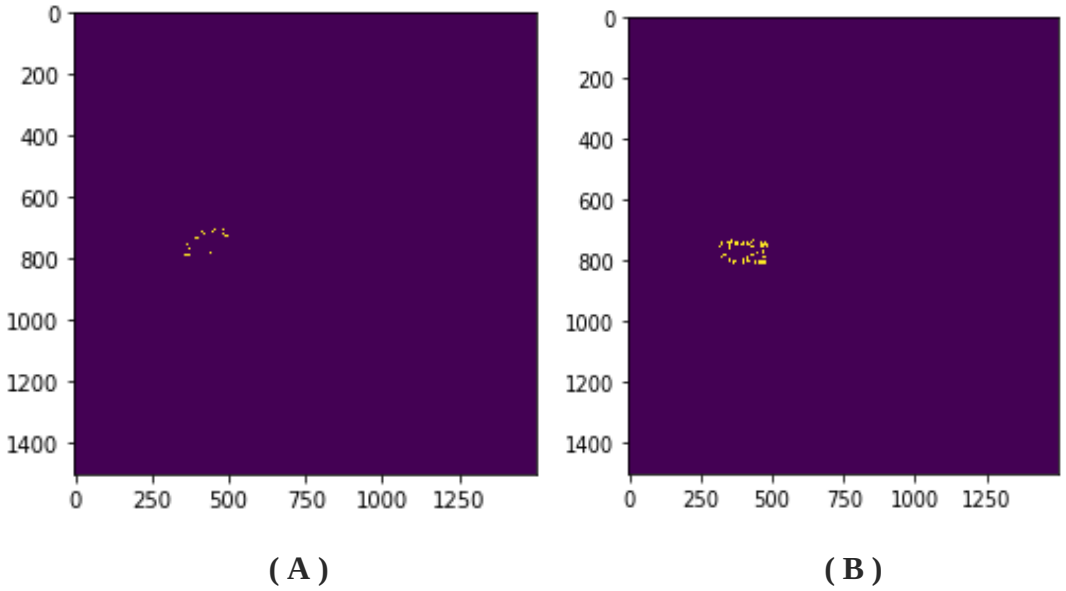
As the next step, we are extracting only the newly constructed buildings and roads if there are any. We are doing that by changing the value of the output array. We are converting the pixel value which denotes there is a change to 1 and the rest of the pixel value to 0. For example, consider there are new buildings have constructed in the empty area. In that case the pixel value for the place where a newly constructed building will be 32. Now we are converting that 32 to 1 and the rest of the pixel value to 0. While doing so we will get only the newly constructed building image. Now we will open that image in the QGIS to know the coordinates. With the help of the coordinates, we can easily find the location of the buildings where it is constructed and the government officials can easily identify whether that building is constructed legally or illegally. The following image explains the extraction of the particular area from the image.



(A) **(B)**

Figure 6.3 (A) Image in the government repository, (B) Image we obtain now

Here figure A represents the image in the government repository which is already checked by the government officials and states each and every roads and buildings in the image are constructed legally. Now figure B represents the newly obtained image of the same area. Now we need to check the presence of any newly constructed buildings and roads are present in that area. The following image shows the comparison results.



(A) **(B)**

Figure 6.4 (A) construction of extra roads in that particular area (b) construction of extra building in that particular area.

The above figure represents the output of the comparison of the two images. In the comparison process, we got to know that there are newly constructed roads and buildings in the comparison image. (A) represents that there are newly constructed roads in that particular area and (b) represents that there are newly constructed buildings in that particular area. After we obtain those images we will upload those images in QGIS to get the coordinates of the newly constructed buildings and roads. Once we get the coordinates it will be easy to find whether those roads and buildings are constructed in a legal way or not.

CHAPTER 5

CONCLUSION AND FUTURE WORK

In this paper, we are detecting illegally constructed roads and buildings using satellite images. As future work, we planned to make the whole process automated. Since here for the matching of the segmented image, we are manually uploading the image to QGIS for getting the coordinates of the buildings and roads. So instead of doing that we intend to do that automatically as our future work. In addition to that, we planned to merge two or more residual models for the prediction process to improve the processing time of the system, in this thesis, we have projected the U-Net model for streets and building extraction from high-resolution remote sensing data.

To sum up, the projected approach combines the advantages of residual learning and U-Net. The operation's good paradigm results in an average precision of 0.98. By analysing the city database, we have developed a model that will assist in locating illegally built buildings in both urban and rural areas. Authorities can manually check unmatched buildings and roads which we get as the output against the city database to detect illegal structures and roads. Apart from that, we've seen few application of U-Net implementation in this article, which gives us a general understanding of the U-Net architecture. The artificial intelligence for detecting patterns is enhanced by combining machine learning and deep learning techniques. As a result, we can infer that CNN and other pre-training classifiers are beneficial not only to the biomedical industry, but also to the remote sensing industry.

REFERENCES

1. Times of India https://timesofindia.indiatimes.com/india/over-1-51-lakh-died-in-road-accidents-last-year-up-tops-among-states/articleshow/72078508.cms

2. Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "ImageNet classification with deep convolutional neural networks." *Communications of the ACM* 60.6 (2017): 84-90.

3. Mnih, Volodymyr, and Geoffrey E. Hinton. "Learning to detect roads in high-resolution aerial images." *European Conference on Computer Vision*. Springer, Berlin, Heidelberg, 2010.

4. He, Kaiming, et al. "Deep residual learning for image recognition." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016.

5. Ronneberger, Olaf, Philipp Fischer, and Thomas Brox. "U-net: Convolutional networks for biomedical image segmentation." *International Conference on Medical image computing and computer-assisted intervention*. Springer, Cham, 2015.

6. Rawat, Waseem, and Zenghui Wang. "Deep convolutional neural networks for image classification: A comprehensive review." *Neural computation* 29.9 (2017): 2352-2449.

7. STAROVOITOV, Valery, and Aliaksei MAKARAU. "Multispectral Image Pre-Processing for Interactive Satellite Image Classification."

8. Kumar, Gaurav, and Pradeep Kumar Bhatia. "A detailed review of feature extraction in image processing systems." *2014 Fourth international conference on advanced computing & communication technologies*. IEEE, 2014.

9. Leng, Chengcai, et al. "Local feature descriptor for image matching: A survey." *IEEE Access* 7 (2018): 6424-6434.

10. Abraham, Lizy, and M. Sasikumar. "A fuzzy based road network extraction from degraded satellite images." *2013 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*. IEEE, 2013.

11. Anil, P. N., and S. Natarajan. "A novel approach using active contour model for semi-automatic road extraction from high resolution satellite imagery." *2010 Second International Conference on Machine Learning and Computing*. IEEE, 2010.

12. Barzohar, Meir, and David B. Cooper. "Automatic finding of main roads in aerial images by using geometric-stochastic models and estimation." *IEEE Transactions on Pattern Analysis and Machine Intelligence* 18.7 (1996): 707-721.

13. Baumgartner, Albert, et al. "Automatic road extraction based on multi-scale, grouping, and context." *Photogrammetric Engineering and Remote Sensing* 65 (1999): 777-786.

14. Unsalan, Cem, and Beril Sirmacek. "Road network detection using probabilistic and graph theoretical methods." *IEEE Transactions on Geoscience and Remote Sensing* 50.11 (2012): 4441-4453.

15. Lan, Zeying, and Yang Liu. "Study on multi-scale window determination for GLCM texture description in high-resolution remote sensing image geo-analysis supported by GIS and domain knowledge." *ISPRS International Journal of Geo-Information* 7.5 (2018): 175.

16. Das, Sukhendu, T. T. Mirnalinee, and Koshy Varghese. "Use of salient features for the design of a multistage framework to extract roads from high-resolution multispectral satellite images." *IEEE transactions on Geoscience and Remote sensing* 49.10 (2011): 3906-3931.

17. Fua, Pascal, and Yvan G. Leclerc. "Model driven edge detection." *Machine Vision and Applications* 3.1 (1990): 45-56.

18. George, Jobin, Leena Mary, and K. S. Riyas. "Vehicle detection and classification from acoustic signal using ANN and KNN." *2013 international conference on control communication and computing (ICCC)*. IEEE, 2013.

19. Gruen, Armin, and Haihong Li. "Semi-automatic linear feature extraction by dynamic programming and LSB-snakes." *Photogrammetric engineering and remote sensing* 63.8 (1997): 985-994.

20. Heermann, Philip Dale, and Nahid Khazenie. "Classification of multispectral remote sensing data using a back-propagation neural network." *IEEE Transactions on geoscience and remote sensing* 30.1 (1992): 81-88.

21. Herumurti, Darlis, et al. "Urban road extraction based on hough transform and region growing." *The 19th Korea-Japan Joint Workshop on Frontiers of Computer Vision*. IEEE, 2013.

22. Wang, Weixing, et al. "A review of road extraction from remote sensing images." *Journal of traffic and transportation engineering (english edition)* 3.3 (2016): 271-282.

23. Hormese, Jose, and C. Saravanan. "Automated road extraction from high resolution satellite images." *Procedia Technology* 24 (2016): 1460-1467.

24. Barrile, Vincenzo, Antonino Nunnari, and Rosa C. Ponterio. "Laser Scanner for the Architectural and Cultural Heritage and Applications for the Dissemination of the 3D Model." *Procedia-Social and Behavioral Sciences* 223 (2016): 555-560.

25. Chen, Li, et al. "Road extraction from VHR remote-sensing imagery via object segmentation constrained by Gabor features." *ISPRS International Journal of Geo-Information* 7.9 (2018): 362.

26. Zhou, Tingting, Chenglin Sun, and Haoyang Fu. "Road information extraction from high-resolution remote sensing images based on road reconstruction." *Remote Sensing* 11.1 (2019): 79.

27. He, Songtao, et al. "Sat2Graph: Road Graph Extraction through Graph-Tensor Encoding." *arXiv preprint arXiv:2007.09547* (2020).

28. Tan, Yong-Qiang, et al. "Vecroad: Point-based iterative graph exploration for road graphs extraction." *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020.

29. Unsalan, Cem, and Beril Sirmacek. "Road network detection using probabilistic and graph theoretical methods." *IEEE Transactions on Geoscience and Remote Sensing* 50.11 (2012): 4441-4453.

30. Yamashita, R., et al. "Convolutional neural networks: an overview and application in radiology. Insights Imaging 9 (4), 611–629 (2018)."

31. Vakalopoulou, Maria, et al. "Building detection in very high resolution multispectral data with deep learning features." *2015 IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*. IEEE, 2015.

32. Benarchid, Omar, and Naoufal Raissouni. "Support vector machines for object based building extraction in suburban area using very high resolution satellite images, a case study: Tetuan, Morocco." *IAES International Journal of Artificial Intelligence* 2.1 (2013).

33. Bischke, Benjamin, et al. "Multi-task learning for segmentation of building footprints with deep neural networks." *2019 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2019.

34. Long, Jonathan, Evan Shelhamer, and Trevor Darrell. "Fully convolutional networks for semantic segmentation." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015.

35. Badrinarayanan, Vijay, Alex Kendall, and Roberto Cipolla. "Segnet: A deep convolutional encoder-decoder architecture for image segmentation." *IEEE transactions on pattern analysis and machine intelligence* 39.12 (2017): 2481-2495.

36. https://www.kaggle.com/c/dstl-satellite-imagery-feature-detection

37. Ronneberger, Olaf, Philipp Fischer, and Thomas Brox. "U-net: Convolutional networks for biomedical image segmentation." *International Conference on Medical image computing and computer-assisted intervention*. Springer, Cham, 2015.

38. Patravali, Jay, Shubham Jain, and Sasank Chilamkurthy. "2D-3D fully convolutional neural networks for cardiac MR segmentation." *International Workshop on Statistical Atlases and Computational Models of the Heart*. Springer, Cham, 2017.

39. Stoian, Andrei, et al. "Land cover maps production with high resolution satellite image time series and convolutional neural networks: Adaptations and limits for operational systems." *Remote Sensing* 11.17 (2019): 1986.

40. Brostow, Gabriel J., et al. "Segmentation and recognition using structure from motion point clouds." *European conference on computer vision*. Springer, Berlin, Heidelberg, 2008.

41. Shotton, Jamie, Matthew Johnson, and Roberto Cipolla. "Semantic texton forests for image categorization and segmentation." *2008 IEEE conference on computer vision and pattern recognition*. IEEE, 2008.

42. Shotton, Jamie, et al. "Textonboost: Joint appearance, shape and context modeling for multi-class object recognition and segmentation." *European conference on computer vision*. Springer, Berlin, Heidelberg, 2006.