PROJECT REPORT

ON

DETECTION OF SOLAR FARMS AND SAND-DUNES BY DEEP LEARNING AND SEMANTIC SEGMENTATION OF MEDIUM RESOLUTION SATELLITE IMAGES USING GOOGLE EARTH ENGINE

BY

Name(s) of the Students(s)	ID.No.(s)
PRATEEK GARG	2018A3PS0412P
OMATHARV VAIDYA	2018B4A70354G
ABICHAL GHOSH	2018A7PS0172G
PARAG RATHI	2018A7PS0231G
ROHAN SACHAN	2018B3A70992H
SIDDHARTHA GOSWAMI	2018A3PS0523H

AT
Regional Remote Sensing Centre, Jodhpur

A Practice School-I Station of

BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE, PILANI

JUNE, 2020

A REPORT

ON

DETECTION OF SOLAR FARMS AND SAND-DUNES BY DEEP LEARNING AND SEMANTIC SEGMENTATION OF MEDIUM RESOLUTION SATELLITE IMAGES USING GOOGLE EARTH ENGINE

BY

Name(s) of the Student(s)	ID.No.(s)	Discipline(s)
PRATEEK GARG	2018A3PS0412P	B.E. Electrical and Electronics Engineering
OMATHARV VAIDYA	2018B4A70354G	MSc. Mathematics & B.E. Computer Science
ABICHAL GHOSH	2018A7PS0172G	B.E. Computer Science
PARAG RATHI	2018A7PS0231G	B.E. Computer Science
ROHAN SACHAN	2018B3A70992H	MSc. Economics & B.E. Computer Science
SIDDHARTHA GOSWAMI	2018A3PS0523H	B.E. Electrical and Electronics Engineering

Prepared in partial fulfillment
of the Practice School-I
Course Nos. BITS C221/BITS C231/BITS C241

AT
Regional Remote Sensing Centre, Jodhpur

BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE, PILAN

JUNE, 2020

ACKNOWLEDGMENT

We would like to thank BITS Pilani, Practice School Division for giving us this excellent opportunity to do our Practice School-I internship program at Regional Remote Sensing Centre, Jodhpur.

We would also like to express our gratitude to all the scientists at Regional Remote Sensing Centre, Jodhpur, whose efforts have made it possible for us to get vital exposure and professional experience.

We would like to express our genuine appreciation towards **Dr. Srinivasa Rao Sitiraju**, General Manager of Regional Remote Sensing Centre, Jodhpur, for allowing us to be part of an elite organization and for providing us such a great learning opportunity.

Further, we would like to thank **Dr. Gaurav Kumar**, our mentor, for his guidance and for giving us this fantastic opportunity to work under him. Gaurav sir provided us with tremendously valuable resources including scientific literature in order to help us gain deep insight into this project. He has always been keen on helping us by solving our doubts through continuous interactions. We are grateful to him for the same.

We would also like to thank our faculty mentor, **Dr. H Vishwanathan** for his continuous motivation and support. He has been guiding us constantly so that we maintain discipline in our work and make the best out of this opportunity. We are thankful to him for his dedication towards students currently in the PS, arranging several seminars for our benefit, and his quick response to all our queries.

Lastly, we would like to thank our parents, friends, and relatives for their immense support throughout the project.

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE PILANI

(RAJASTHAN) Practice School Division

Station: Regional Remote Sensing Centre **Centre:** Jodhpur

Duration: 18/05/20 TO 27/06/20 (6 WEEKS)
Date of Start: 18 MAY 2020 **Date of Submission:** 25/06/2020

Title of the Project: Detection of Solar farms and Sand dunes by deep learning and semantic segmentation of medium resolution satellite images using Google Earth Engine.

STUDENT DETAILS:

PRATEEK GARG	2018A3PS0412P	B.E. Electrical and Electronics Engineering
OMATHARV VAIDYA	2018B4A70354G	M.Sc. Mathematics & B.E. Computer Science
ABICHAL GHOSH	2018A7PS0172G	B.E. Computer Science
PARAG RATHI	2018A7PS0231G	B.E. Computer Science
ROHAN SACHAN	2018B3A70992H	MSc. Economics & B.E. Computer Science
SIDDHARTHA GOSWAMI	2018A3PS0523H	B.E. Electrical and Electronics Engineering

EXPERT DETAILS

Name: Dr. GAURAV KUMAR
Designation: Scientist and Engineer
Name of PS Faculty: DR. VISHWANATHAN HARIHARAN

Key Words: Machine Learning, Deep Learning, Image Processing, Computer Vision
Project Area: Deep Learning and Semantic Segmentation

Abstract: In this project, we will be using semantic segmentation, a deep learning-based image processing technique. Semantic segmentation involves pixel-wise classification of a given image into different class labels. It has prominent applications in fields of remote sensing, medical imaging, meteorology, astronomy, etc. In particular, we will be using semantic segmentation for the detection of Solar Farms and Sand Dunes on Landsat-8 images, available on Google Earth Engine. The efficient detection of solar farms will aid in better monitoring, surveillance, and proper analysis of renewable energy resources. Our work involves high usage of geospatial rasterized images and geo-processing tools. We intend to evaluate the performance of our architecture using accuracy metrics and visual results.

Signature(s) of Student(s) *Signature of PS Faculty*

Date: 25/06/20 Date:

TABLE OF CONTENTS

Introduction to Organisation

Indian Space Research Organisation (**ISRO**) is a space agency established by the Department of Space. Its vision is to harness space technology for national development while pursuing space science research & planetary exploration. Its primary objective is to use this technology for application to various national tasks.

ISRO has four vertices over which it works -

1) Space Transportation - Deals with missiles and launch vehicles like PSLV, GSLV, Modular LV, and Reusable LV
2) Space Infrastructure - Deals with Observations, Navigation, Communication, and Space Science and Planetary missions
3) Space Applications - Deals with research laboratories involved in space applications like SAC, NRSC, RRSC
4) Space Capacity building - Involved in institutions which promote space education like IIST and IIRS, International collaborations and outreach, and Human Resource development

Figure 1 offers a detailed structure of ISRO.

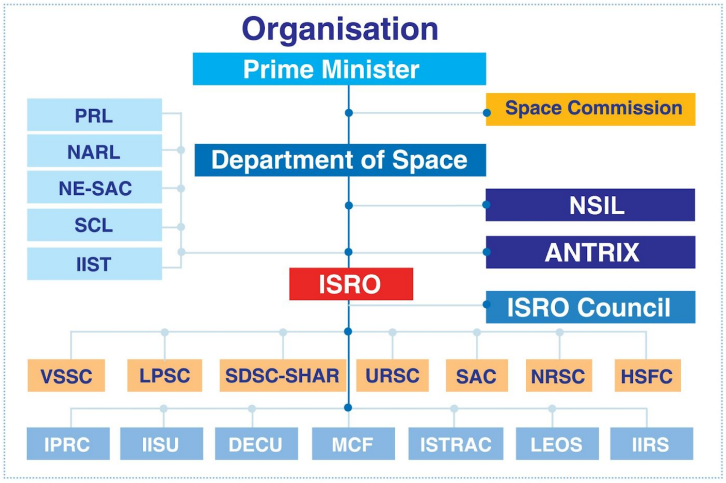


Figure 1: ISRO Organization structure

Regional Remote Sensing Centre West (RRSC-West) Jodhpur, is an ISRO research centre, which became operational in January 1988. It is located in the western part of the country and serves the needs of the western

states of India. This setup was a part of regional centres created under NNRMS (National Natural Resources Management System).

RRSC Jodhpur is a part of 5 Regional remote sensing centres in distinct parts of the country - North, West, Central, East, and South. They work on remote sensing tasks as well as research in natural resources, forestry, oceanography, environment, geology, water resources, and urban planning. These centres are headed by the National Remote Sensing Centre (NRSC), Hyderabad. It is involved in the establishment of ground stations for receiving satellite data, generation of data products, dissemination to the users, development of techniques for remote sensing applications including disaster management support, capacity building for professionals, faculty and students, and geospatial services for good governance.

RRSC Jodhpur tackles issues like Drought conditions, Desertification of land, and Natural Disasters. It is working on Remote Sensing research and GIS projects in areas such as the mapping of groundwater potential zones, mitigation of flood, irrigation facilities, detection and monitoring of dust storms, public health, reservoir capacity estimation, mapping of land use, the salinity of water bodies, water-logged areas and wastelands, etc. It is an important figure in National level mission projects like **NISAR**. It has done a number of projects for the benefit of the society with NGOs, like **Jal-Shakti Abhiyan** - Aimed at the conservation of water and promotion of irrigation efficiency.

Some of the major projects it is involved in include: **EPRIS** - Empowering Panchayati Raj Institutions Spatially, *Bhuvan Panchayat Portal* - Information support for planning on the panchayat level, **India WRIS** - Water Resources Information System, and **NHRR** - National Health Resources Repository. Some of the products and services it provides involve: **Bhuvan** - Web-based Thematic map utility for users based on ISRO maps, **Mosdac** - Repository of Weather and Ocean data, **Bhuvan-Panchayats Activity Planning Mobile Application** - Provides details on spatial planning and monitoring of the activities in 17 flagship schemes out of 66 centrally sponsored schemes.

Two of the major products and services ISRO has worked on are - WRIS (Water Resources Information System) and NHRR (National Health Resources Repository).

India-WRIS:

RRSC Jodhpur has worked considerably in the past on the issues of management of water resources. RRSC worked on a project titled ‘Generation of Database and Implementation of Web Enabled Water Resources Information System’. It was jointly executed in collaboration with CWC, New Delhi in 2009.

Figure 2: India WRIS logo

Five different versions of WebGIS software were launched in the public domain. It's portal has been designed and developed effectively so that all types of users, from all sections of society, could use it based on existing guidelines by CWC and requirement of regular updates to access near real time data accessibility, data security domains and scale of information. This portal also provides an additional Automatic Map and Report Generation facility. It is estimated that about 20,000 people visit the portal per month.

National Health Resource Repository (NHRR) Project:

The NHRR Project was launched by the Union Ministry of Health and Family Welfare. In fact, when launched, it became India's first ever healthcare establishment census which collects data from both public and private healthcare establishments. This resource repository also enables research towards ongoing and upcoming healthcare challenges coming from other determinants of health such as disease and the environment. The census is conducted under the Collection of Statistics Act 2008.

Figure 3: NHRR logo

It collects information from private and public healthcare establishments around the country such as Indian Railways, Defence, Petroleum, etc. It takes data on over 1400 variables of over 20 lakh healthcare establishments like hospitals, doctors, clinics, pharmacies, diagnostic labs, nursing homes, etc. About 4,000 trained professionals are working with dedication to approach every healthcare establishment to collect important information. Indian Space Research Organization (ISRO) is the technology partner for this project mainly for data security and analysis.

Objective

There are three parts to this project, the first part being the detection of buildings in the Tanzania dataset ,the second part includes detection of solar farms, and the third part being the detection of sand dunes using semantic segmentation.

The first part was done as a reference project. The objective in this part was to detect the pixels in the image which represented buildings using image segmentation technique. This project was done so that we could gain some prior knowledge on working with geospatial data and its preprocessing. We learned more about the UNET architecture as it was used in this part for image segmentation.This project proved to be a very good starting point for the other two parts.

In the second part of this project we will be using semantic segmentation for the detection of Solar Farms on Landsat-8 images, available on Google Earth Engine .The efficient detection of solar farms will aid in better monitoring, surveillance, and proper analysis of renewable energy resources. Our work involves high usage of geospatial rasterized images and geo-processing tools. Mapping the location of solar farms and tracking it’s installation progress is particularly important for the following aspects:

1. It allows the government to gauge the development of the solar power industry and make different strategies for development purposes.
2. It helps the solar power company to quantify and optimize the efficiency of solar panels.
3. It is useful for investors to evaluate the operation of solar power companies. Obviously, it is impractical to locate solar farms on maps manually.

The third part of the project involves applying image segmentation techniques for the detection of sand dunes. The effective detection of sand dunes in a particular area gives a rough idea about the drought conditions, overgrazing, unsustainable agricultural practices etc. If the organisation has proper data then use of soil conditioners and windbreaks can be proposed to mitigate the impacts of sand dune encroachment.

The final aim of our project is to be able to implement a network model that will allow us to perform semantic segmentation on satellite images in order to detect solar farms and sand dunes from the images so that they can be precisely tracked and monitored, and serve as a key indicator in managing disasters as well as promoting sustainable growth.

Objective Overview:

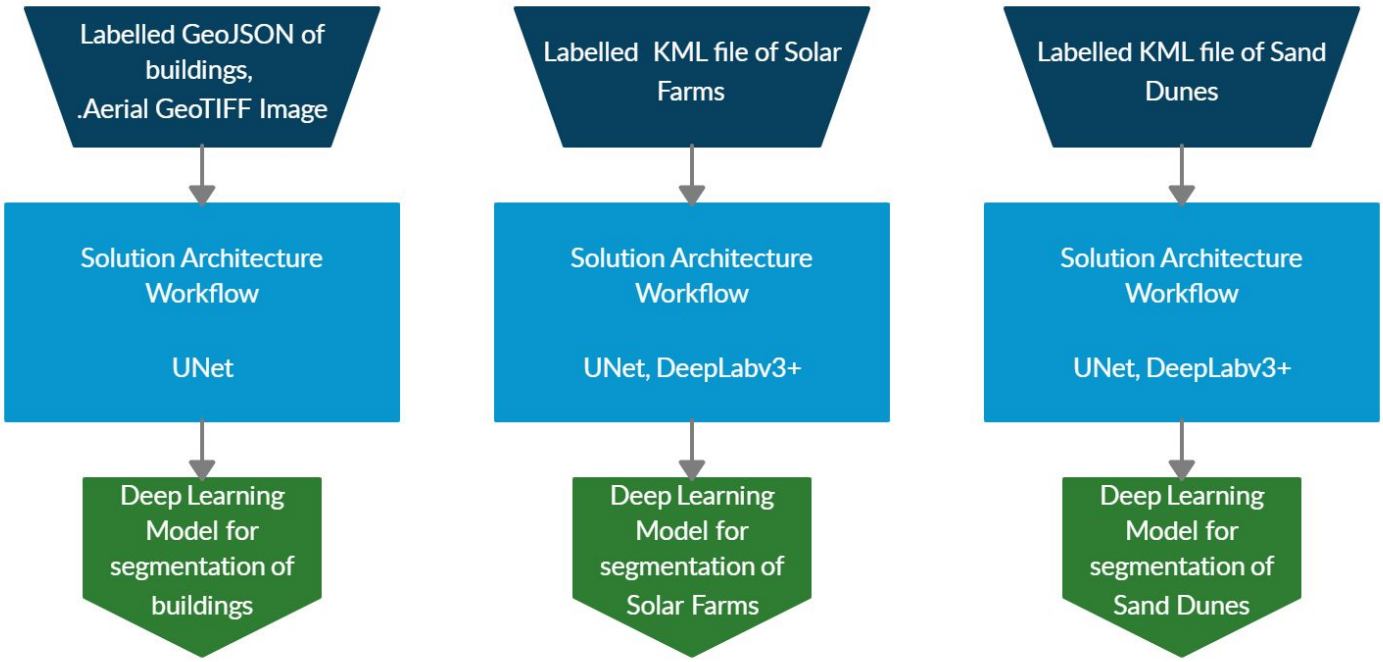


Figure 4: Overview of the project objectives

Project Requirements

- **Softwares Required:**

1. Google Colab:

   Colaboratory, also called “Colab” in short, is an open-source product which is made available by Google Research. Collaboratory is a Jupyter notebook type environment which is provided by Google for free. It provides an executable document which allows us to write, run, and share code in python. The main advantages of using Colab are as follows-

   a. Colab notebooks allow you to combine **executable code** and **rich text** in a single document, along with **images**, **HTML**, **LaTeX** and more.
   b. The Colab notebooks created by us get automatically stored in our Google Drive account.
   c. Colab notebooks can be easily shared with co-workers or friends, allowing them to comment on your notebooks or even edit them.

d. Access to GPU which allows us to run our code faster. Colab provides us with 12gb Ram and disk spaces.

2. Google Earth Engine:

   Google Earth Engine is a cloud-based platform that is used for planetary-scale environmental data analysis. It is a platform for scientific analysis and visualization of geospatial datasets, for academic, non-profit, business and government users. Earth Engine also provides APIs and other tools to enable the analysis of large datasets.

   (The detailed use of GEE is provided in later part of the report with the code snippet)

● Programming Languages used:

1. Python 3.6 : Python is one of the best languages for data analysis and for research and development in Machine Learning/Deep Learning. Python provides us with an extensive database of libraries which are useful for artificial intelligence and machine learning. This abundance of massive libraries and frameworks makes coding easy and also helps in saving development time. The whole code of our project is written in python and uses many different python libraries.

2. JavaScript: It is a client scripting language which is used for web development,mobile applications etc.In our project javascript was used for converting the farm labels present in the .shp file to a geotiff image using Google Earth Engine.

● Python Libraries Required:

1. Geopandas: This is an open-source project which makes working with high-resolution geospatial data in python much easier. Basically the data types which are used by pandas to allow spatial operations on geometric types are extended by geopandas such as polygons etc.It allows one to easily read, write & manipulate vector data (like .geojson) and returns a DataFrame object for the same.

2. Rasterio: Rasterio is a special Python-based library that is used to read and write the rasterized imagery such as GeoTIFF images. Rasterization is the process of representing images as a collection of smaller

segments (like pixels).Since satellite images are of very high resolution in nature, rasterizing them promotes efficient management of smaller image segments.

3. Supermercados: This library helps to divide a geometry into uniformly shaped square Mercator tiles. These tiles carry additional information along with them, i.e. X, Y & Z. X and Y represent the tile coordinates whereas Z represents the zoom level at which tiling was done.

4. Rio-tiler:It is a special Rasterio plugin, which helps to break the GeoTIFF raw image and Geojson geometry label into the tiling format as generated by supermercado. By using this library, we break a high-resolution TIFF image and JSON labels to a normal dataset that can be used for training a deep learning model. Each image tile generated by this is a normal image in PNG format.

5. Solaris: This library helps in converting the image labels into a special 3 channel RGB format.
   1)The 1st channel represents the building footprints
   2)The 2nd channel represents building boundaries
   3)The 3rd channel represents the contact areas between the adjacent geometries
   By converting the labels into this format, the deep learning model will be better able to appreciate the boundaries, footprints, and closeness of the objects, since we have embedded this information into the labels in the form of channels.

- **Tools used for converting files:**

1. kml to geojson - The data of the solar farm that was provided was present in .kml format. So for the processing part we had to convert it to the geojson file format. (https://github.com/mapbox/togeojson).

2. GeoJSON into Shapefile - For plotting the solar farms in Google Earth Engine we need to Import the data in shp. file format.(https://github.com/jdesboeufs/geojson2shp).

Expected Outcome:

After the completion of our segmentation and training part:

1. We should be able to correctly identify the solar farms in the given geospatial image which was not present in the dataset before.

2. Labeling more solar panels from the satellite imagery data in various circumstances, such as the solar panels on the roof in residential areas.
3. This same model can then be extended to various other geospatial images for training purposes by just changing the loss function.

Overview Workflow of the Project Code:

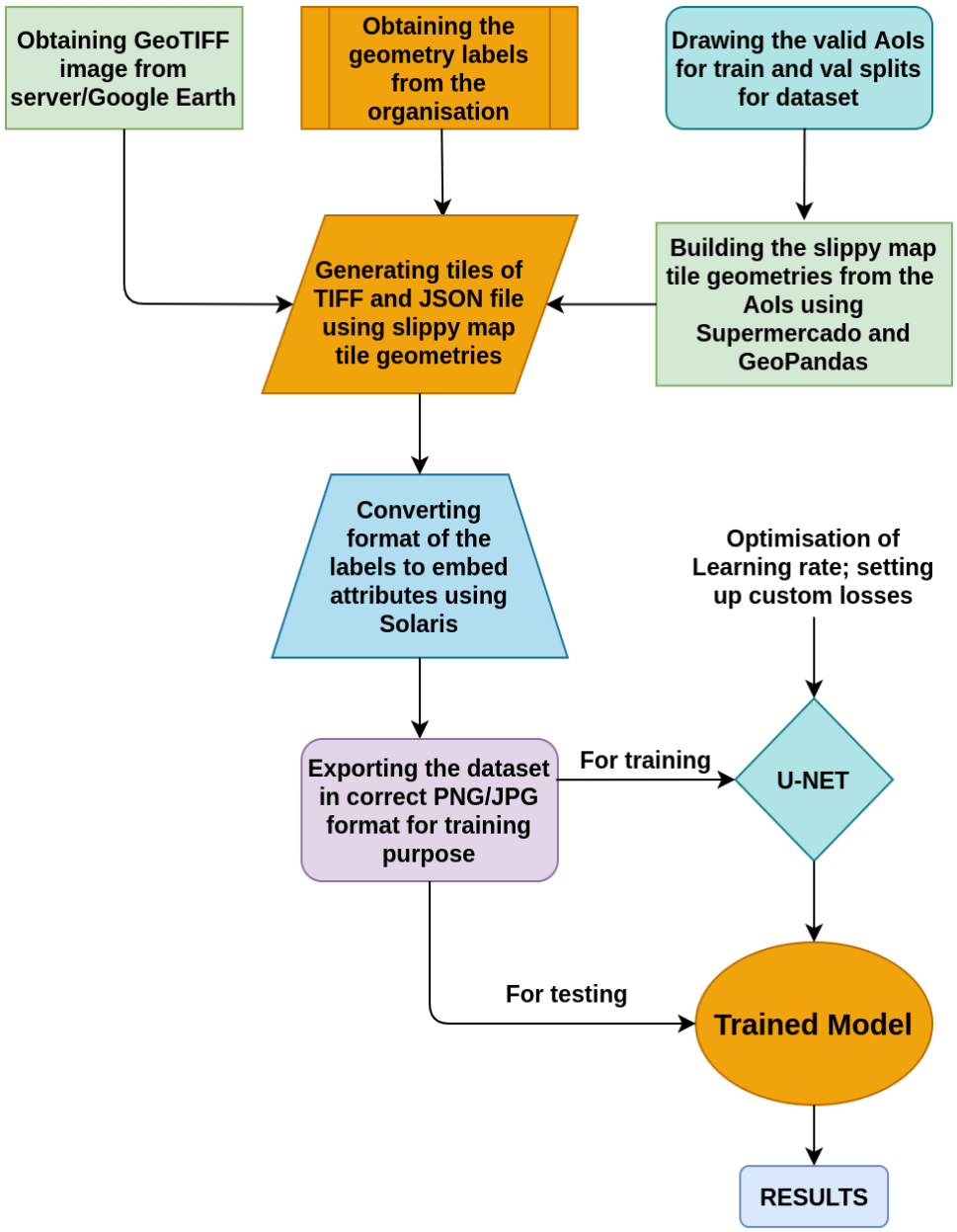


Figure 5: Workflow of the project

Semantic Segmentation - An Overview:

The project we are working on comes under the category '*Semantic Segmentation'* of Images. Semantic Segmentation is a slightly advanced version of Image Classification. It is all about labeling each pixel of an image with respect to a predefined set of classes.

Figure 6: How the image of a room looks like after Semantic segmentation

For example, in Figure 6, the left image is what we input to the system. Each pixel is then classified as 'Bed', 'Pillow', 'Wardrobe', 'Mirror', 'Carpet', etc. The output we get is essentially the same image, but with pixels coloured corresponding to their allotted classes.

Semantic segmentation is slightly different from Object detection. The reason is, in object detection boundaries of every object are detected by predicting bounding boxes around the object. Here, we do not differentiate between instances from the same object. For example, if multiple apples are kept on a table together, they are categorized to the same label. One wouldn't be able to distinguish between 2 apples. This is shown in Figure 7.

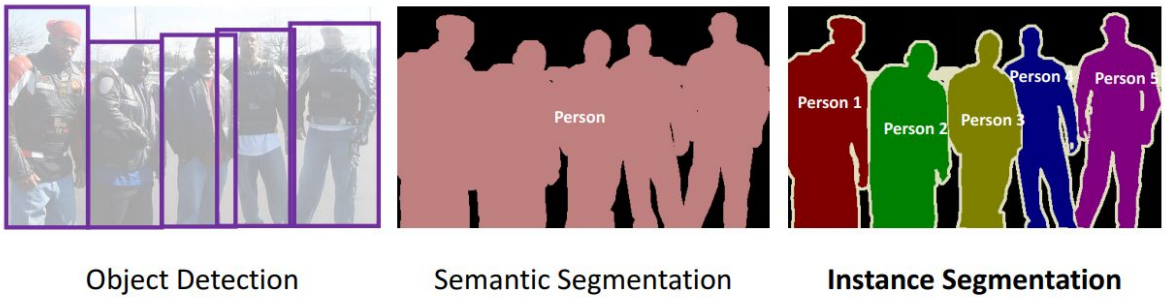


Figure 7: How Semantic segmentation is different from Object detection

When a 2-D (coloured) image is input to a computer, it is usually interpreted as an array of pixels; or more precisely - a **tensor**. A tensor can be represented with the shape: Width x Height x 3. The z dimension corresponds to the pixel values of the three R-G-B colour channels. A grayscale image is of the form Width x Height x 1, since it requires only one colour channel. During segmentation operations, the coloured images are converted to the shape Width x Height x 1, with each pixel assigned to its corresponding class (pixel value represents that class). Figure 8 indicates this concept.

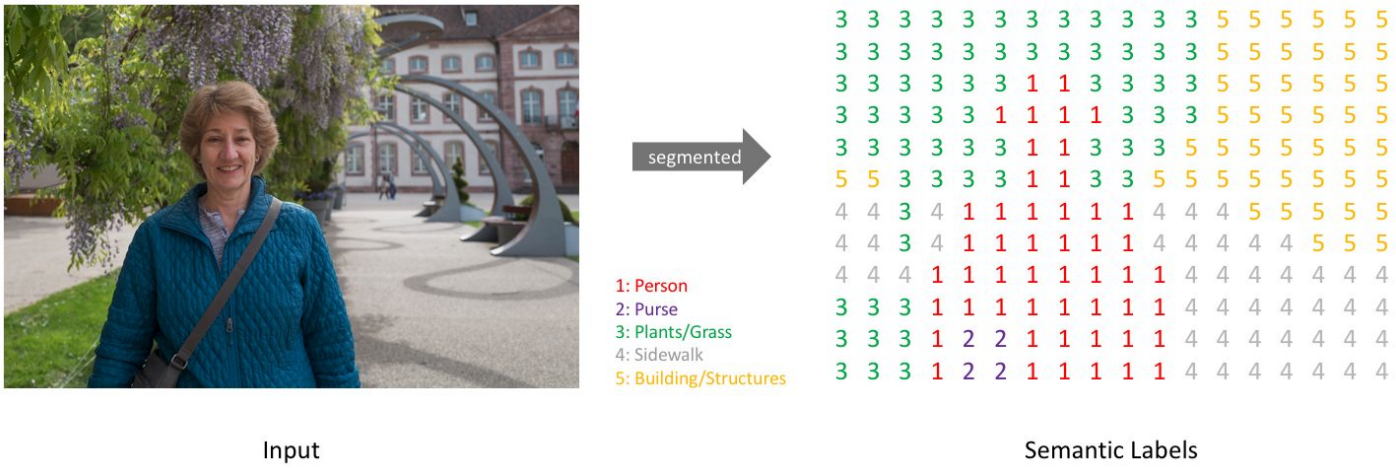


Figure 8: How the classification is actually done

We use one-hot encoding to create an output channel for each of the labels separately. This would be a multidimensional array showing a boolean output for each label arranged as ‘slices’ of the tensor.

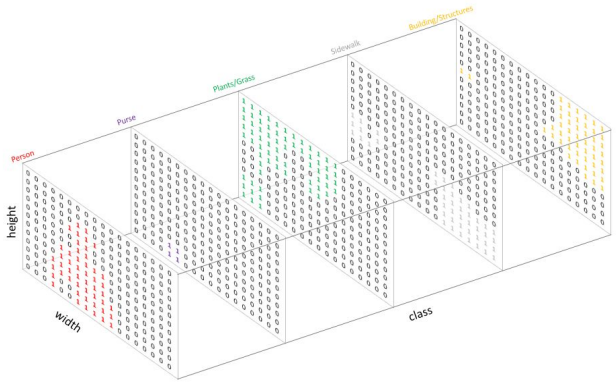


Figure 9: Output prediction

Its target/prediction can later be collapsed into a single channel and overlayed over the observation, using the **argmax function**, to form a mask, which illustrates the regions of an image to showcase where a distinct class is present.

Solution Architecture

Detailed Solution Architecture and Workflow:

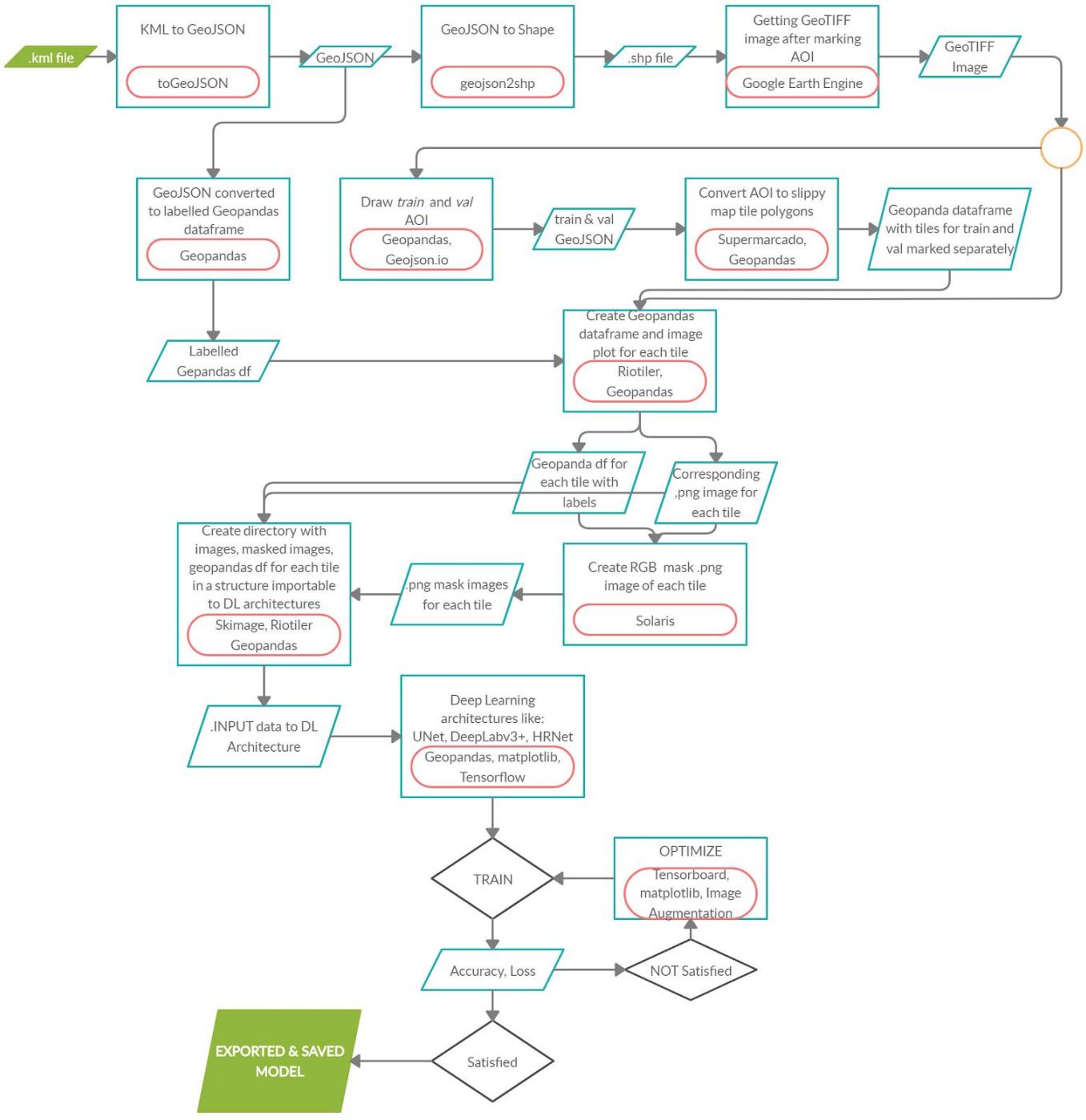


Figure 10: Overall Architecture used in the project

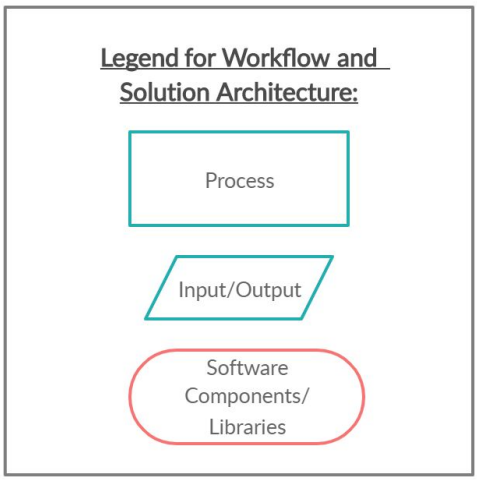


Figure 11: Legend for Figure 10

1) Google Earth Engine

Google Earth Engine (GEE) is a powerful web-based platform for cloud-based processing of remote sensing data on large scales. Satellite data and imagery from several sources can be used for analyzing and visualizing geospatial data sets. The coding language used for performing operations is JavaScript. Since we were unable to work directly in the RRSC center, we didn't have direct access to the satellite images; hence GEE was the perfect platform for us to use and export the required images for Satellites for the project.

The KML file, shared with us by the organisation, contained the labelings for the solar farms. We converted the KML to the GeoJSON format and then into three files in the formats - .SHP, .PRJ and .SHX. These three files were used as an input to the Google Earth Engine.

Figure 12: Forming polygons over labelling for filtering satellite images

When the labelings were imported on the GEE, Figure 12 (left) accurately describes how they could be visualized . The satellite images corresponding to these labelings are required. For our project, we used two satellite imageries - Sentinel-2 and Landsat-8. Landsat-8 has a spatial resolution of 30m whereas Sentinel-2 has 10m.

'Polygons' were used for obtaining satellite images corresponding labels. These are geometrical shapes that could be drawn on the map and used for filtering the required geometry. This is shown in Figure 12 (right). Other filters were also used in order to get the right images. R-G-B bands (i.e. B4, B3, and B2) bands were separated from the other bands in the image collection. The Median function was used to find out the median of all images in the remaining collection to obtain a single image. The code snippet is depicted in Figure 13.

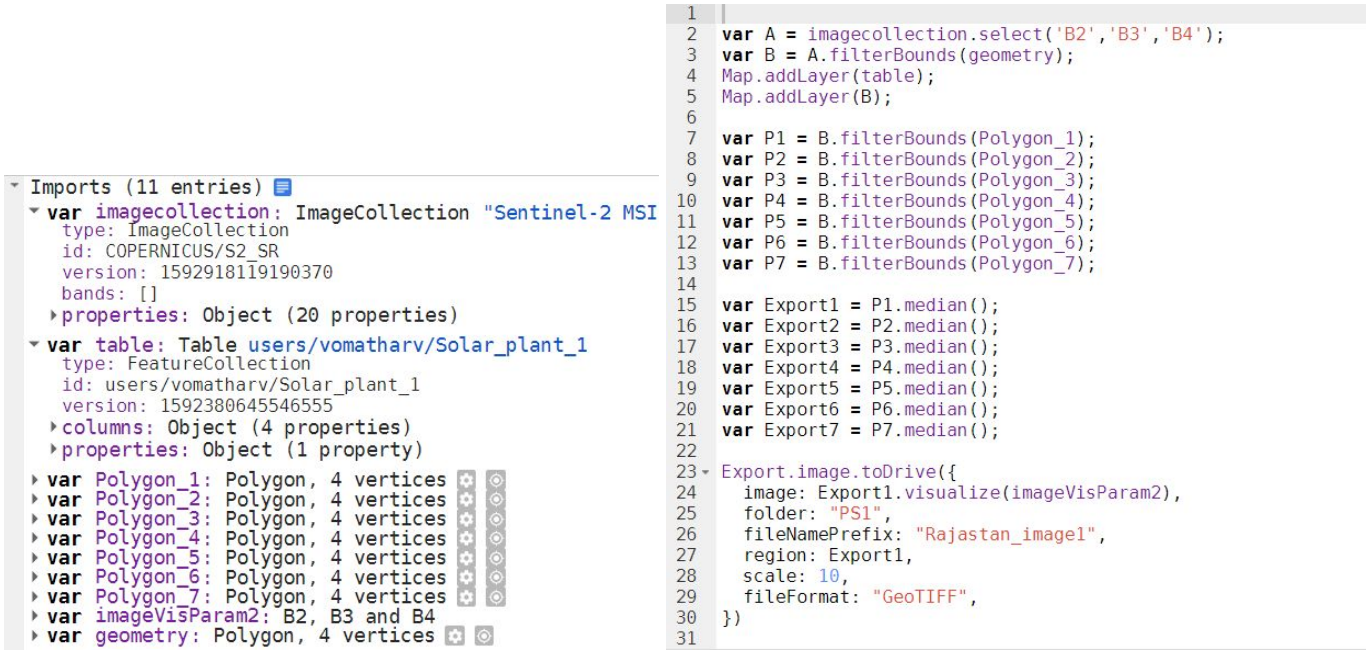


Figure 13: Code snippets

Once all the polygons were made, and all the filters were done, the Export function coupled with other Image parameters, was used to export a GeoTIFF image to the Google drive. And this image could be downloaded and used on Jupyter or collab for performing further preprocessing. Figure 14 (left) shows the satellite image along with the Polygons. Figure 14 (right) shows the Task tab accessed to specify the characteristics of the export.

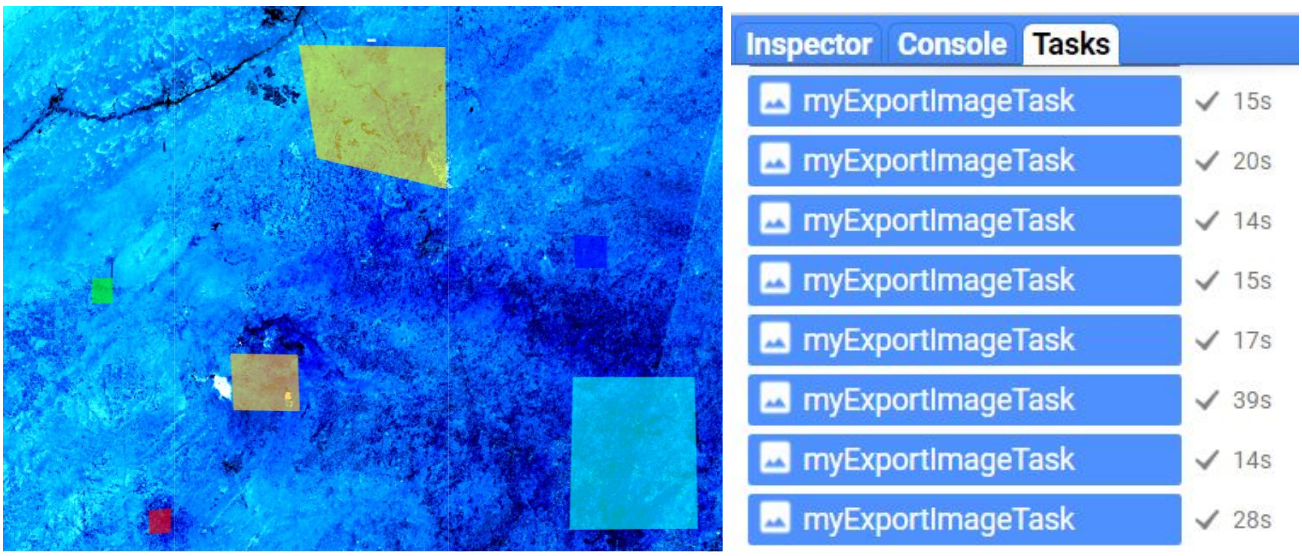


Figure 14: Sentinel-2 layer and output task

2) Image pre-processing techniques

For acquiring grip over the actual project work, our team was assigned to perform a practice project that involved the segmentation of buildings in drone imagery. We had used the Tanzania Open AI Challenge Dataset for this purpose, which consisted of 7 cm high-resolution drone imagery. Since the imagery was in geospatial raster TIFF and JSON format, we made use of some geoprocessing data science tools for efficiently previewing the dataset and converting it into a valid deep learning dataset, i.e. a .PNG/.JPG format. In particular, the following libraries and tools were used:

1. **Geopandas:** It is an extension of the Pandas' Python framework, and aids the user in executing spatial operations on geospatial geometries such as Points, Lines, Polygons, etc quite effectively. It allows one to easily read, write & manipulate vector data (like .geojson) and returns a DataFrame object for the same. A DataFrame is basically a table, which consists of some geometries and properties associated with it. For e.g. following is an instance of GeoPandas' DataFrame:

idx	pop_est	continent	name	geometry
1	53950935	Africa	Tanzania	POLYGON(coordinates)
2	326625791	North America	USA	POLYGON(coordinates)

Table 1

Here, the geometries are listed under 1 column and the properties are listed adjacent to it. The geometries can be plotted on a map using the .plot() function.

2. **Rasterio:** Rasterio is a special Python-based library that is used to read and write the rasterized imagery such as GeoTIFF images. Rasterization is the process of representing images as a collection of smaller segments (or pixels). When these segments are displayed together, they represent the complete picture. Since aerial images are of very high resolution in nature, rasterizing them helps to reduce computation cost and promotes efficient management of smaller image segments.

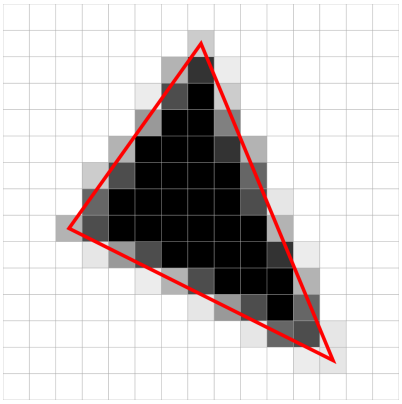


Figure 15

3. **Supermercado:** This library helps to divide a geometry into uniformly shaped square Mercator tiles. These tiles carry additional information along with them, i.e. X, Y & Z. X and Y represent the tile coordinates whereas Z represents the zoom level at which tiling was done.

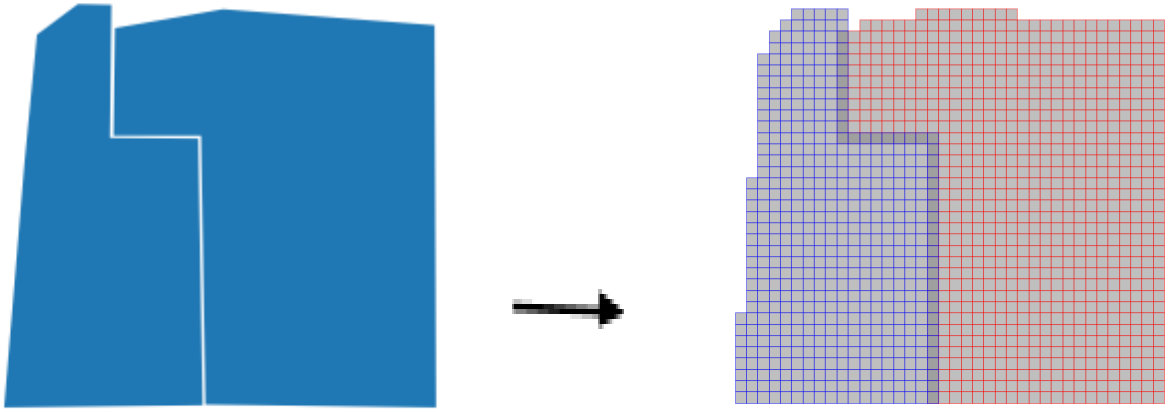


Figure 16

4. **Rio-Tiler:** It is a special Rasterio plugin, which helps to break the GeoTIFF raw image and Geojson geometry label into the tiling format as generated by supermercado. This is an essential step for generating the dataset, since using this library, we break a high-resolution TIFF image and JSON labels to a normal dataset that can be used for training a deep learning model. Each image tile

generated is now a normal image in PNG format.

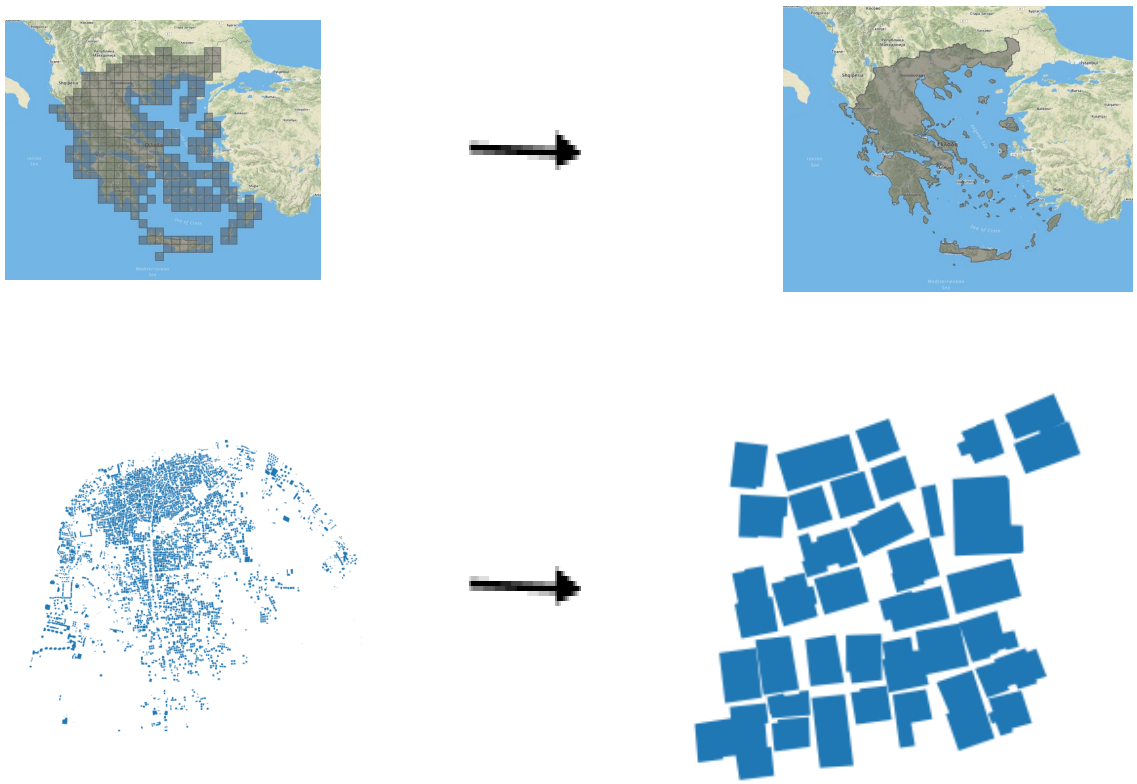


Figure 17

5. **Solaris:** This library helps in converting the image labels into a special 3 channel RGB format. The 1st channel represents the building footprints, the 2nd channel represents building boundaries, whereas the 3rd channel represents the contact areas between the adjacent geometries. By converting the labels into this format, the deep learning model will be better able to appreciate the boundaries, footprints, and closeness of the objects, since we have embedded this information into the labels in the form of channels.

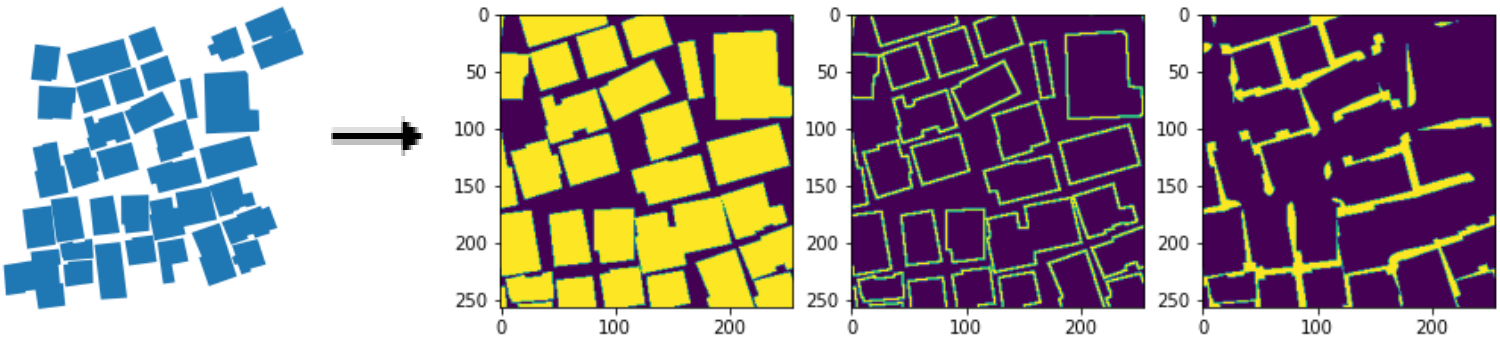


Figure 18

Workflow of the Pre-Processing module:

1) To generate the dataset, we made use of the **znz001** image grid of the dataset, which covers the northern tip of Zanzibar's main island of Unguja. We downloaded the GeoTIFF raw image of znz001 as well as the corresponding building labels in geojson format.
2) Using geojson.io, we divided the image map into training and validation areas of interest (AoIs), by drawing the required geometries. This is done to divide the dataset into train and val splits.
3) For the generation of slippy map tiles (or Mercator tiles), we executed the supermercado library on the two AoIs extracted in step 2. The tile size was set at 256 pixels and a zoom level of 19. These tiles were then stored in a geojson format. For dropping the duplicate tiles and adding some extra meta-information (i.e. XYZ locations of the tiles) to the tile file, we opened it with geopandas and made the necessary changes.
4) Using the tile geometries stored in the geojson tile file, we then mapped them and cropped the GeoTIFF image and geojson labels obtained in step 2. This way, we cropped tiles of size 256x256 from the original image and label. These newly generated tile images have a normal size and are quite easy to manage, as compared to directly using high-resolution GeoTIFF images.
5) We then used the Solaris library, which helped us to reformat the building labels for correctly executing the segmentation task. The simple RGB channels of the labels were converted into 3 channels having the information of building footprints, boundaries, and contact areas.
6) Thus, we made a huge collection of training and validation images from just a single znz001 image grid of the dataset. We then used this dataset for training the U-Net segmentation architecture.

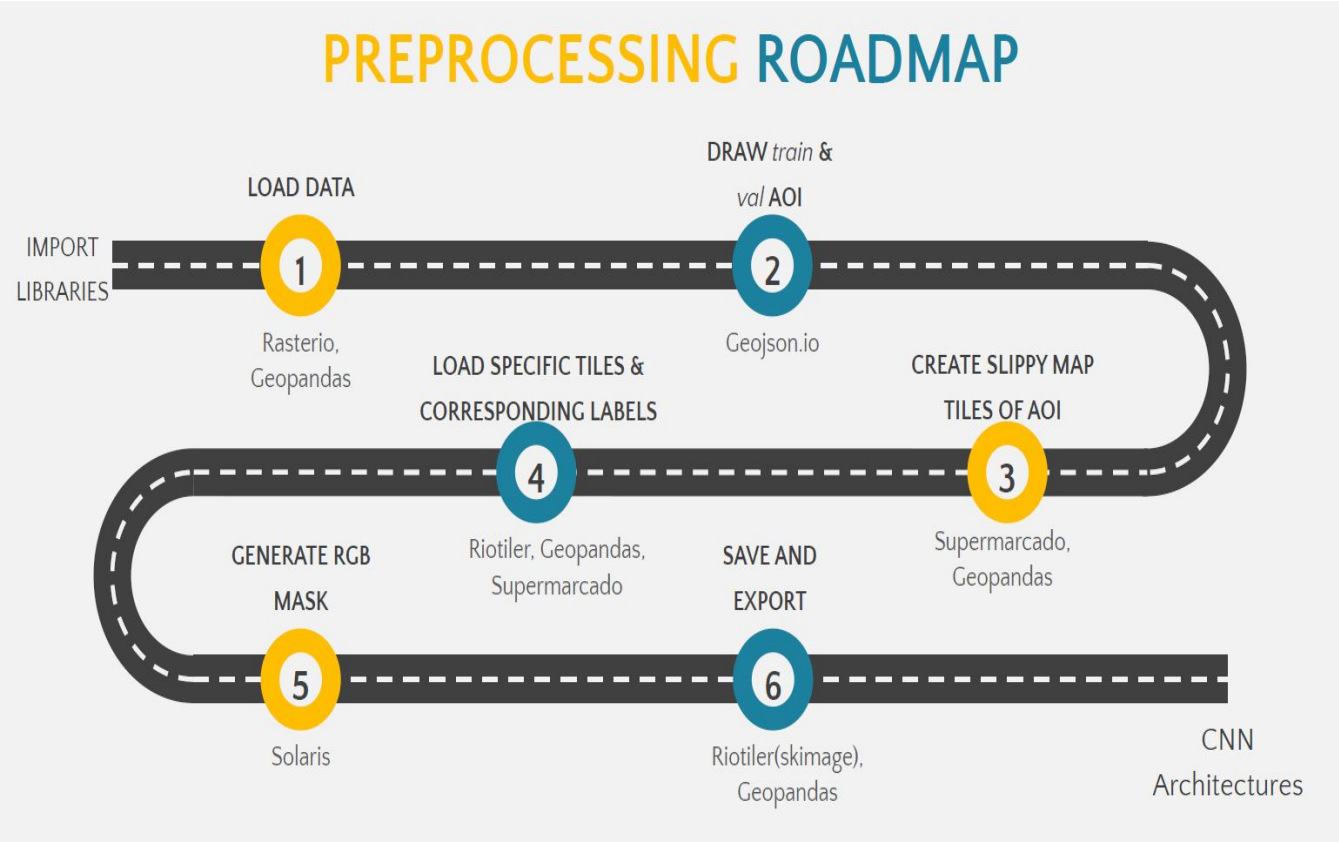


Figure 19: Road Map

3) Data Augmentation Techniques:

Deep Learning based approaches for semantic segmentation adopt very heavy and deep architectures for training the detector/model on data. For such deep neural networks, if the data is obsolete or very less in quantity, the model starts suffering from the issue of **'Overfitting'**, which hampers the performance of the model. Basically, in overfitting, the model starts **rote learning** the representations present in the training images. This causes the model to perform extremely poorly on test images or unseen images to be precise.

To solve this problem, we increased the size of our dataset by heavily augmenting the images. Initially, we had 95 images that were extracted from the Google Earth Engine. Using Data Augmentation, we increased this data to around 5 times, i.e. 500 images. This helped the model to better understand the varied features and representations and perform well on the test set too.

Library Used: https://github.com/mdbloice/Augmentor

Types of Augmentations Used:

1) **Flip Left, Right, Top & Bottom**

   This type of augmentation inverts the image towards the specified direction, i.e. either left, right, top or bottom, depending on the probability distribution we assign it to. Below is one example for flip right augmentation:

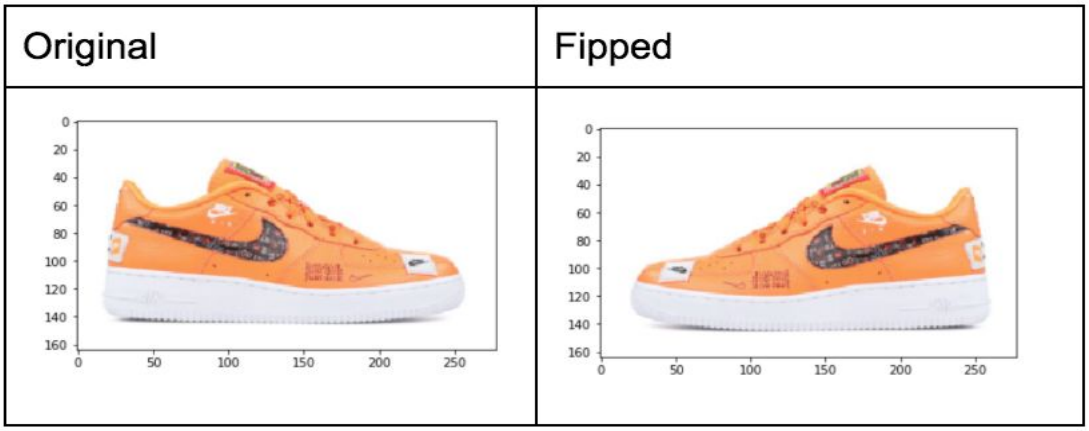


Figure 20

2) **Random Rotations**

   This augmentation randomly rotates and zooms in an image so that it looks quite different from the parent image. We can specify the max & min value for this rotation range.

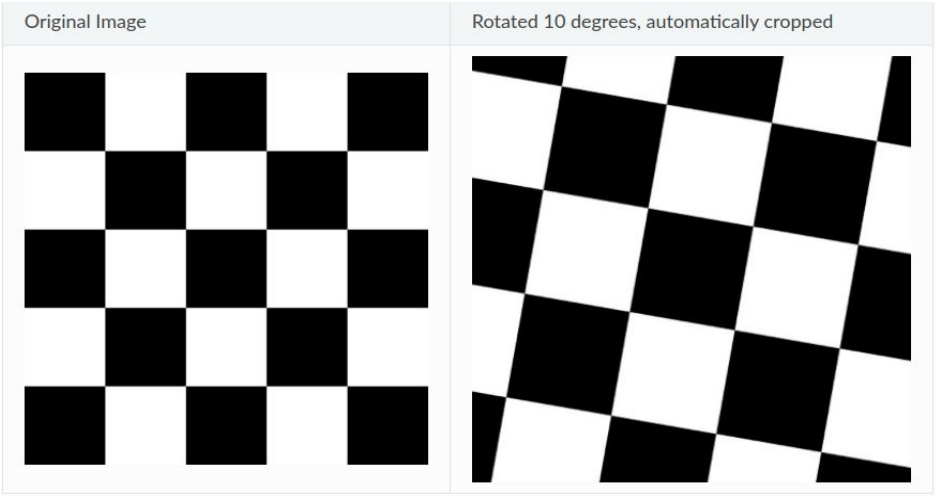


Figure 21

3) Shears

We also subject the dataset to shear augmentation which basically displaces some pixels of the image to some extent. Below is an example:

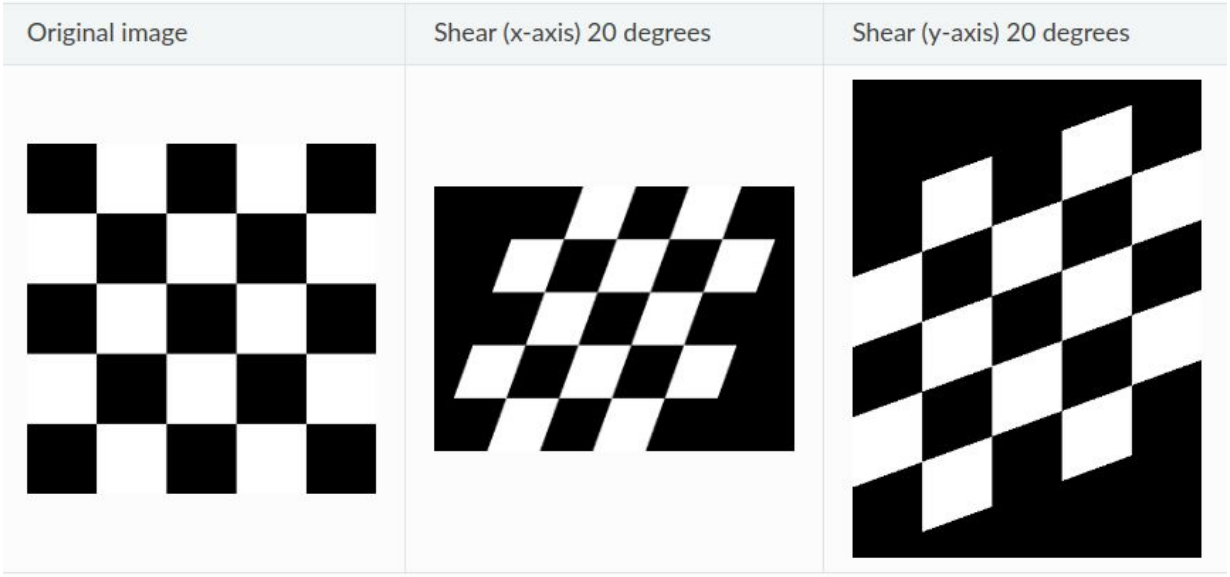


Figure 22

Code Snippet Used for Augmentation:

```
import Augmentor


p = Augmentor.Pipeline("./images")

# Point to a directory containing ground truth data.

# Images with the same file names will be added as ground truth data

# and augmented in parallel to the original data.

p.ground_truth("./masks")

# Add operations to the pipeline as normal:

p.rotate(probability=0.9, max_left_rotation=5, max_right_rotation=5)
```

```
p.flip_left_right(probability=0.7)

p.zoom_random(probability=0.6, percentage_area=0.8)

p.flip_top_bottom(probability=0.8)

p.shear(probability=0.7, max_shear_left=12, max_shear_right=10)

p.sample(20)
```

Figure 23: Code Snippet

4) Semantic Segmentation with UNET Architecture

1. We make use of the fastai library which is built on Pytorch as our deep learning framework.
2. We also use Google Colab GPUs of certain memory size to ensure that the model trains without facing any memory issues and a reasonable training time per epoch.
3. Then, we implement code to ensure that the tile images and their mask files are fed correctly for the training and validation of the model.
4. We then verify if the images and their masks are matched appropriately.

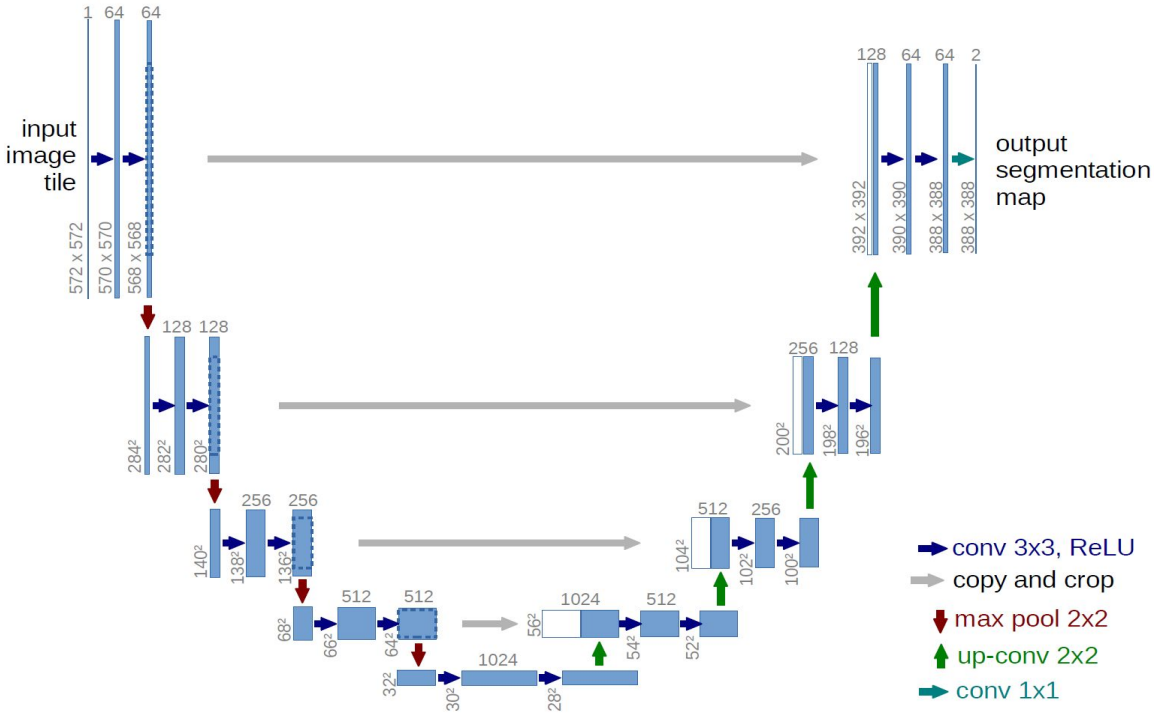


Figure 24

Structure of Dynamic UNET:

1. The Dynamic UNET model is made of a pre-trained encoder (downsampling) network on the left side and a decoder (upsampling) network on the right side (this is where the name UNET is derived from, due to the U-shape of the network).
2. While the downsampling operation involves convolution and max-pooling operation, the upsampling operation makes use of transposed convolution and concatenation followed by convolution to get the output. This is done so that the feature maps are of the same size as from the encoder network since the output of segmentation is the same as the input image.
3. We concatenate the decoder feature maps with the corresponding encoder feature maps for better localization and to prevent loss of features.

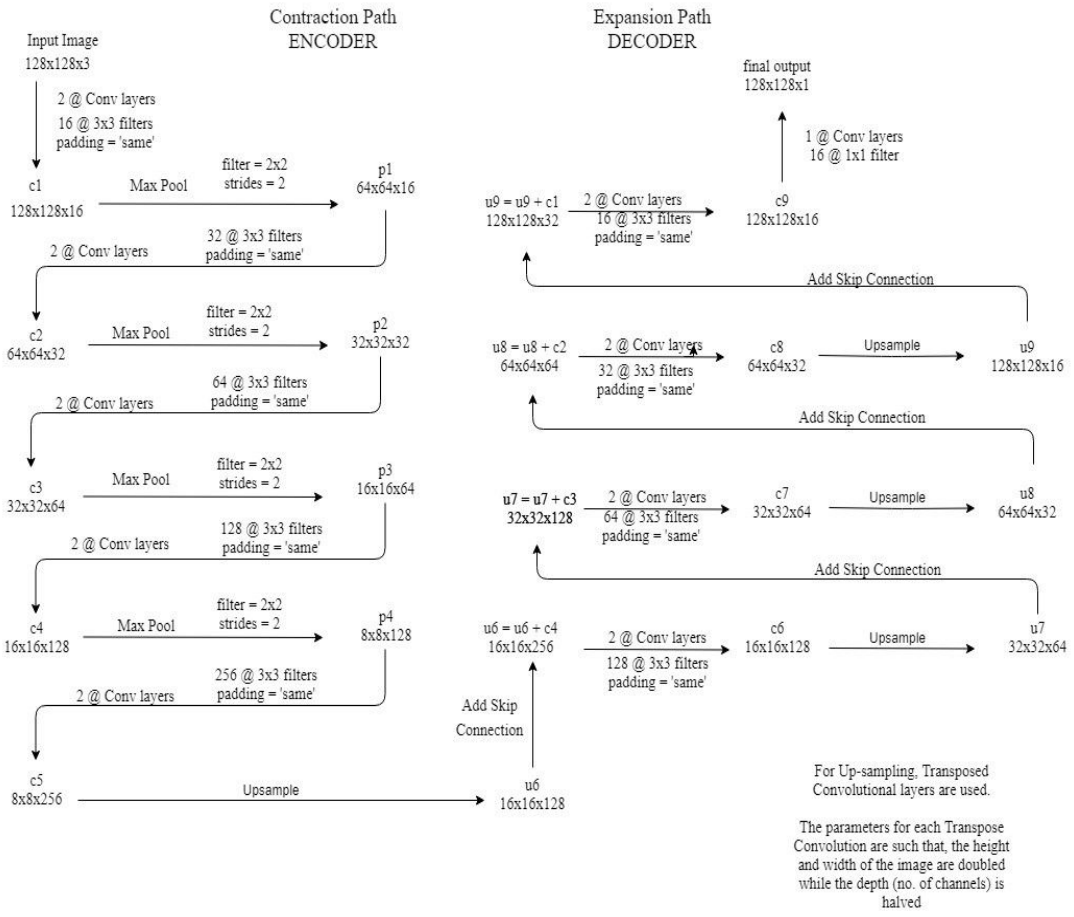


Figure 25

4. Other key features of the Dynamic UNET model used are the 1cycle learning method, AdamW optimization, and pixel shuffle technique.
5. The loss function used is a combination of dice and focal loss.

Training Methodology:

1. First, we make fine adjustments to the decoder part of our network (keeping the pre-trained encoder weights fixed).
2. Then, we unfreeze all the layers and train for some more epochs.
3. We monitor the dice score and extract the best model at the end.
4. We also test which images produce the highest error.
5. Finally, we test if the model generalizes well on unseen drone images with varying zoom levels and test if the model takes more time on GPU or CPU.

5) DeepLab V3+ Architecture

DeepLab V3+ model was invented by Google in 2018 and performs very well on Semantic Segmentation tasks. It is a state-of-the-art architecture for Semantic Segmentation and has secured 2nd place in the PASCAL VOC challenge. The salient features of the DeepLab V3+ model are as follows:

1. Atrous Spatial Pyramid Pooling (ASPP) Convolution in encoder for capturing localization information of various scales.
2. An encoder decoder module as the core build up.
3. Use of Xception-65 pretrained model which helps in extracting the features from the images.

Atrous Convolution

Atrous convolution is carried out by means of an ‘atrous rate’ which basically is the number of gaps present between 2 adjacent pixels of the filter kernel. For e.g. the above image has atrous rates of 1, 6 and 24. The naive atrous convolutions are supplemented with depth wise separable convolutions in DeepLab. Depthwise separable convolution splits a regular convolution process into a depthwise convolution (to change the resolution of the image feature map) followed by pointwise convolution to adjust the number of channels, which reduces the computational complexity. Thus, atrous depthwise convolutions, which are a hybrid of various techniques play a key role in generating correct predictions.

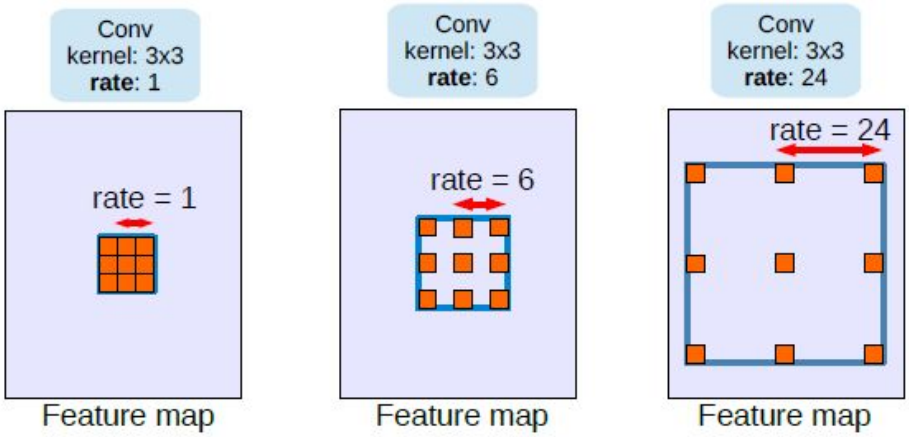


Figure 26

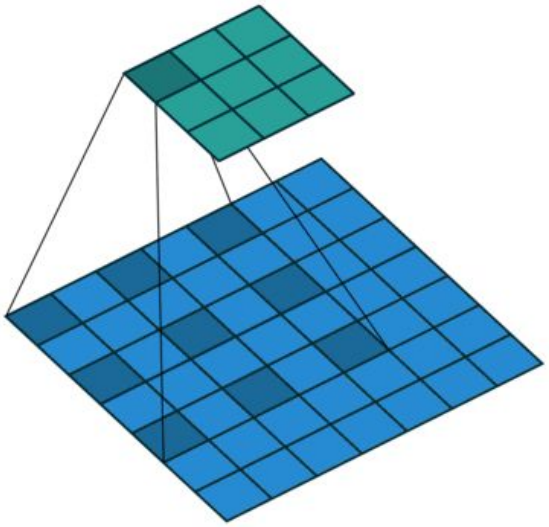


Figure 27

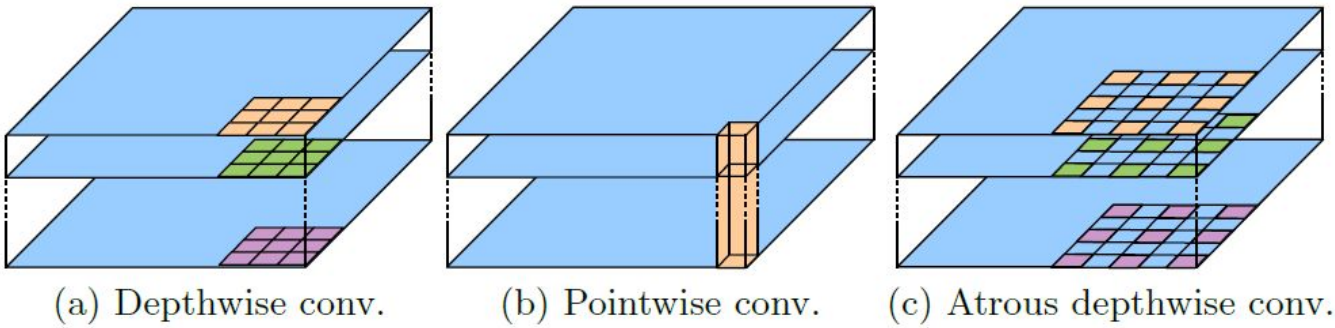


Figure 28

Dilated (atrous) convolution allows an adjustable field of view We deploy these dilated convolutions in the form of an **ASPP** module, which stands for **Atrous Spatial Pyramid Pooling**. In ASPP, a tensor is passed parallely to 4 types of dilated convolutions (having different atrous rates). All the resulting output feature maps from these convolutions are collected and then concatenated to form the output tensor. This tensor consists of a lot of information about the type of objects and their scale variations. Below is a visualisation:

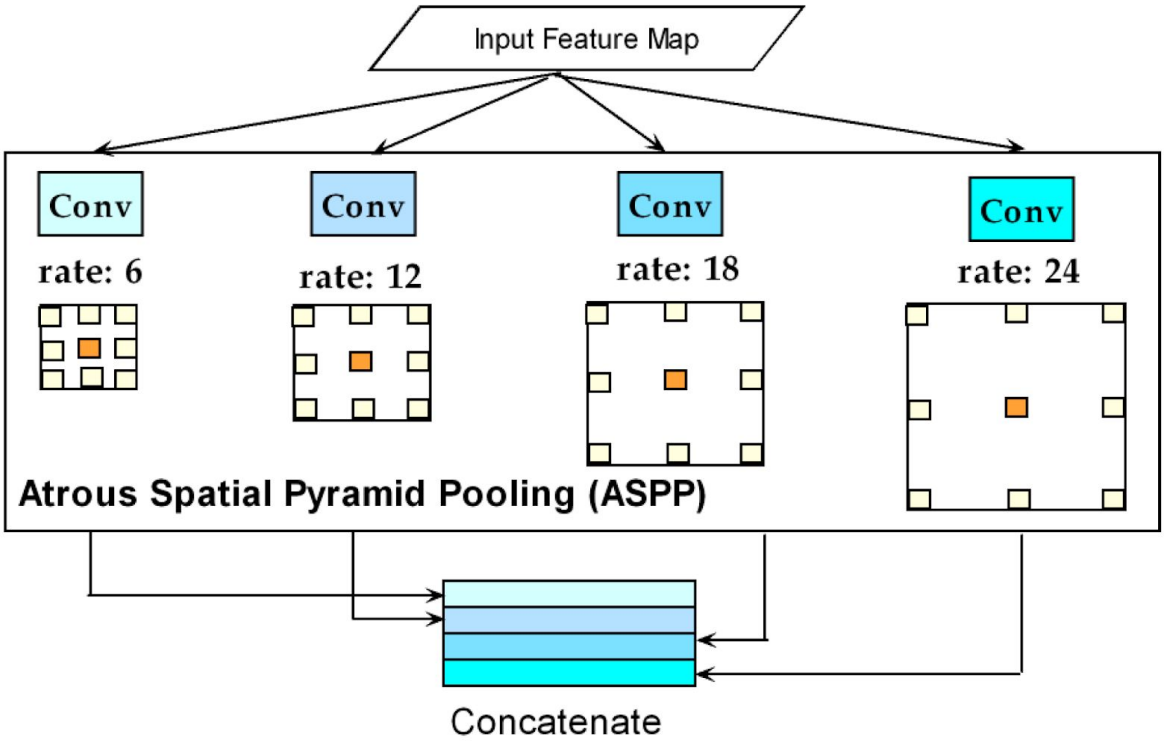


Figure 29

Architecture of Encoder-Decoder Model

Encoder Architecture

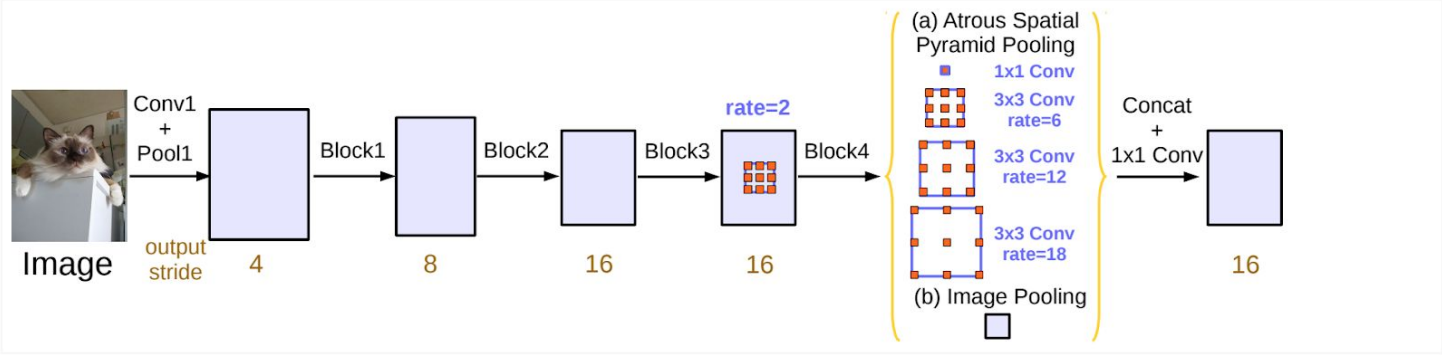


Figure 5. Parallel modules with atrous convolution (ASPP), augmented with image-level features.

Figure 30

Output stride is defined as ratio of input to output feature map resolution. In this encoder model, we perform successive convolution operations followed by an atrous convolution of rate=2 and we set the output stride to 16 (i.e. the feature map size at the end of this encoder is 16 times smaller). Basically, the encoder module decreases the feature map size and simultaneously increases the number of channels in the tensor. It does so using the pretrained weights of the Xception-65 backbone. The tensor is then made to pass through the ASPP module which is responsible for multi scale capturing of features present in the images. The outputs from the encoder network is sent to the Decoder for decoding the 'WHERE' information.

Decoder Network

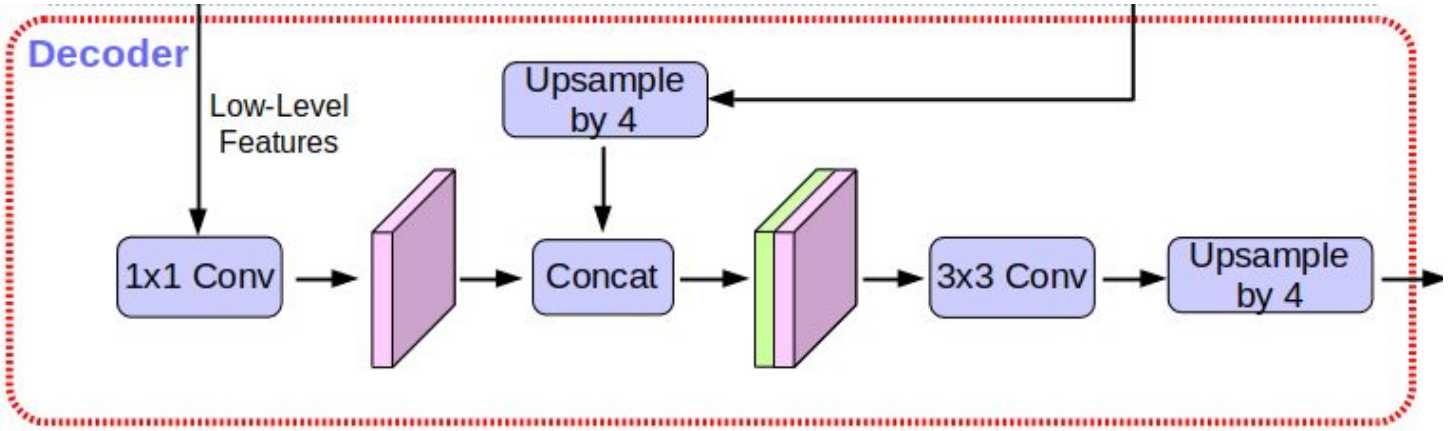


Figure 31

The decoder network obtains input maps from two sources- The feature maps from the Xception-65 backbone and the ASPP module are obtained (after upsampling) and are concatenated and merged together. This is followed by a 3X3 convolution and then upsampling by 4, after which we get the semantic predictions of the model. The Encoder, ASPP and decoder are joined together to arrive at the final model:

Encoder-Decoder Network Final Architecture

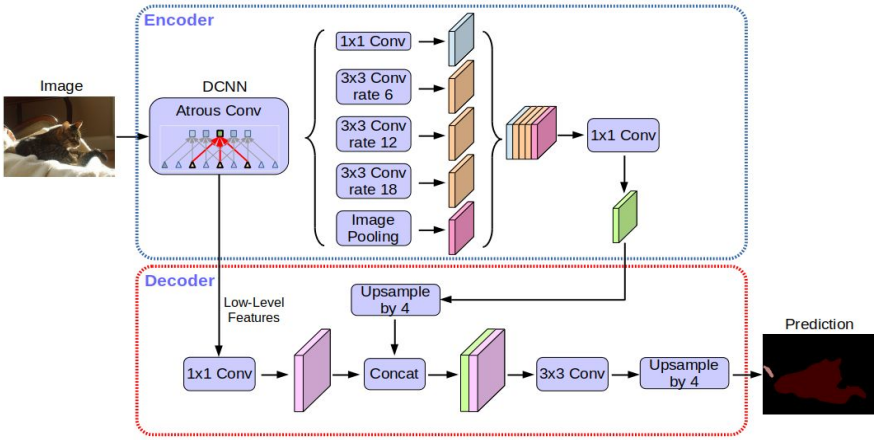


Figure 32

6) Post-processing

After training the data set if we want to evaluate our model performance against the reality we should use a different set of labeled data on which our model has not seen before. Therefore we will again make use of our libraries such as Solaris and geopandas and using them convert our model's prediction as 3-channel pixel raster output into GeoJson file by following these steps:

1) Combining our rasterized output prediction into a 1 channel binary pixel mask.
2) Polygonizing this binary pixel mask obtained in the 1st step into vectors which will represent the predicted output for each building.
3) Georegistering the display coordinates of these vectorized building shapes into longitude, latitude coordinates.

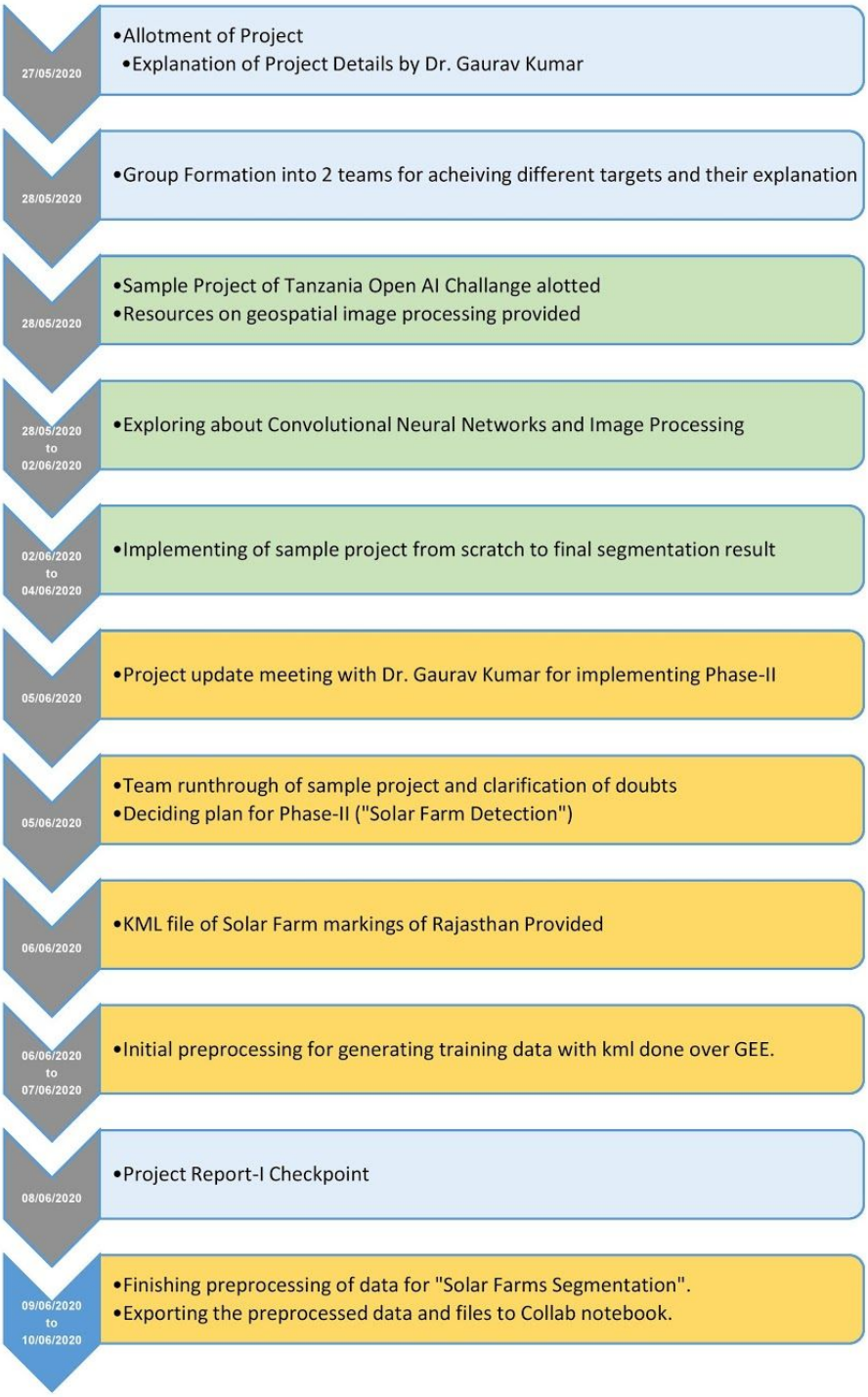
After the last step, we can calculate the F1 score which is a common evaluation metric for models applied to satellite imagery.

Some techniques for better accuracy :

1) Larger training and test datasets can be used so that models can cover a more different variety of images.
2) The images selected should be such that the ratio of building pixels per image should be as high as possible.
3) The model should be tested on many different architectures such as UNET,resnet50,densenet, etc so that the best one can be identified and accuracy can be improved.
4) Different combinations of hyperparameters and loss functions can be used while training the model.
5) Data augmentation techniques can be used to increase the size of the dataset and avoid underfitting etc.
6) Training the model for different learning rates and for larger numbers of epochs can also have a great impact on accuracy.

Project Plan:

1) Initial Plan and checkpoint as on 08/06/2020



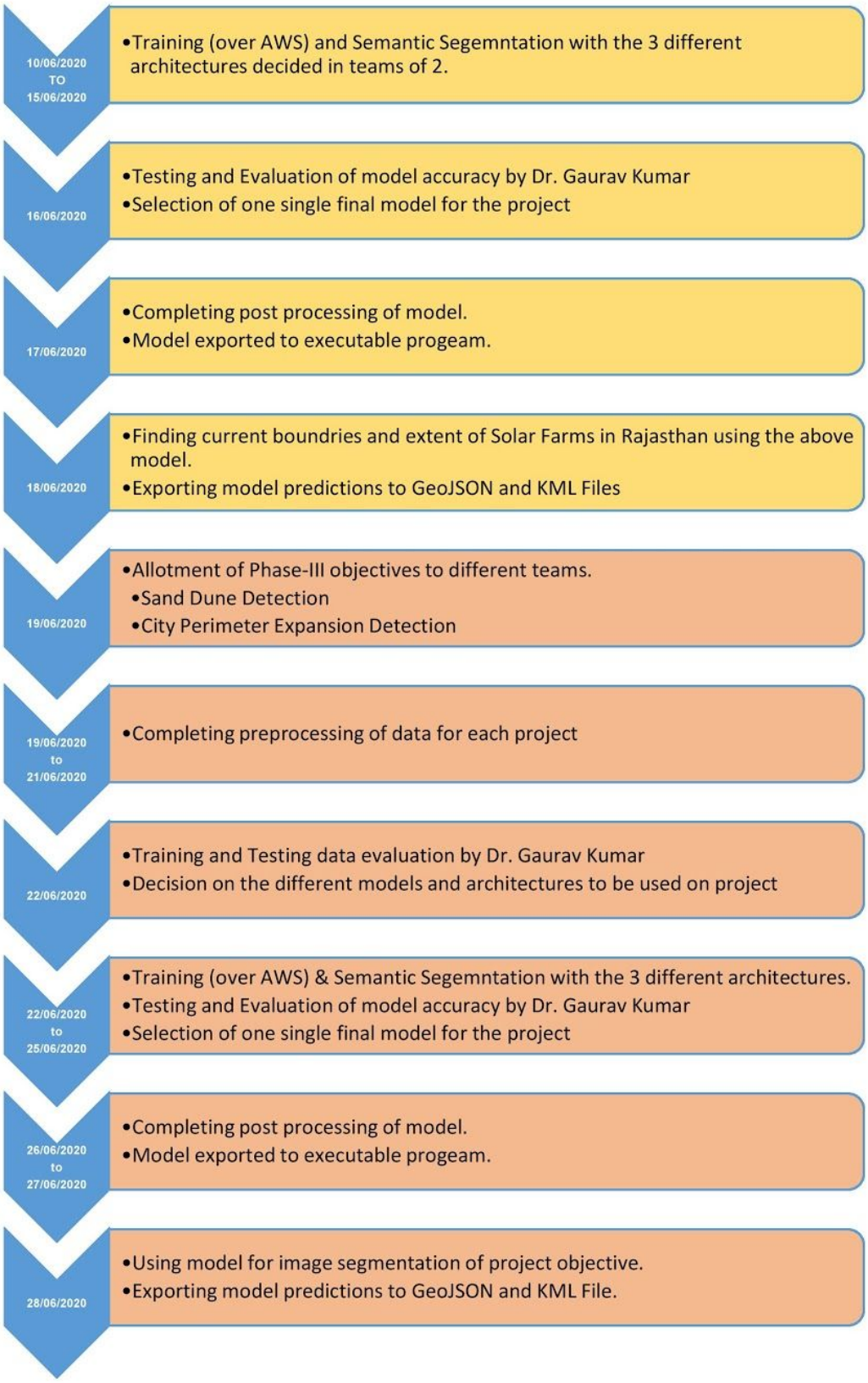
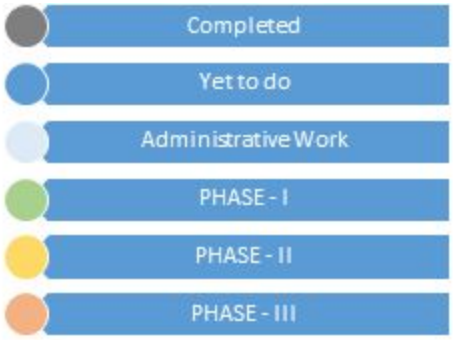

10/06/2020 TO 15/06/2020
•Training (over AWS) and Semantic Segemntation with the 3 different architectures decided in teams of 2.
16/06/2020
•Testing and Evaluation of model accuracy by Dr. Gaurav Kumar
•Selection of one single final model for the project
17/06/2020
•Completing post processing of model.
•Model exported to executable progeam.
18/06/2020
•Finding current boundries and extent of Solar Farms in Rajasthan using the above model.
•Exporting model predictions to GeoJSON and KML Files
19/06/2020
•Allotment of Phase-III objectives to different teams.
•Sand Dune Detection
•City Perimeter Expansion Detection
19/06/2020 to 21/06/2020
•Completing preprocessing of data for each project
22/06/2020
•Training and Testing data evaluation by Dr. Gaurav Kumar
•Decision on the different models and architectures to be used on project
22/06/2020 to 25/06/2020
•Training (over AWS) & Semantic Segemntation with the 3 different architectures.
•Testing and Evaluation of model accuracy by Dr. Gaurav Kumar
•Selection of one single final model for the project
26/06/2020 to 27/06/2020
•Completing post processing of model.
•Model exported to executable progeam.
28/06/2020
•Using model for image segmentation of project objective.
•Exporting model predictions to GeoJSON and KML File.

Figure 33



Legend

Figure 34

2) Gantt chart of actual progress:

Gantt chart is an important project management tool used by organizations to track the progress (%Done) of their project, subtasks, project leads and also to map efficiency in terms of number of days the various tasks took. We have used a similar modified version of Gantt chart created over MS Excel which shows how our project and it's different subtasks progressed throughout the 6 weeks. This can also be used to map the time spent on administrative tasks, delays in comparison to actual project work and implementation.

Semantic Segmentation of Satellite Images for different Objectives

BITS Pilani in collaboration with RRSC, Jodhpur

Project Start Date 5-18-2020 (Monday) Display Week 1

Project Lead Dr. Gaurav Kumar, Prof. Vishwanathan Hariharan

WBS	TASK	LEAD	START	END	DAYS	% DONE	WORK DAYS
1	**PS I - Semantic Segmentation**			-			-
1.1	PS I Allotment		Mon 5-18-20	Mon 5-18-20	0	100%	1
1.2	Introductory Meet		Wed 5-20-20	Wed 5-20-20	1	100%	1
1.3	Project Allotment Gap		Wed 5-20-20	Wed 5-27-20	8	100%	6
1.4	Building Segmentation Project Allotment		Thu 5-28-20	Thu 5-28-20	0	100%	1
1.5	Learning Sessions - CNN, Image Processing		Thu 5-28-20	Tue 6-02-20	6	100%	4
1.6	Building Segmentation Project		Tue 6-02-20	Fri 6-05-20	4	100%	4
1.7	Solar Farm Segmentation Project Allotment		Sat 6-06-20	Sat 6-06-20	0	100%	0
1.7.1	Pre-processing for GEE data		Sat 6-06-20	Wed 6-10-20	5	100%	3
1.7.2	Training & Optimizing with Unet		Wed 6-10-20	Sun 6-14-20	5	100%	3
1.7.3	Pre-processing for OpenStreetMap.org data		Sun 6-14-20	Wed 6-17-20	4	100%	3
1.7.4	Training & Optimizing with DeepLabv3+		Thu 6-18-20	Sat 6-20-20	3	100%	2
1.8	Sand Dune Segmentation Project Allotment		Thu 6-18-20	Thu 6-18-20	0	100%	1
1.8.1	Pre-processing for GEE data		Thu 6-18-20	Sat 6-20-20	3	100%	2
1.8.2	Pre-processing for OpenStreetMap.org data		Sun 6-21-20	Sun 6-21-20	1	100%	0
1.8.3	Training & Optimizing with Unet		Sun 6-21-20	Wed 6-24-20	4	60%	3
1.8.4	Training & Optimizing with DeepLabv3+		Wed 6-24-20	Thu 6-25-20	2	0%	2
1.9	Submission of Models and Precitions		Fri 6-26-20	Fri 6-26-20	1	100%	1

Figure 35

3) Project Task division:

S.No.	Task	Prateek Garg	Omatharv Vaidya	Abichal Ghosh	Parag Rathi	Rohan Sachan	Siddhartha Goswami
1	**Building Segmentation Project**						
2	Pre-processing of data						
3	Training with Unet						
4	Post Processing and Optimization						
5	**Solar Farm Segmentation Project**						
6	Extraction of data from GEE						
7	Pre-processing for GEE data						
8	Training & Optimizing with Unet						
9	Extraction of data from OpenStreetMap.org						
10	Pre-processing for OpenStreetMap.org data						
11	Training & Optimizing with DeepLabv3+						
12	**Sand Dune Segmentation Project**						
13	Extraction of data from OpenStreetMap.org						
14	Pre-processing for OpenStreetMap.org data						
15	Training & Optimizing with Unet						
16	CASE STUDY: Semantic Segmentation (Stanford)						
17	CASE STUDY: Solar Farm Segmentation in China						
18	Seminar I ppt						
19	Midsem Project Report						
20	Seminar II ppt						
21	Compre Project Report						
		Highlighted columns mark the task done by indiviudal members of the team.					

Figure 36

Results:

The results after applying UNET:

We trained the UNet architecture end-to-end on the Solar Farms and Sand Dunes dataset. We then tested the model using some unseen images from the validation set. Below are some of the results:

Training loss for the solar farms dataset:

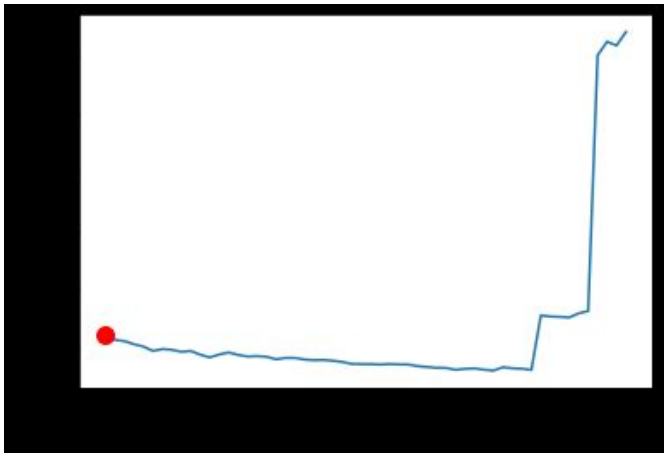


Figure 37

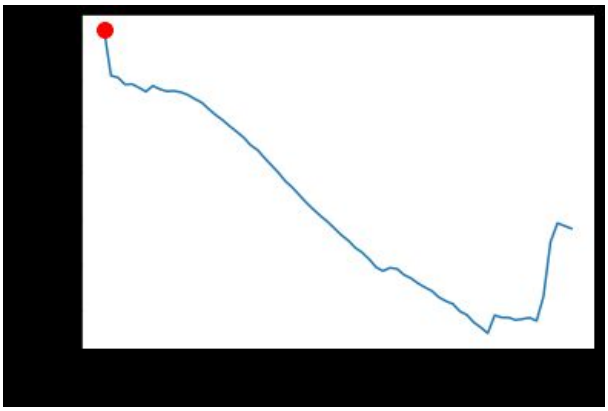


Figure 38

Results of Solar Farms Predictions using UNET:

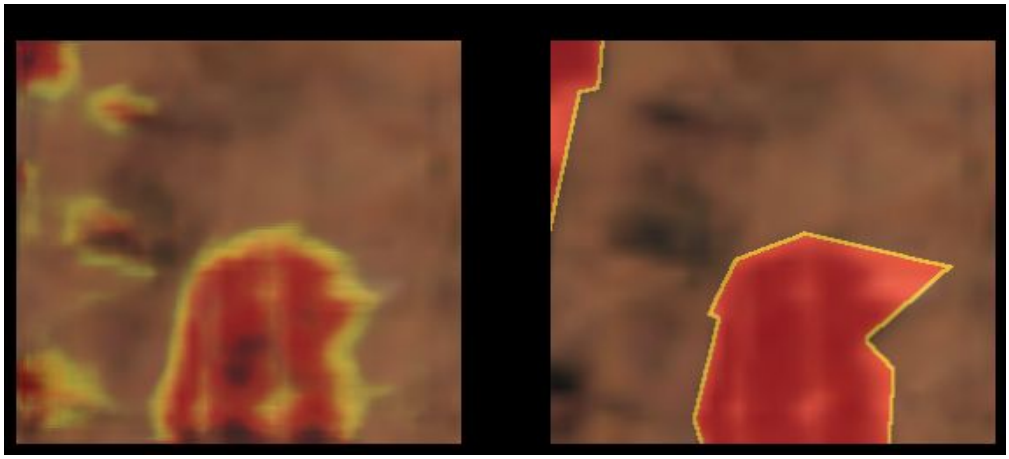


Figure 38

Figure 39

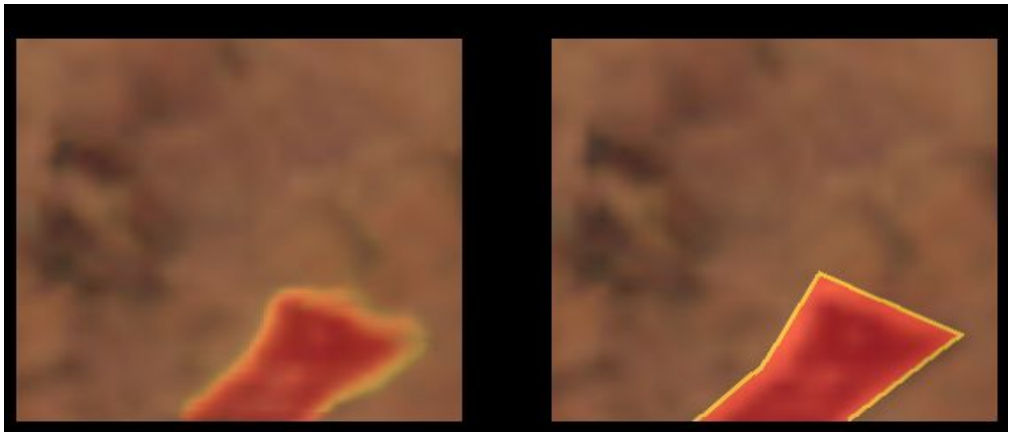


Figure 40

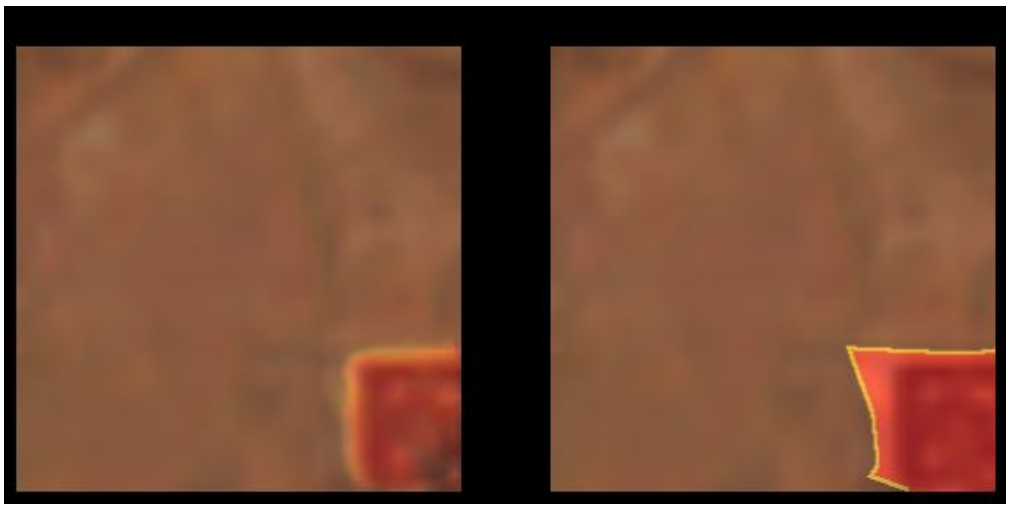


Figure 41

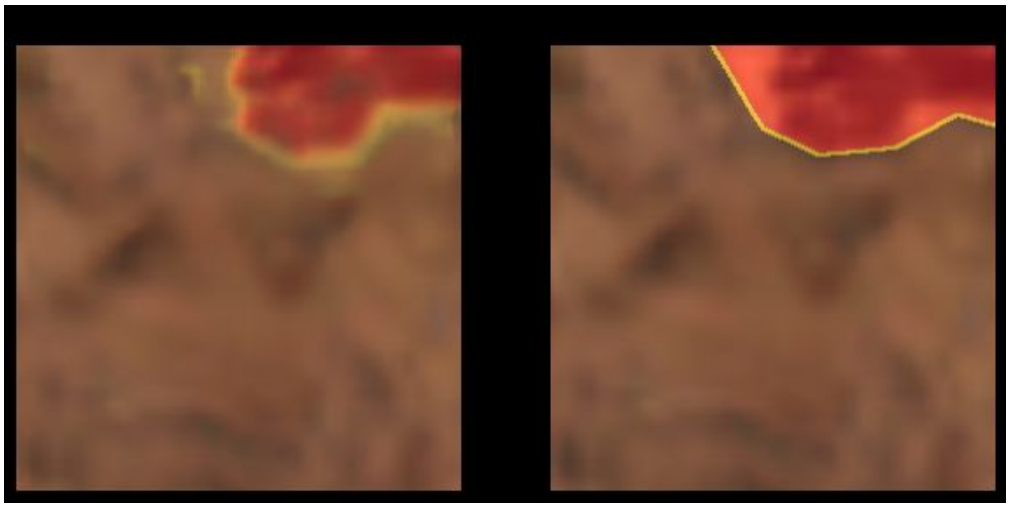


Figure 42

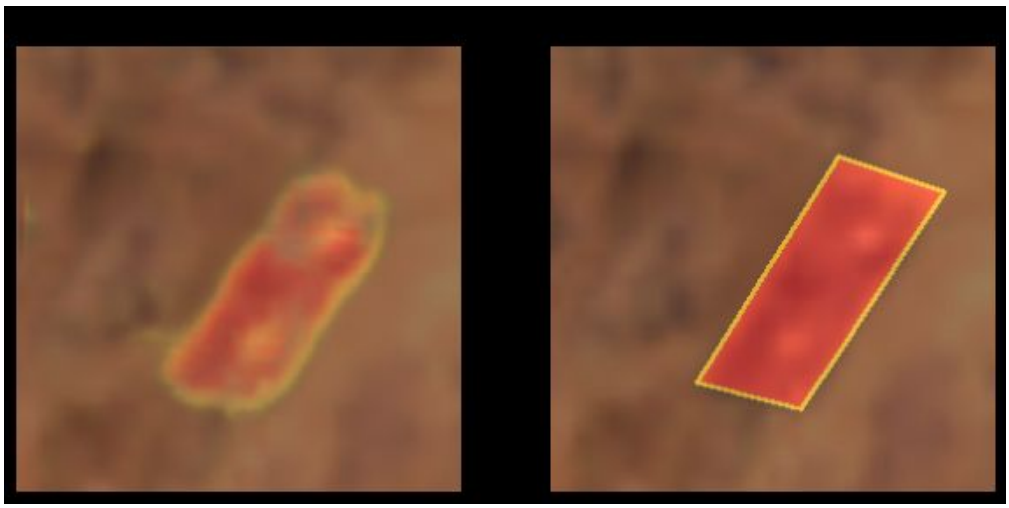


Figure 43

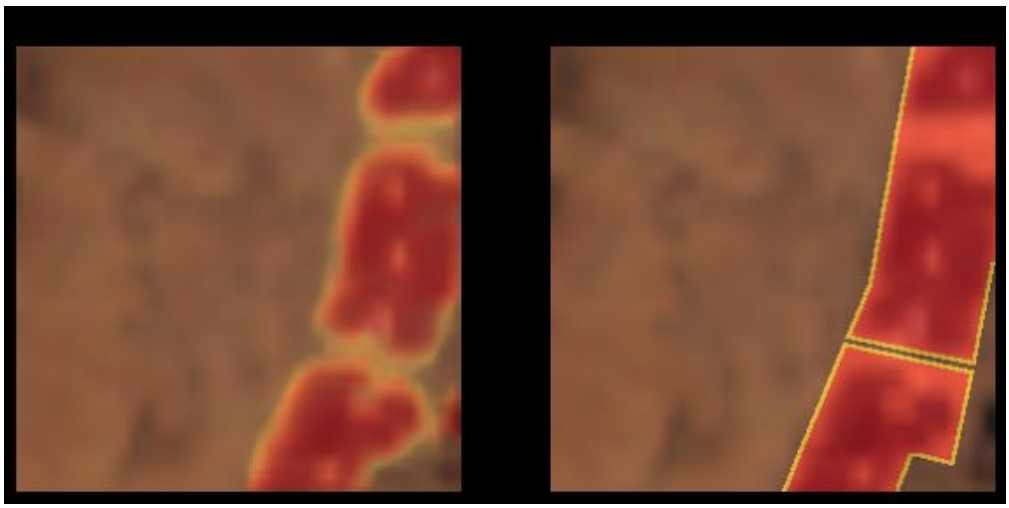


Figure 44

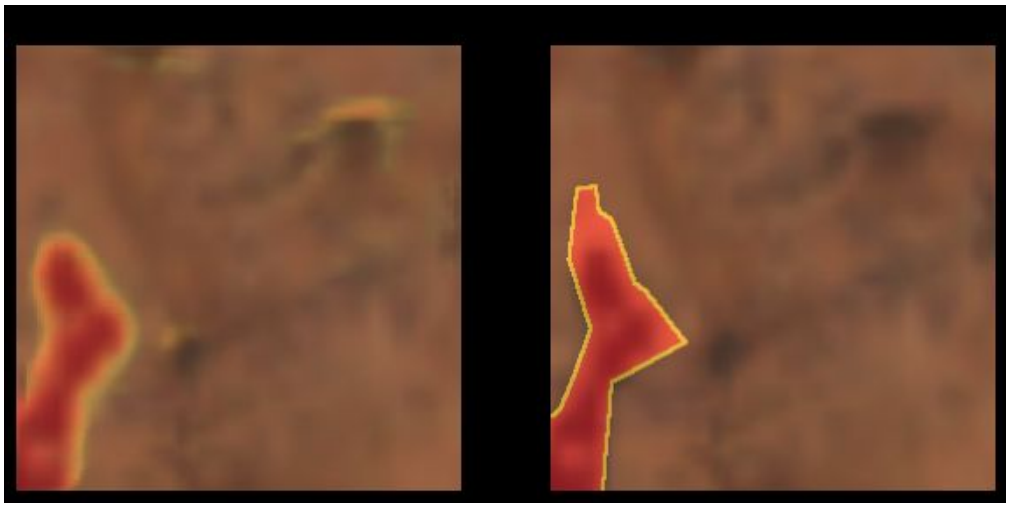


Figure 45

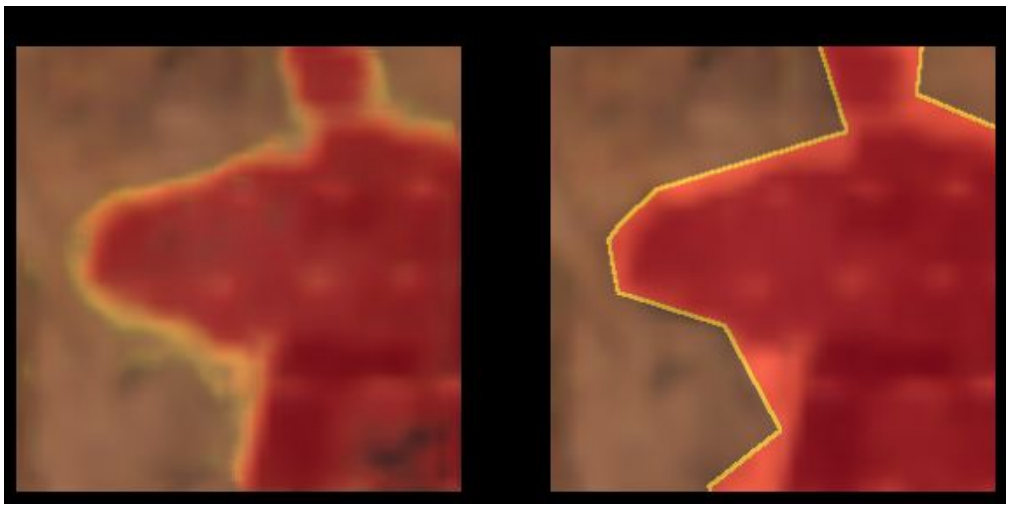


Figure 46

Training loss for the Sand Dunes dataset:

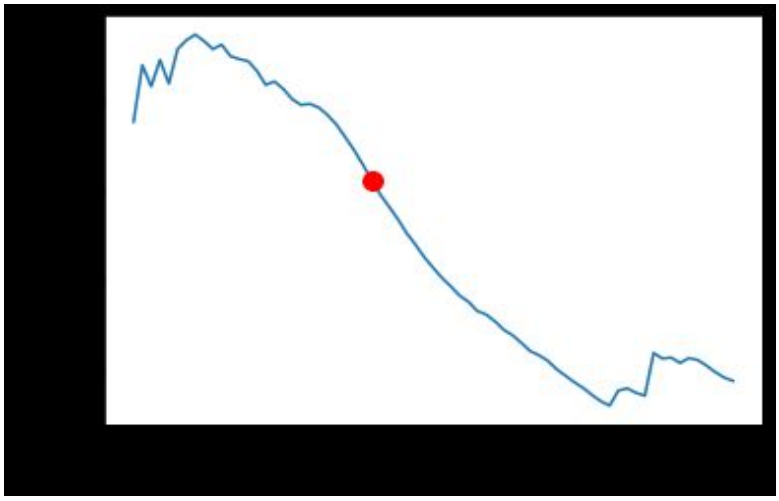


Figure 47

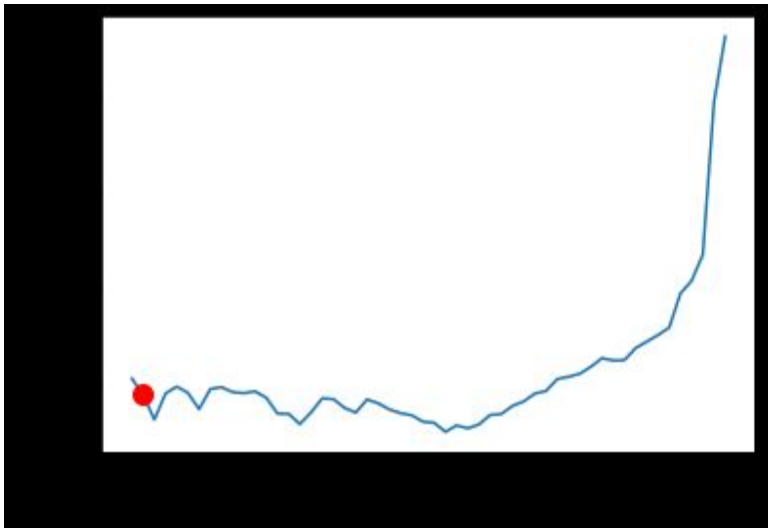


Figure 48

Results of Sand dunes Predictions using UNET:

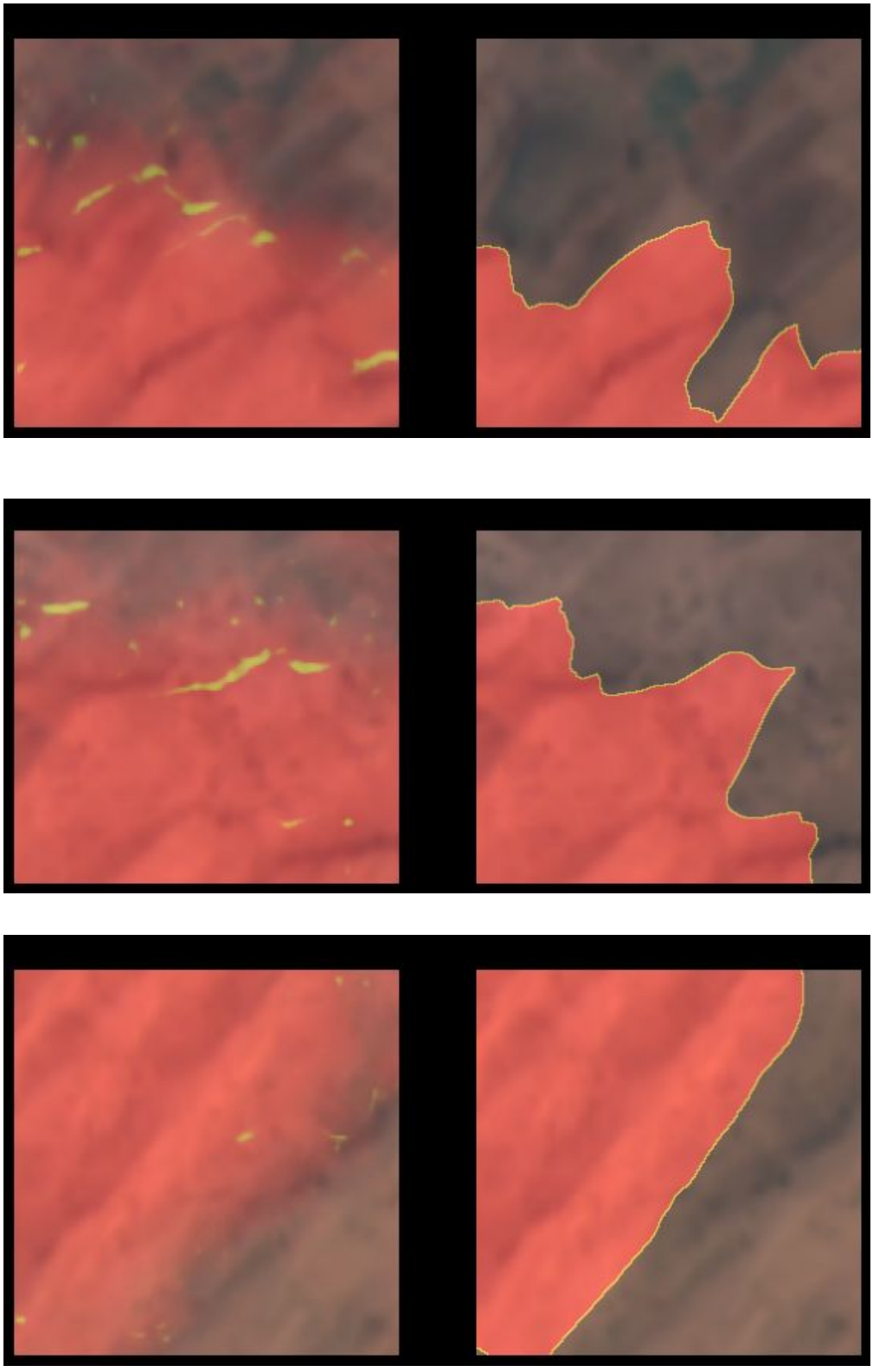
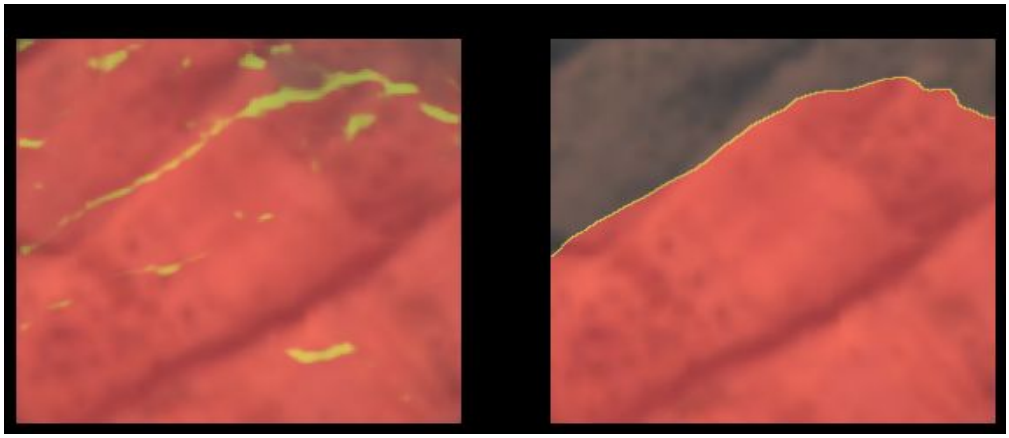


Figure 49



Training Configuration for DeepLab V3+:

Batch Size: **8 images** per batch on a single **Colab TPU**

Learning Rate for training loop: **0.0001**

Pretrained Backbone: **Xception-65**

Atrous Rates used for Spatial Pyramid Pooling: **6, 12 and 18**

Momentum: **0.95**

Weight Decay: **0.00001**

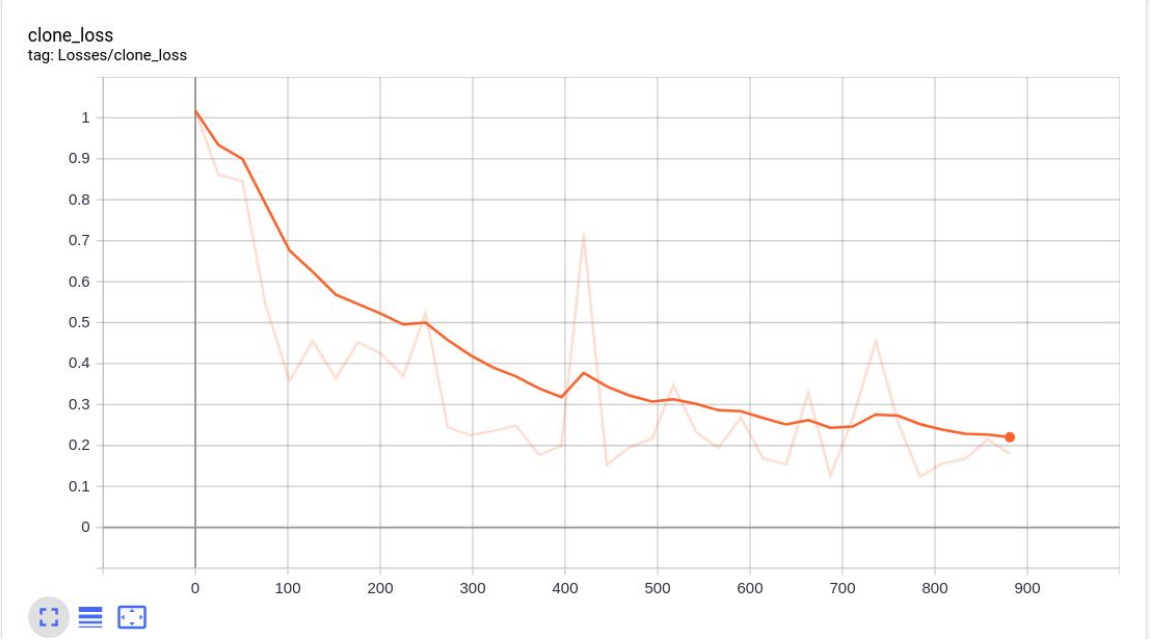
Training Crop Size: **256x256**

Number of Training Steps: **880** (Roughly **14 epochs**)

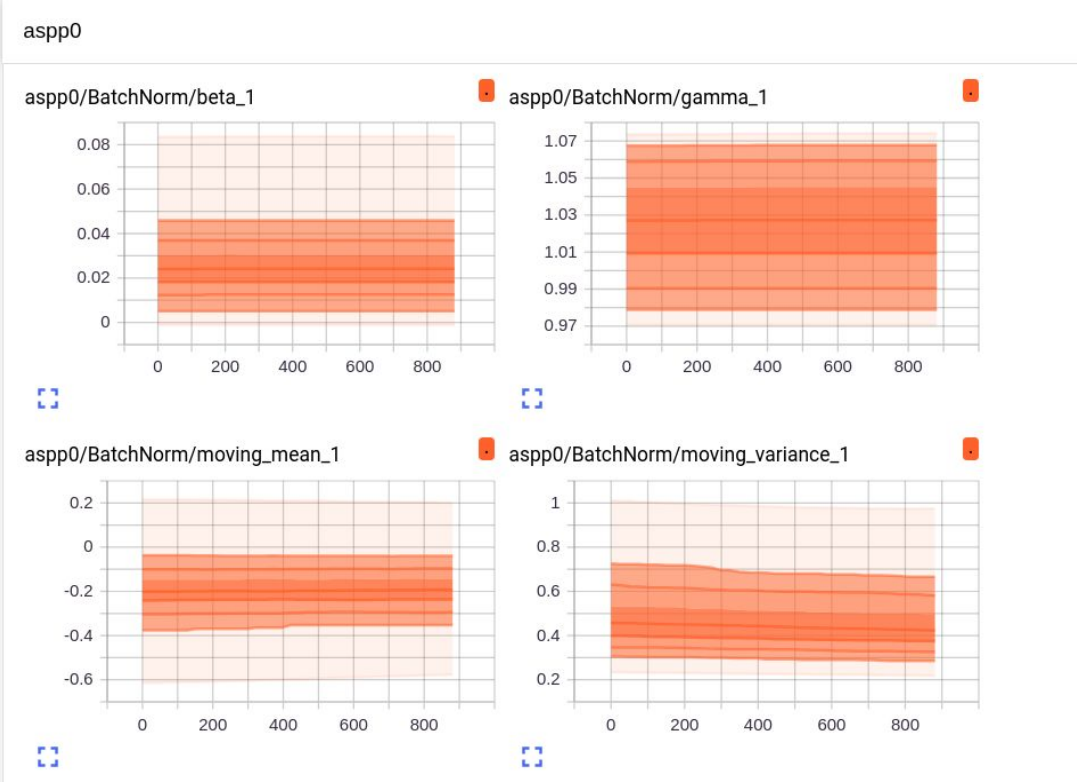
Loss Weights for Background & Solar Farm respectively: **1.0 & 4.0** respectively

(Different Loss weights were used because the images had more of background features and less of solar farm features. By increasing the loss weight for Solar Farms, we compensated for this.)

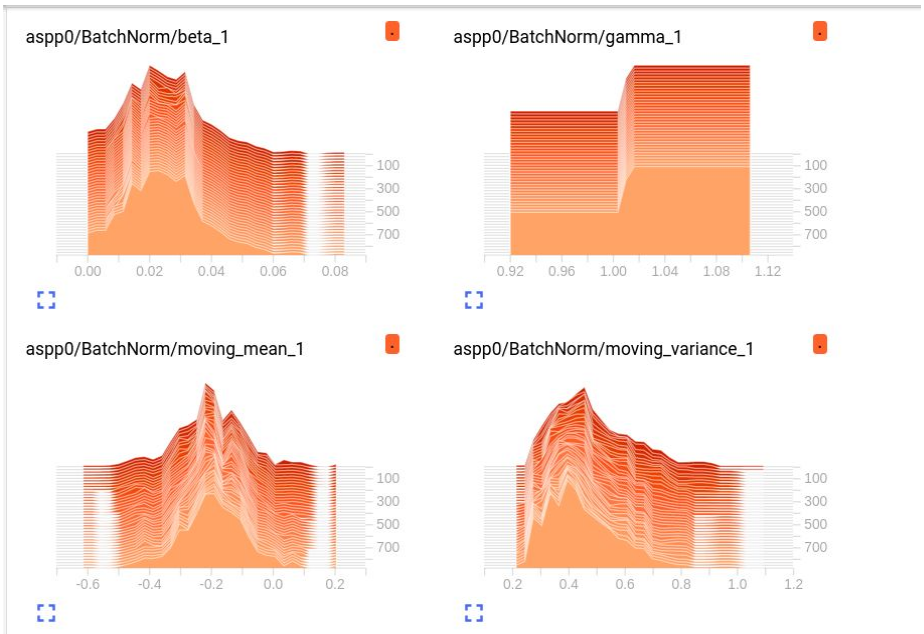
Training Loss:



Training Distributions for BatchNorm:

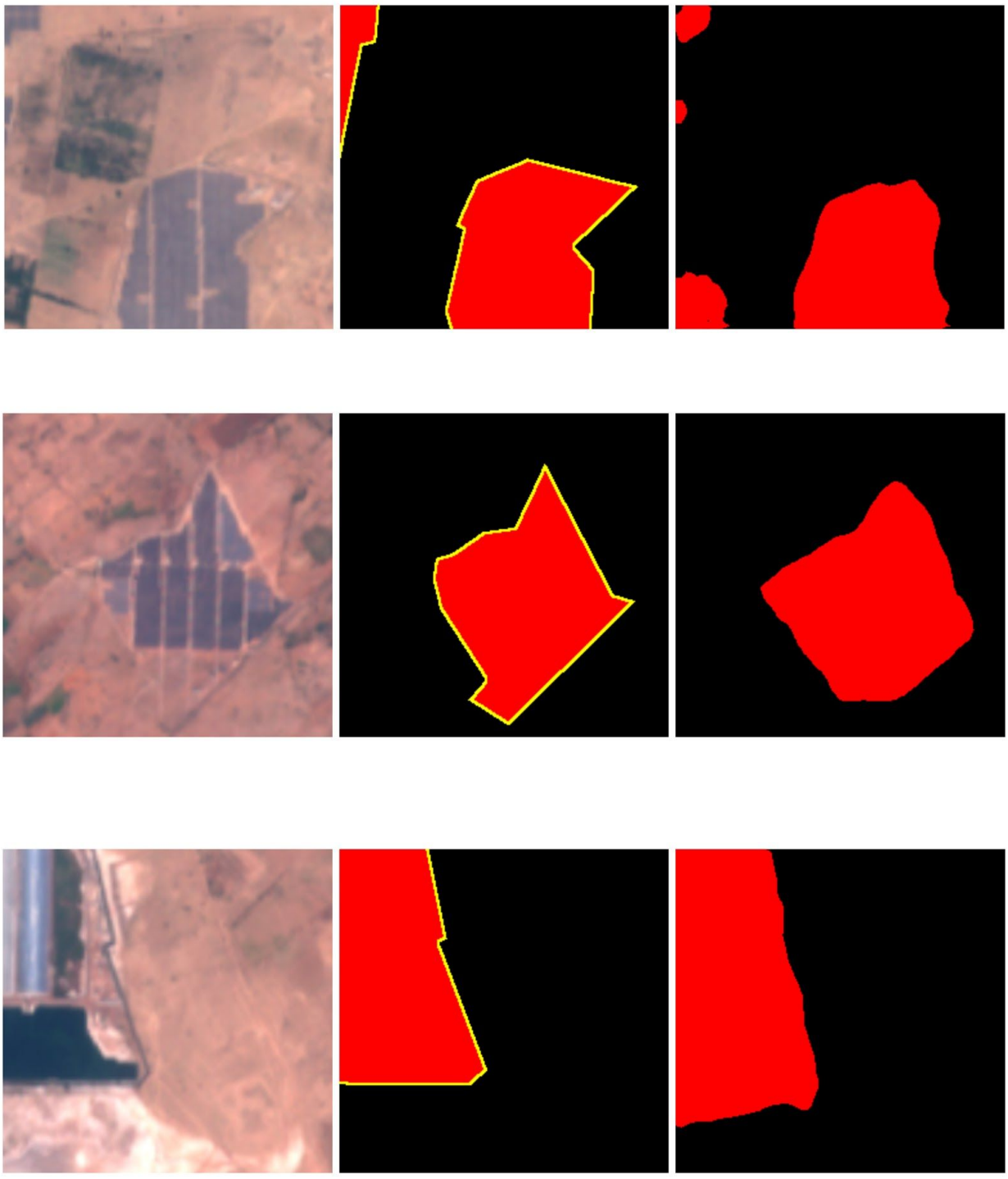


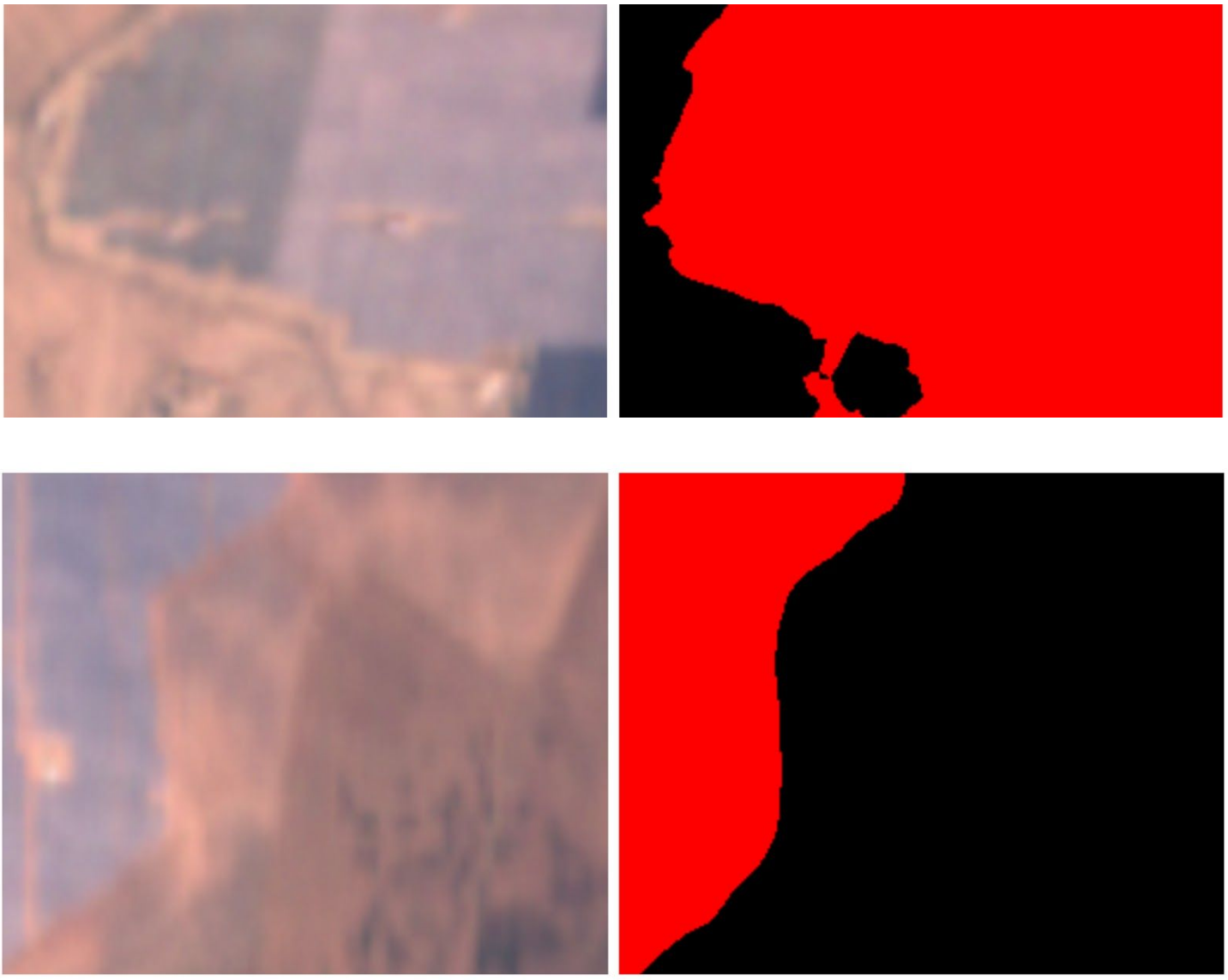
Training Histograms for the ASPP Module:



Results of Solar Farms Predictions using DeepLab V3+

After training the DeepLab model for **880 steps**, we then tested it on a variety of unseen images that were a part of the validation set. The results were quite appealing and supported the fact that the trained model started to learn the feature representations of the Solar Farms. Below are some visual representations for taking cues. The central images are ground truths and right images are predictions by the model:





These results clearly demonstrate the effectiveness of the DeepLab V3+ model for segmenting the solar farms geometries on the aerial imagery. The precise and accurate results show that this working prototype is indeed effective in proper delineation of sharp boundaries in various test cases. Through correct set of hyperparameters and training configuration, we could successfully automate the process of labelling the sand dunes and we believe that this indeed will help in reducing the arduous manual labour which is involved in the labelling process as well as provide key insights into the solar farm demography of the country.

Key Challenges Faced:

1) **Lack of Resources:** We did not have access to high grade professional GPUs (like NVIDIA) remotely. Due to this, we had to switch over to public cloud engines such as Google Earth Engine & Google Colab, which had limited GPU and RAM memory.

2) **Limited computational power:** The Colab GPUs were for general use and slow in deep computational processing. One training loop took around 6 hours to complete, thus making the process of optimization of the model parameters a very slow and difficult process.

3) **Scarce geometrical data:** The geometry data (in KML format) given to us for building the dataset was limited in quantity and belonged to very old times. Moreover, the dataset pertained only to the regions near Rajasthan. This decreased the quantity of the dataset and could not encapsulate the variations in the data.
4) **Lack of access to ISRO datasets:** We didn't have access to the high resolution imagery captured by the ISRO satellite due to remote operation. We had to rely on public image collections such as SENTINEL-2 which had a scale accuracy of 10m only.
5) **Colab Runtime Issues:** The Google Colaboratory has prescribed a runtime of 12 hours for the continuous utilisation of GPU. We were facing frequent disconnections of GPU Runtime since the training time was too long. Due to this, we had to resume the training of the model numerous times, which hampered the performance of the model to some extent.
6) **Frequent Package Issues:** We have been using various type of geo-processing data science tools and libraries for correctly processing the data. However, using all the libraries simultaneously frequently threw package errors, since some packages of a particular library were not compatible with the other. It took a lot of manual effort to resolve all the compatibility errors.

Key Learnings:

1. We gained introductory knowledge about the fields of **Machine Learning, Deep Learning, Digital Image Processing**. We learnt to perform study, research and understand popular convolutional neural network architectures such as **UNET** and **DeepLab V3+** for semantic segmentation and implement them using popular Python libraries **FastAI & TensorFLow.**
2. We learnt about **team collaboration techniques** for remote working using resources and softwares like **Google Meet, Menti.com, Google Collab, Creately**, etc.
3. We learn how to use **GitHub** platform for collaboration in a software oriented project.
4. We learnt how to extract datasets, images with the help of **Google Earth Engine**. So now we are familiar with **Geospatial data and other preprocessing** required for machine learning projects.
5. Gained knowledge of various satellites like **LANDSAT**, **Sentinel** etc. and various sensors, methodology used by them for capturing **satellite imagery and other data**.
6. Various Project management skills like preparation of effective and concise flowcharts, process maps, Gantt charts as well as efficient division of tasks along with creating a project milestone timeline to regularly map the progress of the project.
7. **The Group Discussion** and various other components that happened as a part of PS improved our communication skills as well as presentation skills. This also helped us to improve our skills related to report writing and documentation for **Software Architecture projects** as well as in general.

References

(1) Dave Luo, How to Segment Buildings on Drone Imagery with Fast.ai & Cloud-Native GeoData Tools

Url:https://medium.com/@anthropoco/how-to-segment-buildings-on-drone-imagery-with-fast-ai-cloud-native-geodata-tools-ae249612c321

(2) An introduction to Solaris

Url: https://solaris.readthedocs.io/en/latest/intro.html

(3) Satellite Image Segmentation: a Workflow with U-Net

Url: https://medium.com/vooban-ai/satellite-image-segmentation-a-workflow-with-u-net-7ff992b2a56e

(4) A Beginner's Guide to Deep Learning-based Semantic Segmentation using Keras

Url:https://divamgupta.com/image-segmentation/2019/06/06/deep-learning-semantic-segmentation-keras.html

(5) Understanding Semantic Segmentation with UNet

Url: https://towardsdatascience.com/understanding-semantic-segmentation-with-unet-6be4f42d4b47

(6) An overview of Semantic Image Segmentation

Url: https://www.jeremyjordan.me/semantic-segmentation/

APPENDIX:

Appendix - I

Applications of Semantic Segmentation in Real-World Tasks

Semantic segmentation essentially aims to classify the various objects present in an image at the pixel level. This special kind of segmentation aids in parsing the environment better, irrespective of the spatial context in which the object is present. This attribute makes semantic segmentation a prominent candidate in areas where information to be processed is scattered sparsely but over a large part since segmentation involves pixel-level accuracy. In recent times, high-resolution drone and satellite imagery have become commercially viable and freely accessible, promoting a spur in the segmentation techniques. Following are some of the important applications of semantic segmentation:

1. GeoSensing and Remote Sensing Applications:
   i) Aerial images have the capacity to span quite a large area of the land, and hence capture numerous objects simultaneously. By applying the segmentation techniques on these images, we can execute real-time surveillance of the traffic, which is an essential smart city application.
   ii) It is also very trivial to semantically segment the house and building covers on the ground. This process contributes majorly to the Government's urban planning and development programs.
   iii) Applications under this domain also include the monitoring for geographical attributes such as deforestation trends, rainfall predictions, disaster prediction and management, estimation of vegetation indices, population density, etc. The last category falls largely under ‘multi-class semantic segmentation’, due to the presence of a large number of attributes and classes associated.
   - Commonly used dataset: https://spacenetchallenge.github.io/datasets/datasetHomePage.html

2. Autonomous/Self Driving Cars (SDCs):
   SDCs involve the usage of highly precise computer vision tasks that help in the perception of a dynamic environment and trigger a response as a reaction. Semantic segmentation highly benefits the SDC’s computation systems by giving precious information about the drivable road segments (labeled with a specific colour) as well as local information about pedestrians and other vehicles’ locations. Since the

car now has access to pixel-level information of the surroundings, it will be better able to make complex decisions.

- Commonly used dataset: https://www.cityscapes-dataset.com/

3. Bio-Medical Image Processing:
   Medical images acquired through scanning procedures such as MRIs and CT Scans have invaluable information embedded into them that may often slip a radiologist's naked eye. This information may reveal various cancerous cell states at the benign stage itself. Through segmentation, we can automate the process of discovering these cells and tissues and make it a really quick process. Semantically segmenting these images would outline the location and contours of the areas of interest. Recently, deep learning methods have supported immense research work in the detection of **Corona Virus** infection using image scans.
   - Commonly used dataset: http://medicaldecathlon.com/

4. Agricultural Precision: Through semantic segmentation, we can discover those areas in the harvested land which have been infected by locusts, blights, and parasites, that render the crop useless and unfit for consumption. We can also support real-time segmentation of the health state of these crops using a drone capturing system. Moreover, segmentation also aids in the detection of unwanted plants such as weeds and fungi. Deep Learning has immense potential to reduce manual effort and reduce the overall cost of producing agricultural output.
   - Commonly used dataset: https://www.agriculture-vision.com/dataset

5. Facial Segmentation: Face segmentation is highly useful to build robust security systems that are able to uniquely identify a person's gender, age, expressions, and ethnicity.

The entire operation of 'Semantic Segmentation on Images' can be broadly divided into three sequential processes:

1) Pre-processing of the images/dataset
2) Segmentation task using various deep learning architectures and libraries. (For eg. U-net segmentation using Fast-ai and PyTorch framework)
3) Post-processing of the results obtained after semantic segmentation
4) Validating the performance of the trained model.

Appendix II - Stanford Report

We analyzed a document (http://cs229.stanford.edu/proj2017/final-reports/5243715.pdf), as well as the model which was used in this document, was extremely similar to the one that we will be implementing in the project. Below we have written an analysis of the document.

Introduction-

1. The topic of this project was to Automatically detect buildings from satellite images of high resolution using the U-Net model.
2. Detecting buildings has a lot of potential applications such as evaluating the available surfaces in order to implant solar panels on the roof.
3. Other applications include the assessment of building damages that occur due to natural disasters. This assessment can help the formulation of adequate responses in the targeted areas.
4. Being able to detect new buildings directly from satellite images will also be extremely useful in regions where populations move very rapidly (because they are nomadic, displaced, or migrating).
5. What exactly is being done here is to apply semantic segmentation of satellite images by classifying each and every pixel of the satellite image as either belonging to a building or not.
6. In this project, they had developed a Convolutional Neural Network suitable for Image Segmentation and this Neural Network was inspired by the U-Net.
7. The U-Net is a specific type of FCN. FCN refers to Fully Convolutional Networks. U-Net is mostly used for the segmentation of biomedical images using a reduced data-set, but it has also been proven to be efficient for the pixel-wise classification of satellite images.

Working with the dataset/Preprocessing

1. The project followed the advice of Drew Bollinger and they collected their data from OpenStreetMap(OSM). OSM is an open-source platform for editing maps.
2. As this particular project was mainly focussed on the detection of buildings only the building layers of the part of the maps that they required were downloaded. Through this download of labeled data, 34547 RGB images of size {256 256 3} from the territory of France, along with their corresponding RGB building mask were acquired.

3. For preprocessing, the images that were acquired from the satellite were mapped along with their corresponding masks. And the sizes of the labeled maps were altered as only buildings were being detected.
4. In order to train the model on a pre-built network in the project, they first had to resize the images and label masks to a 224 224 3 size. Also as the labels that were extracted consisted only of the buildings, there was no actual need for a complete RGB channel. Thus the label images were converted to a binary mask of size 224 224 1 with 1 for pixels labeled as ”buildings”, 0 otherwise. An example of one of the images and it’s a corresponding mask(obtained from the labeled data) -

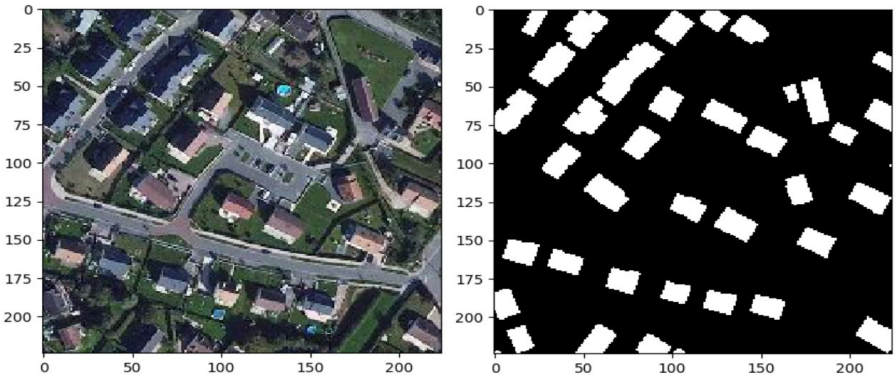


Figure 16

5. As the dataset had been labeled by humans it led to resulting in quite a high accuracy. But it also caused a problem as some of the data was inconsistent and had to be rectified before using.
6. The threshold of the minimum ratio of building pixels per image was kept to be 15%(as without this most of the buildings were not being detected). This led to the end up with 3042 images in the dataset. The images in the dataset were divided into train/dev/test in 75:15:15 ratio. This led to there being 2129, 456, and 456 images in the train, dev, and test respectively. But clearly, the dataset that was acquired was quite small.
7. Due to the small dataset, data augmentation was applied due to certain reasons. After this the images were centered, normalized, and finally shuffled in the dataset.
8. The U-Net model worked well with a limited number of training examples only if data augmentation was applied heavily.
9. Accordingly, in order to maintain a reasonable amount of images, and above all to avoid overfitting by ensuring a sufficient invariance and robustness of the network real-time data augmentation techniques were applied to the training set.
10. Finally, for the data processing, the satellite images were shifted, flipped, and rotated, which led to being able to train the model on a considerably larger set of images. This task was done using the Keras framework which augmented the data in real-time when feeding the network with batches.

Architecture of the model

1. The model that was used for training was the U-net. The U-net was originally developed for biomedical image segmentation, but it has been proven to work astonishingly well for satellite images of high resolution as well.
2. The U-net basically builds upon the Fully Convolutional Network.
3. The network was able to output a pixel-wise binary classification of the building and not a building pretty well.
4. In the U-net, we observe a contracting path that extracts the features of various levels through a sequence of convolutions, ReLU activations, and max poolings. This leads to the capture of the context of each pixel properly.
5. After the contracting path of the U-net, we observe a symmetric expanding path which upsamples the result to increase the resolution of the detected features. In the U-net architecture, skip-connections or contractions are added in between the contracting path and the expanding path, which allows precise localization as well as context.
6. The expanding path thus consists of a sequence of up-convolutions and concatenations with the corresponding feature map from the contracting path, followed by ReLU activations. The number of features in each of the levels of downsampling is doubled. The figure that corresponds to the U-net that was referred to is -

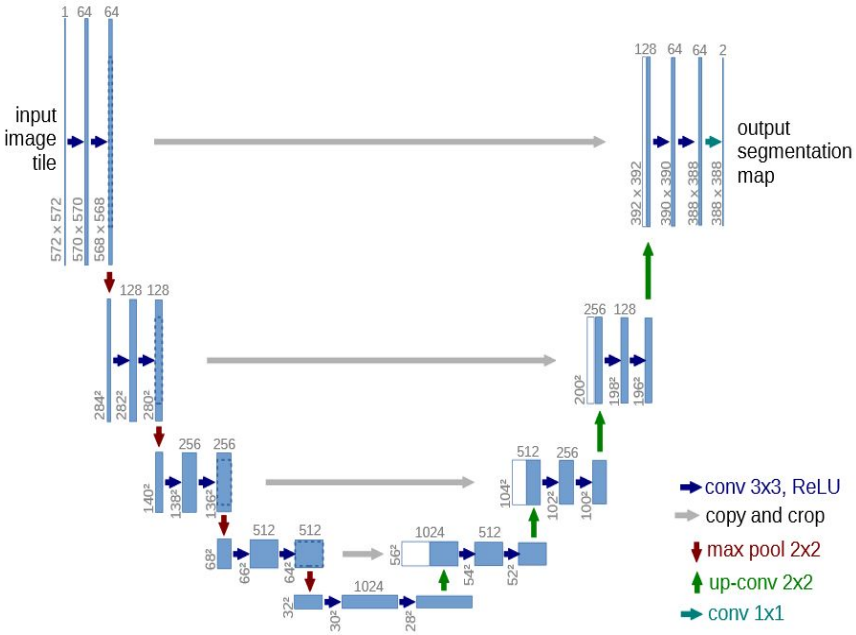


Figure 17

Modifications in the Model of UNET

1. The stochastic gradient descent is replaced with the Adam Optimizer. As Adam optimizer tends to converge faster during training.
2. The dimensions of input images were changed as per the U-net model to 572 × 572 ×3.
3. The padding was changed to 'same' in order to prevent shrinking of the image matrices when implementing convolutions, added batch normalization after each ReLU activation to speed-up training, and used a loss based on the Dice coefficient instead of the cross-entropy loss.
4. Finally, in order to ease the optimization and tackle vanishing gradients, the final downsampling layer of 1024 depth was removed.

Important Formulas Used

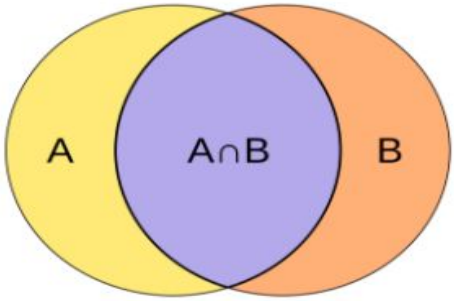


Figure 18

Following formulas were used for the loss calculations:

$$Dice = 2 \times \frac{|A \cap B|}{|A| + |B|} = 2 \times \frac{precision \times recall}{precision + recall}$$ - **Equation (1)**

$$Jaccard = \frac{|A \cap B|}{|A \cup B|}$$ - **Equation (2)**

Training Process

1. The GPU that was used for training was Floydhub (Tesla K80) and the batch size kept was 32.
2. The Keras library has a callback named ReduceLROnPlateau that was used. This callback helped in reducing the learning rate by 50% after a certain number of epochs.

Testing and result

1) In the project, we see that after we evaluate the trained model on the test set we obtain a Dice coefficient of 0.75, and a Jaccard coefficient of 0.60.
2) One of the important observations that were observed was that the score that was obtained turned out to be highly dependent on the resolution of the images that were used.
3) Due to data augmentation (which we had used in order to improve the working of the U-Net model), as we converge more towards a Dice coefficient of approximately 0.75 for both the training set and the development set.

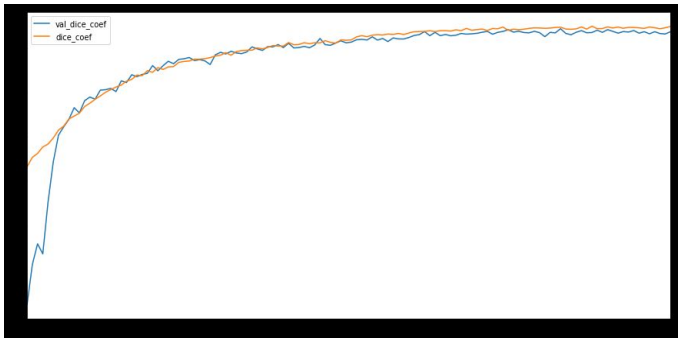


Figure 19

Some of the image predictions are:

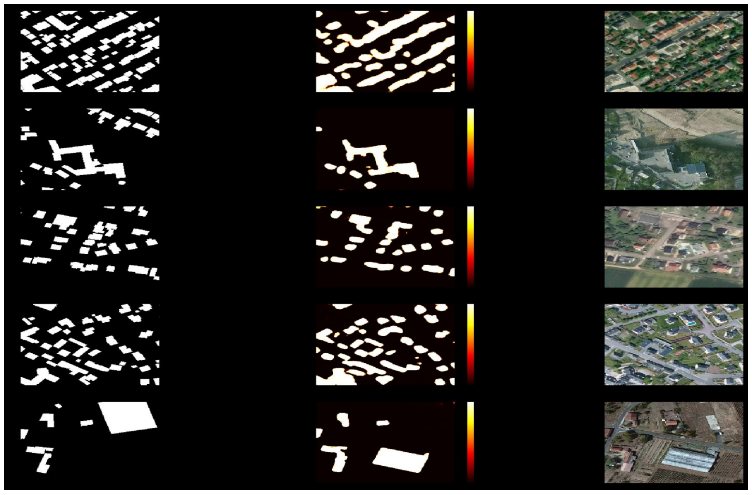


Figure 20

Challenges encountered in training

1. The major loss came from the images in which the ratio of building pixels per image was very low i.e the image in which most areas were open grounds etc. It was concluded that the networks failed to detect mislabeled or blurry data.
2. In many cases, either the masks were wrongly defined by the users of OSM or the satellite images were not up to date. So for improvement, they tried to improve boundary detection in the images.
3. The model had a tendency to give soft edges instead of sharp as well as defined ones. This is especially a major issue in densely populated areas where buildings were so close to each other.

Method used for improving boundaries-

1. The problem of boundary detection was tackled by using the weights of the inner and outer borders of the buildings. Looking at the formula of the Dice coefficient and its value and by weighting negatively the pixel on the outer boundaries of the building instead of just no value or zero, it will reduce the loss to a great extent if the model outputs a non-zero probability on these pixels.
2. On the other side, if we want the model to accurately classify the inner boundaries of a building, we can just weigh positively the inner border (if the model outputs a high probability for this pixel, the loss will decrease).
3. The modifications especially the last one allows us to compensate for some of the pixels weighted negatively in the denominator of the fraction. After this modification we got the Dice coefficient to be 0.74 and the Jaccard coefficient to be 0.59.

Appendix III - More about UNet

UNET is an architecture which uses a Fully Convolutional Network Model for the semantic segmentation task. In order to understand the UNET architecture, there are a few things that we will need to understand first. First, we need to understand the images which are present in our dataset. After this, we will need to understand the basic building blocks of the UNET architecture.

Understanding the training set :

An example of one of the satellite image and the corresponding image that we are using in the dataset is as follows-

The image to the left is the satellite image. And the image to the right is the mask which basically is the ground truth label and shows the building footprint, building boundary and the contact areas through 3 channels(this was done by using Solaris and keeping 3 RGB channels such that the 1st channel represents the building footprints, the 2nd channel represents building boundaries, whereas the 3rd channel represents the contact areas between the adjacent geometries.

Before the explanation of the UNET architecture we will first take a brief look at the basic building blocks of it which include convolution operation, max pooling and upsampling using transposed Convolution.

Understanding Convolution, Max Pooling and Transposed Convolution:

Convolution operation:

In the convolution operation, we have 2 inputs which are namely-

1. A 3D volume (the input image) of volume/size = $n \times n \times channels$. The channels basically refer to the RGB channels. (n refers to the number of input features)
2. A set of k filters (also called and kernels) where each one of the filter(kernel) is of size/volume= $f \times f \times channels$. f is usually odd and is typically equal to 3 or 5.

The output that we get from a convolution operation is also a 3D volume and the output that we get is of size= $nout \times nout \times f$. Here the value of $nout$ is given by-

$$nout = \lfloor \frac{n+2p-f}{s} \rfloor + 1$$

Where-

n: n refers to the number of input features

$nout$: nout refers to the number of output features

f : refers to the size of the convolution kernel

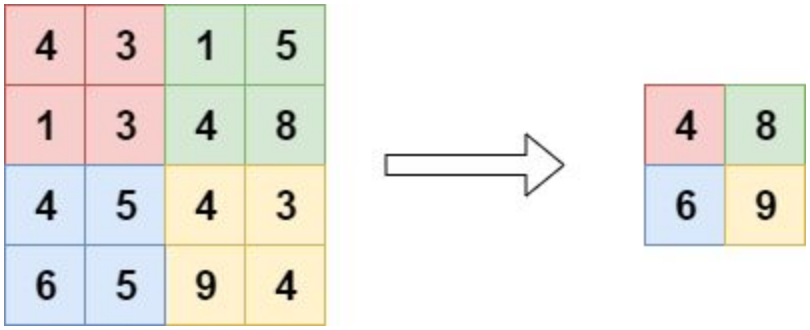
s : refers to the stride size

p : refers to the padding size

Max Pooling Operation:

In max pooling, we are basically reducing the size of the feature map so that we will have fewer parameters in the network

What is done here is from every 2x2 block of the input feature map, the maximum pixel is selected and thus the pooled feature map is obtained. We choose the maximum pixel in order to retain the important features from each region and not use the other information which is not as important as the maximum pixel.



It is easily noticeable that after both the convolution operation and the max pooling operation, the size of the image is reduced. Due to this reduction this process is called down sampling. If the size of an image before max pooling is 4x4, it gets reduced to 2x2 after max pooling. Down sampling basically refers to the conversion of a high resolution image to a low resolution image. Intuitively we can say that via the process of down sampling, the model is better able to understand what exactly is present in the image, but in this process it loses the information of where it is present. But in image segmentation it is not only important to know what is present in the image but it is extremely important to know where exactly the part detected is present.

So in order to get the information of where what we want is present, we need up sampling. In a normal classification problem, we would not require up sampling as in classification we just need to know if what we require is present or not. But here in segmentation, we need to be able to mark every pixel and thus we need to up sample.

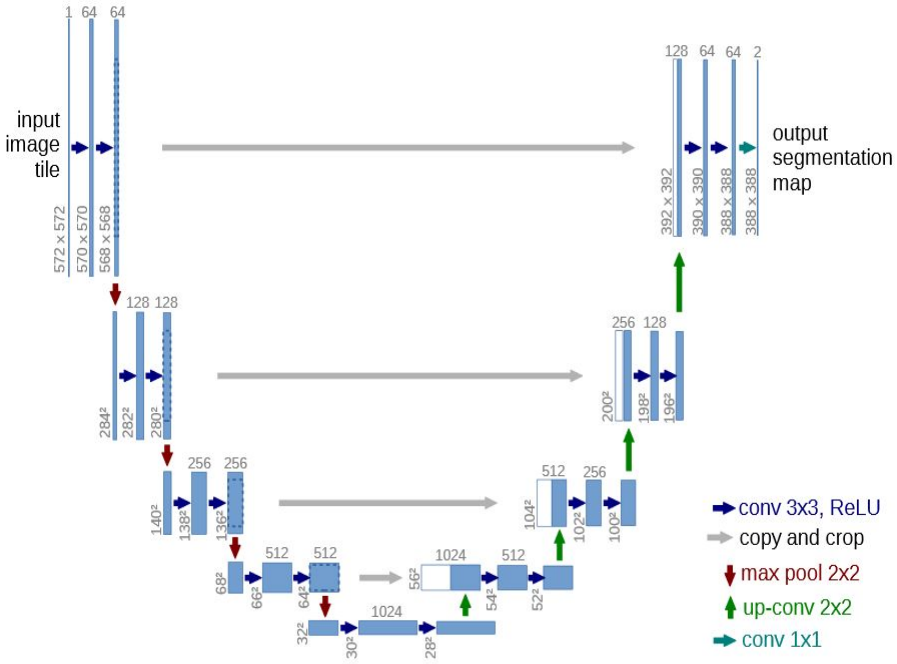
Up sampling:

As stated above we need to know about the where information as well, hence we need to up sample the image. After the convolution operations and the max pooling operations, the high-resolution images get converted into low-resolution images. So we now need to convert the low images back to a high-resolution image in order to recover information.

Transposed convolution can be used for upsampling images in order to find the original representation of a convolutional filter map. Transposed convolution is a technique to perform up sampling of an image. It is like the exact opposite of the process of a normal convolution operation.

The UNET architecture:

The UNET architecture was originally developed for Bio Medical IMage Segmentation but it has been seen to work quite well with the semantic segmentation of satellite images as well. The Unet architecture consists of 2 paths namely the contraction path and a symmetric expanding path. These 2 paths when shown together form a 'U' shape which gives UNET it's name.



A. Contracting path:

The contraction path which is also called as the encoder is simply a stack of convolutional and max pooling layers.

B. Expanding path:

The expansive path is constituted by transposed 2d convolutional layers(basically referring to up sampling). While the downsampling operation involves convolution and max-pooling operation, the

upsampling operation makes use of transposed convolution and concatenation followed by convolution to get the output. This is done so that the feature maps are of the same size as from the encoder network since the output of segmentation is the same as the input image.