MAP SEGMENTATION

A Project Report

Submitted by:

Krupali Mewada (1501056)

Parshwa Shah (1641068)

in partial fulfilment for the award of the degree

of

BACHELOR OF TECHNOLOGY

IN

INFORMATION AND COMMUNICATION TECHNOLOGY (ICT)

at

Ahmedabad University

School of Engineering and Applied Sciences (SEAS)

Ahmedabad, Gujarat

May'2020

DECLARATION

I hereby declare that the project entitled " **MAP SEGMENTATION** " submitted for the B. Tech. (ICT) degree is my original work and the project has not formed the basis for the award of any other degree, diploma, fellowship or any other similar titles.

Krupali Mewada (1501056)
Parshwa Shah (1641068)

Place: SEAS, Ahmedabad University
Date: 6'May'2020

CERTIFICATE

This is to certify that the project titled " **MAP SEGMENTATION** " is the bonafide work carried out by **Krupali Mewada & Parshwa Shah**, students of B Tech (ICT) of School of Engineering and Applied Sciences at Ahmedabad University during the academic year 2019-2020, in partial fulfilment of the requirements for the award of the degree of Bachelor of Technology (Information and Communication Technology) and that the project has not formed the basis for the award previously of any other degree, diploma, fellowship or any other similar title.

Dr Kuntal Patel

Place: SEAS, Ahmedabad University

Date: 6'May'2020

TABLE OF CONTENTS

ACKNOWLEDGEMENT

I would like to communicate my most profound thankfulness to every one of the individuals who gave me the likelihood to finish this report. An exceptional appreciation I provide for our task guide - Dr Kuntal Patel, Mr Vishal Patel, Mr Sidhdharth Patel & Mr Saurabh Bhabhor (BISAG), whose commitment in animating proposals and encouragement, helped me to facilitate my project, invested endeavours in managing the team and gave us the vision to make this stage. Also, grateful for the direction given by all other supervisors too. An uncommon amount of gratitude goes to my group, who helped me throughout the time, who helped me all through the venture by giving their significant commitment as thoughts, structure arranging and usage.

Last yet not the least, I hold in high regard the specialized direction given by Dr Kuntal Patel just as the boards particularly in our task demonstration that has improved our presentation aptitudes on account of their valuable feedback.

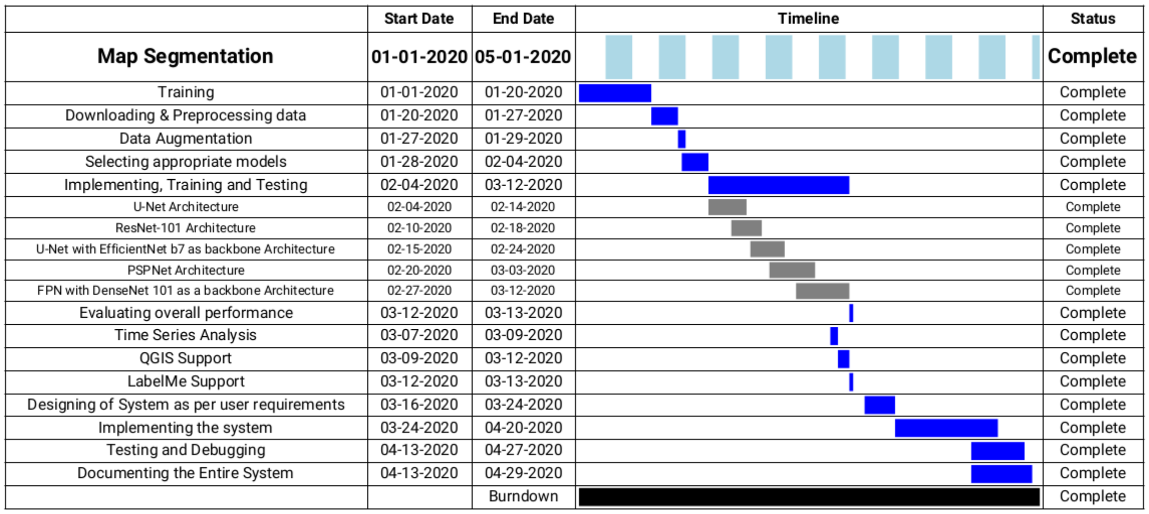
ABSTRACT

Everyday Geospatial Data Storages are deluged with millions of optical overhead imagery captured from airborne or space-borne platforms. Manual data interpretation on such a large amount of data becomes an intractable task, hence machine vision techniques must be employed if we want to make any use of the available data. In this project, we have made an application which would deal with semantic segmentation of high-resolution (aerial) images where a semantic class label is assigned to each pixel via supervised classification. Deep learning techniques have shown impressive performance, particularly for image processing. However, a major drawback of using deep learning techniques is that they are extremely data-hungry and using them leads us in exacerbating the limitations of supervised learning, to get enough annotated training data. But on the bright side, these techniques are immune to noise so instead of taking and annotating large datasets we can use publicly available datasets. Such publicly available datasets might contain errors but in return, a very high amount of data is available to us on which we can train our model. As deep learning models would require high computational power and in order to maintain high scalability we have tried to exploit the benefits of Cloud Computing and RESTful services (Representational state transfer). We have successfully built a REST API and deployed it on the institute's server. Our application segments each map image into various segments through which we can perform time series analysis on images and track development in various parts of our country. Moreover, our time series analysis would help us in tracking various environmental changes such as deforestation, afforestation, urbanization, etc and infrastructural changes such as Rural development. Also, we would provide additional support files to load all the detected classes as the layers in QGIS and improve segmentation parts through LabelMe.

LIST OF FIGURES

LIST OF TABLES

TIME-LINE



	Start Date	End Date	Timeline	Status
Map Segmentation	**01-01-2020**	**05-01-2020**		**Complete**
Training	01-01-2020	01-20-2020		Complete
Downloading & Preprocessing data	01-20-2020	01-27-2020		Complete
Data Augmentation	01-27-2020	01-29-2020		Complete
Selecting appropriate models	01-28-2020	02-04-2020		Complete
Implementing, Training and Testing	02-04-2020	03-12-2020		Complete
U-Net Architecture	02-04-2020	02-14-2020		Complete
ResNet-101 Architecture	02-10-2020	02-18-2020		Complete
U-Net with EfficientNet b7 as backbone Architecture	02-15-2020	02-24-2020		Complete
PSPNet Architecture	02-20-2020	03-03-2020		Complete
FPN with DenseNet 101 as a backbone Architecture	02-27-2020	03-12-2020		Complete
Evaluating overall performance	03-12-2020	03-13-2020		Complete
Time Series Analysis	03-07-2020	03-09-2020		Complete
QGIS Support	03-09-2020	03-12-2020		Complete
LabelMe Support	03-12-2020	03-13-2020		Complete
Designing of System as per user requirements	03-16-2020	03-24-2020		Complete
Implementing the system	03-24-2020	04-20-2020		Complete
Testing and Debugging	04-13-2020	04-27-2020		Complete
Documenting the Entire System	04-13-2020	04-29-2020		Complete
		Burndown		Complete

CHAPTER 1: INTRODUCTION

1.1 Problem Statement

Manually annotating each strand of a satellite image is of Aerial Images is extremely time-consuming. Here, we follow the idea to produce polygonal annotations of objects interactively using humans-in-the-loop and also perform time series analysis on aerial images. This analysis would help us in observing the amount of man-made phenomenons such as infrastructural development of rural and urban places and some natural phenomenon such as deforestation, decrease in ice cover etc over a particular area in a particular amount of time. We had also provided support features on exporting our project data in some standard formats which can be easily used by industries.

1.2 Project overview

This project can be divided into five main modules as listed below:

1.2.1 Map Segmentation into various classes: This module takes aerial images as input and segments the image into various predefined classes such as Roads, Buildings, Vegetation etc. For segmentation of these aerial images into various classes we use several deep learning models such as U-Net, Resnet 101, FPN etc and the user can use a model of his/her choice.

1.2.2 QGIS support: This module takes a segmented image as an input and returns various files through which users can add vector layers of different segmented classes in various GIS softwares. Mainly for QGIS as it is the tool used by the institute but we have also verified it's functionalities for ArcGIS.

1.2.3 Development Tracker: This module performs a time series analysis on a set of segmented images. For eg: we will be providing two images one which is the present image of a particular area and another will be the image of the same area but 10 years back, by doing so we will easily track down how much growth has been done in that particular area in terms of infrastructural development, deforestation, etc.

1.2.4 LabelMe support: LabelMe is also an open-source available tool which helps people annotate data manually. This support helps the user in correcting some errors that might occasionally occur results generated by our deep learning techniques. It saves a lot of time as the user doesn't have to label the entire image instead he/she just has to make amendments.

1.2.5 REST API: It is a REST-based framework made with the help of Django REST API and it implements all the above-mentioned modules as it's one of its services. The entire application is made scalable to multiple users and one can access all the services through simple REST API calls.

1.3 Hardware Requirements

1.3.1 Nvidia Tesla T4: GPU was used for training of various deep learning models in order to perform semantic image segmentation on aerial images. CUDA Toolkit 10.1 was used in order to perform parallel computing.

1.4 Software Requirements

1.4.1 Tensorflow,Version: 1.14.0

TensorFlow provides excellent functionalities and services when compared to other popular deep learning frameworks. These high-level operations are essential for carrying out complex parallel computations and for building advanced neural network models. TensorFlow is a low-level library which provides more flexibility.[1]

1.4.2 Keras, Version: 2.2.5

KERAS is an Open Source Neural Network library written in Python that runs on top of Theano or Tensorflow. It is designed to be modular, fast and easy to use. ... Keras doesn't handle Low-Level API such as making the computational graph, making tensors or other variables because it has been handled by the "backend" engine.[2]

1.4.3 Opencv-python and Opencv-python-headless, Version: 4.1.2.30

OpenCV-Python is a library of Python bindings designed to solve computer vision problems. OpenCV-Python makes use of Numpy, which is a highly optimized library for numerical operations with a MATLAB-style syntax. All the OpenCV array structures are converted to and from Numpy arrays.[3]

1.4.4 Django, Version: 3.0.4

Django is an open-source python web framework used for rapid development, pragmatic, maintainable, clean design, and secures websites. It takes care of a lot of hassle involved in web development; enables users to focus on developing components needed for their application.[4]

1.4.5 Argon2 and Argon2-cffi, Version: 0.1.10 and 19.2.0

Argon2 is a key derivation function that was selected as the winner of the Password Hashing Competition in July 2015. It was designed by Alex Biryukov, Daniel Dinu, and Dmitry Khovratovich from the University of Luxembourg.[5] Argon2 and Argon-cffi is an implementation of argon key derivation function in python.

1.4.6 Pillow, Version: 7.0.0

Python Imaging Library is a free library for the Python programming language that adds support for opening, manipulating, and saving many different image file formats.[6]

CHAPTER 2: LITERATURE SURVEY

2.1 Existing System

2.1.1 Polygon-RNN++: Polygon-RNN++ is a tool used to produce polygon annotations for semantic segmentation of images using PolygonRNN. Also they inserted some additional features such as CNN encoder architecture, improvised the process of training the model with the help of Reinforcement Learning etc. They have used Cityscapes dataset for training purpose which would easily outperform the original model.[7]

2.1.2 RoadTracer: As mapping road networks is both labour-intensive and expensive RoadTracer was introduced. RoadTracer is a new method used in order to automate the process of constructing road network maps from aerial images.[8]

2.2 Proposed System

As discussed earlier, a major throwback with the existing system is that most of them are designed to run on images captured from within the cities, but when performed on an aerial image it fails to detect the classes as it has been trained on such type of dataset. Our project encapsulates a workflow for using deep learning to understand and analyze geospatial imagery. Our work was on performing semantic segmentation on aerial imagery provided by an ISPRS and Massachusetts Roads and Building dataset.

Also, with our proposed system a lot of time and energy can be saved as all this work is done by manually annotating all the images. However, our system not only helps in tracking different classes but also helps in performing analysis of various parts of it and all on the top of a Django server on which all functionalities can be accessed remotely.

2.3 Feasibility Study

Our project is focused on detecting and classifying various classes, as there is a great amount of diversity present in each category. It is a bit difficult for anyone to gather data which includes such diversity. However we can make our system more robust by further training it on erroneous samples.

For speeding up our system we are currently using some light models such as UNet but if we want to improve our results further more than we have to make use of more deeper networks. As these deeper networks require heavy computation we would require better hardware than we are currently using.

CHAPTER 3: TRAINING PROCESS

3.1 Data Extraction and Preprocessing

3.1.1 Data Selection: For this project, we will be using open source data sets such as Massachusetts Roads and Building Dataset and ISPRS.
Massachusetts Roads and Building Dataset contained 1171 aerial images, along with their respective maps. All the images are of 1500 x 1500 in dimension and are in .tiff format. While ISPRS Dataset contained 40 aerial images, along with their respective maps. They are 6000 x 6000 in dimension encoded into 32-bit values and are in .tif format.

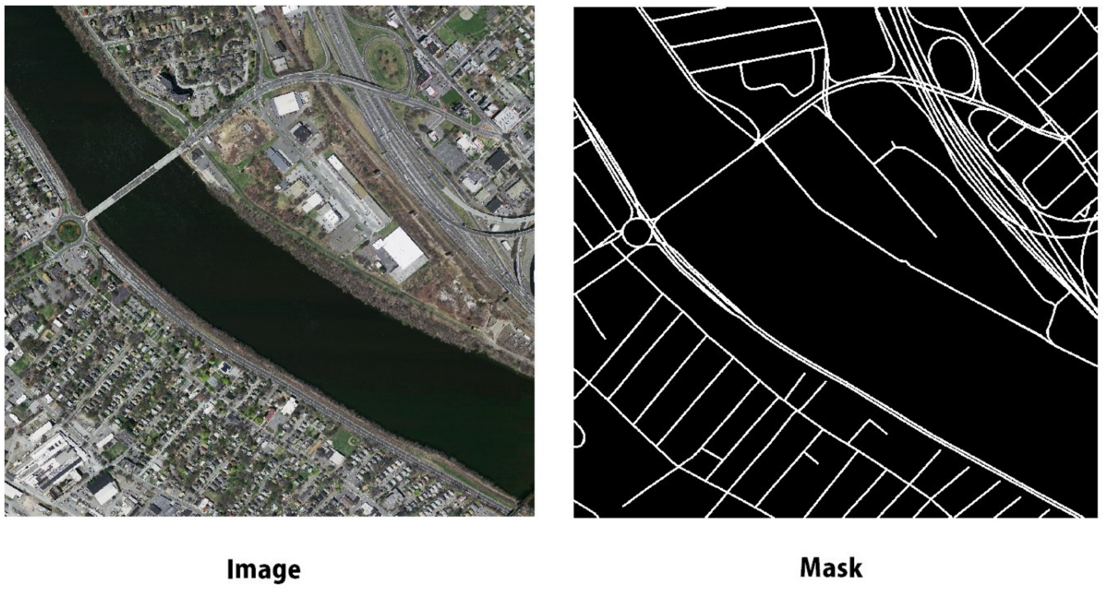


Figure 1: A sample of how our data set looks like [9]

3.1.2 Data Preprocessing: Here our motive was to preprocess data in order to increase the efficiency of our model.

Massachusetts Roads and Building Dataset Preprocessing

The major steps of preprocessing of Massachusetts Roads and Building Dataset is listed below:

1. **.tiff to .pnf:-** First of all, we converted all the .tiff images to .png images with three colour bands (RGB) and with a resolution of 1.2m.
2. **Hand-picking:-** There were few images present in the dataset where a chunk of the aerial images was missing. As it can affect our model's performance, we had to remove them manually.
3. **Cropping instead of resizing:-** Training our model on large images was not only resource-intensive but also time-consuming. Resizing images to lower dimensions was the answer to our problem. So in order to avoid losing information, we had resized all the images without interpolating them.
4. **Augmentation:-** We had augmented our data (~ 7x times) by rotating and mirroring both the mages and masks.
5. **Thresholding and binarization the labels:-** We had converted our masks into grayscale and converted all the pixel values either to 0 or to 255 with a threshold value of 128. Now we divided all the masks by 255 and normalized the masks ending up with only two values - 0 and 1.
6. **Conversion to .h5py:-** Supplying images to the model from the system during training (using ImageDataGenerator) ended up consuming extra time. So for training the model on our system we had packaged all the images and masks files into two separate .h5py files and loaded them onto the RAM. Doing so speeded up the training process.

ISPRS Dataset Preprocessing

The major steps of preprocessing of ISPRS Dataset is listed below

1. **.tif to .pnf:-** First of all, we converted all the .tiff images to .png images with three colour bands (RGB).

2. **Cropping instead of resizing:-** Training our model on large images was not only resource-intensive but also time-consuming. Resizing images to lower dimensions was the answer to our problem. In order to avoid losing information, we had resized all the images without interpolating them.
3. **Converting RGB Masks to Grayscale:-** For some specific pixel values in masks and mapped them to value 255 and other values to 0 (For eg:- (0, 255, 255) -> (255)).
4. **Augmentation:-** We had augmented our data (~ 7x times) by rotating and mirroring both the mages and masks.
5. **Binarization:-** Now we divided all the masks by 255 and normalized the masks ending up with only two values - 0 and 1.
6. **Conversion to .h5py:-** Supplying images to the model from the system during training (using ImageDataGenerator) ended up consuming extra time. So for training the model on our system we had packaged all the images and masks files into two separate .h5py files and loaded them onto the RAM. Doing so speeded up the training process.

3.2 Neural Modelling

We have implemented and trained our data on various deep learning models as mentioned below:

3.2.1 U-net Model: The U-net was developed by Olaf Ronneberger et al. for BioMedical Image Segmentation. The entire architecture can be divided into two parts: an encoder and a decoder.

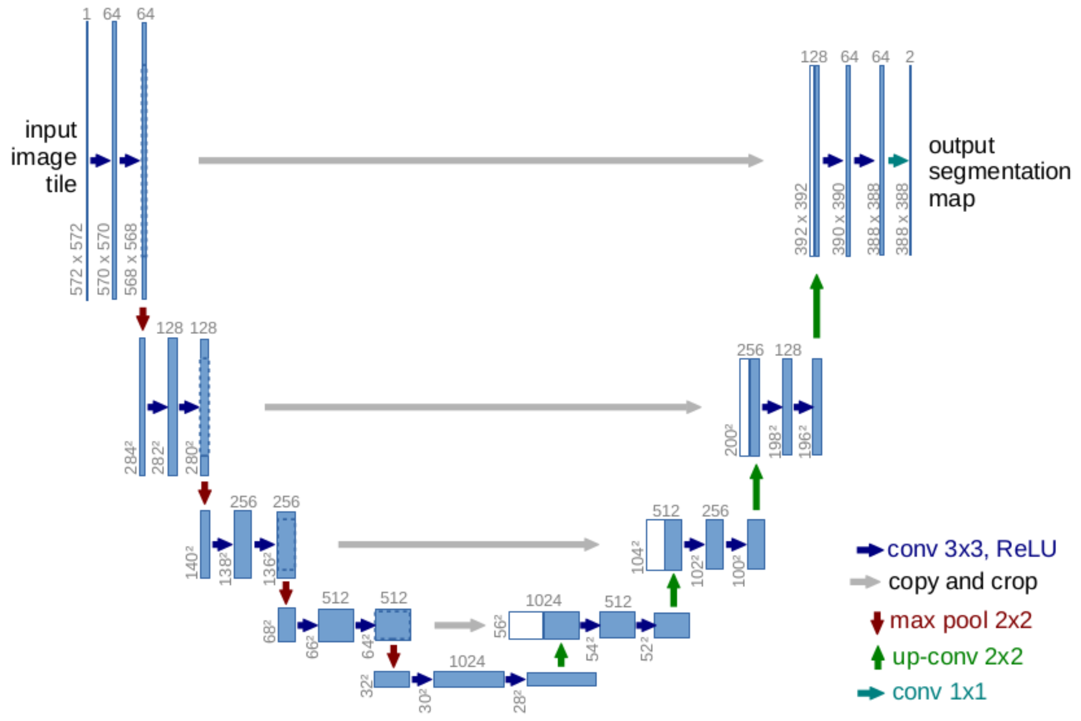


Figure 2: U-net architecture from the original paper[10]

Note that in the original paper, the size of the input image is 572x572x3, however, we had provided an input image of size 256x256x3. Hence the size at various locations will differ from that in the original paper but the core components remained the same.

Encoder (Downsampling):- In this segment convolutional layers are used to learn the temporal features in an image. Also, it makes use of pooling layers in order to downsample it. Due to this segment, we are able to learn about the objects present in an image. This segment learns how a road (or any other class) looks like and can detect it. Not only we added dropout layers in order to prevent overfitting but also we had added Batch Normalization in order to ensure that each layer can learn independently of the previous layer.

Decoder (Upsampling):- Spatial information of the image gets lost because of continuous pooling operations present in the encoder block. With the help of encoder our model becomes aware of the contents of the image but it doesn't know where it is. This is when we make use of our decoder network to reconstruct the spatial information using the feature maps which we have extracted in the previous step. We made use of Transposed convolutions to upsample the image. So instead of using plain interpolation we made use of Conv2DTranspose as it has learnable parameters.

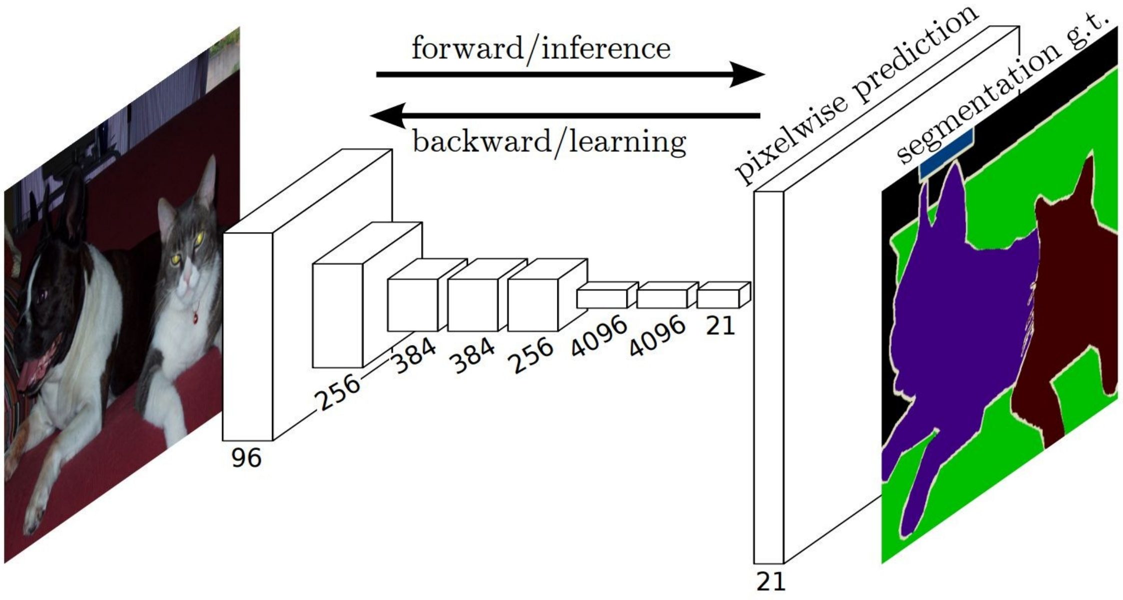


Figure 3: Visual representation of Upsampling[11]

Skip Connections:- Direct connections between the layers in the encoder segment to the layers in the decoder segment are called Skip connections. They are called skip connections because they bridge two layers while ignoring all the intermediate layers. Skip connection helps in providing spatial information to the upsampling layers in order to reconstruct the image.[11]

3.2.2 PSPNet (Pyramid Scene Parsing Network)[14]:

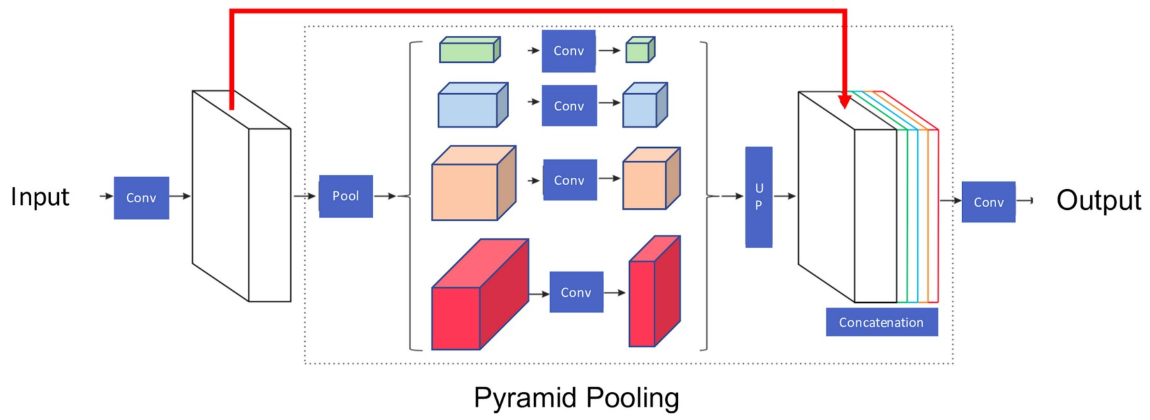


Figure 4: Overview of the pyramid scene parsing networks.[13]

Note:- Here the upsampling block and downsampling block is similar to the above written for U-net's upsampling block and downsampling block. Also, input images were of 240x240 dimensions.

3.2.3 FPN (Feature Pyramid Network)[14]:

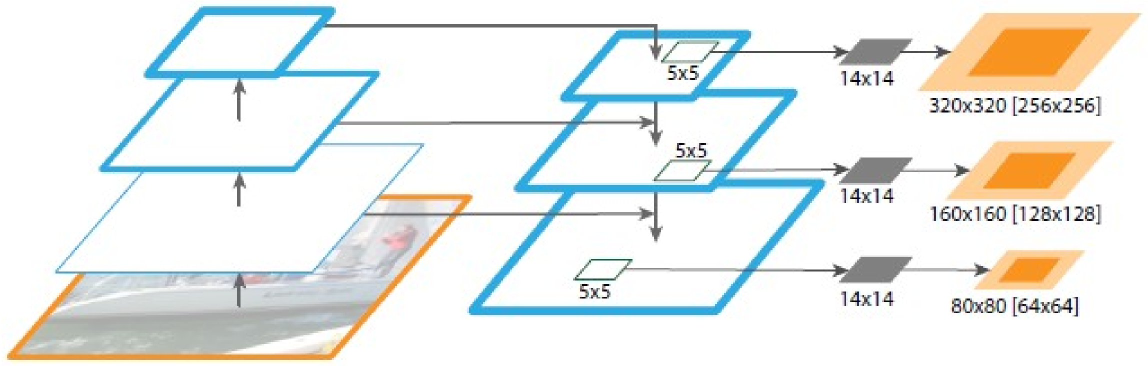


Figure 5: FPN Diagram[15]

3.2.4 ResNet (Residual Network) 101[16]: With the increase in depth of neural networks problem of vanishing/exploding gradient increased so in order to solve that problem a skip/shortcut connection is added which is shown below

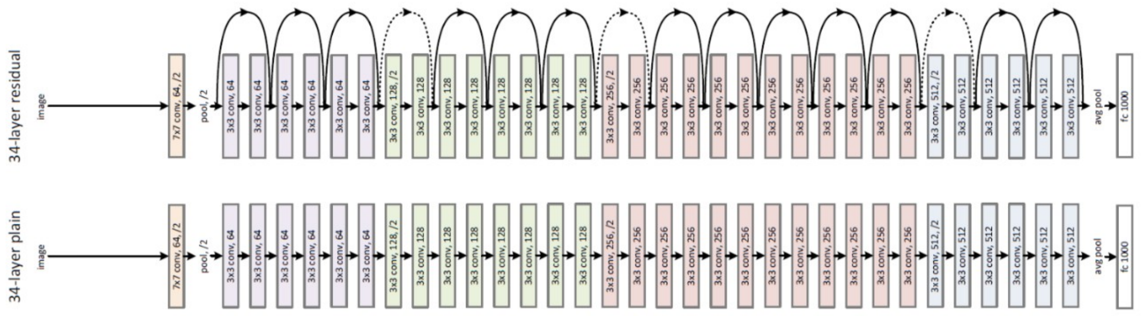


Figure 6: 34-layer plain architecture vs 34-layer residual architecture[17]

3.2.5 DenseNet (Densely Connected Convolutional Networks) 201 [18]: The reason behind selecting dense net as a backbone for FPN was that with the help of DenseNet each layer can gain some extra inputs from all the preceding layers and can pass it further to another output layer. [19]

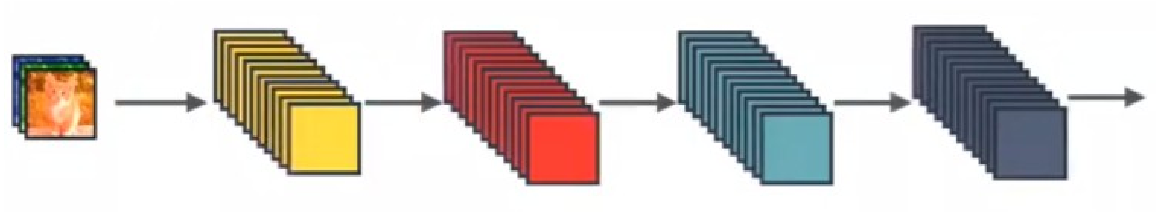


Figure 7: Standard Convnet concept

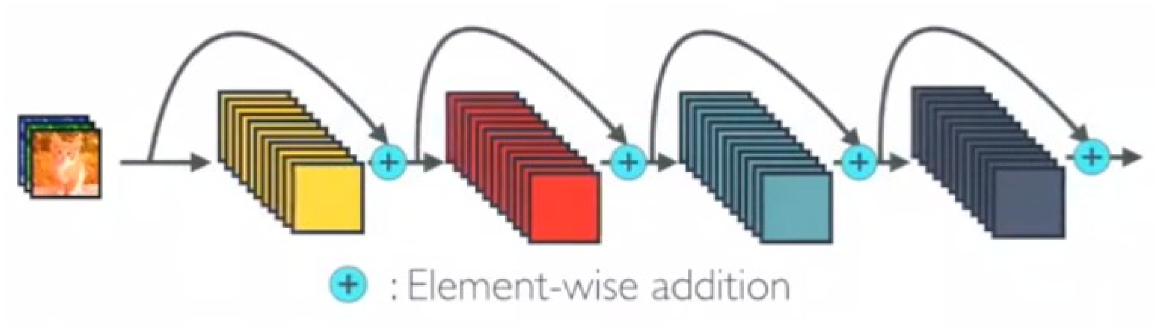


Figure 8: Resnet Concept

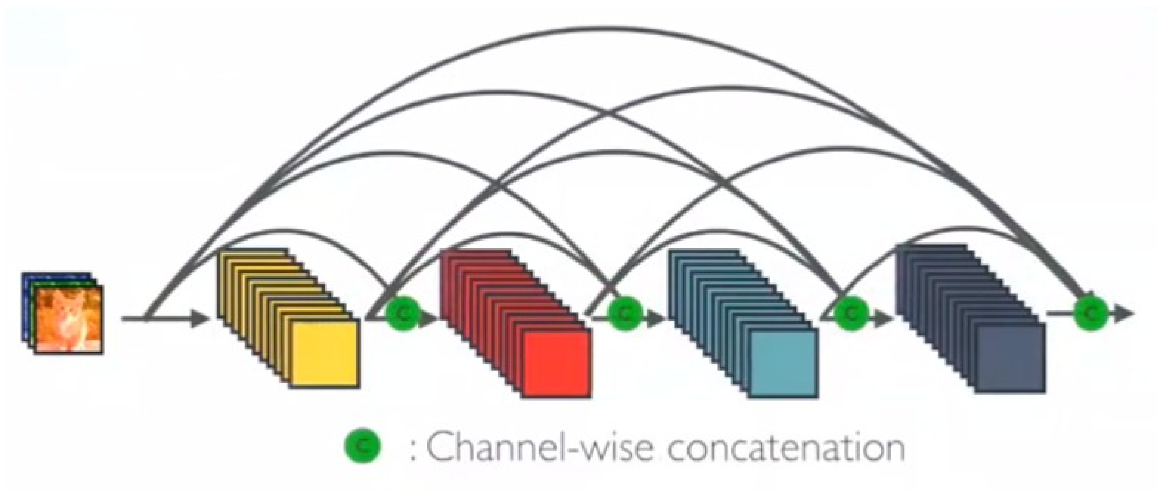


Figure 9: One Dense Block in DenseNet [19]

3.3 Selecting an appropriate model

Below mentioned are some reasons behind our idea of selecting U-net architecture:

1. Pooling or strided convolutions is detrimental for semantic segmentation as spatial information is lost which is avoided in U-net.
2. The architecture by its skip concatenation connections allows the decoder at each stage to learn back relevant features that are lost when pooled in the encoder.
3. Such a deep network can be trained very easily from very few images.
4. Moreover, the network is fast. Segmentation of a 512x512 image takes less than a second on a recent GPU.[10]
5. As the performance provided by a not so deep network such as a simple U-net architecture is very much similar to the performance provided by deep networks such as FPN with different backbones.

3.4 Loss Function

At a pixel level, this segmentation challenge can be considered as a binary classification problem where the model classifies whether each pixel is road (0) or not road (1). One can define the Dice Coefficient as the amount of overlap between the ground truth sample and the predicted sample. Range of the dice coefficient is between 0 and 1 where 0 represents there is no overlap between samples and 1 represents a complete overlap between samples.

Formula of Dice coefficient:- $\frac{2 \times |X \cap Y|}{|X| + |Y|}$

Smooth Dice Loss is simply (1 - Dice Coefficient), this is done to create a minimizable Loss Function.[20]

3.5 Accuracy Metric

We will be using the Jaccard Index, aka Intersection over Union, to tell us how accurate the generated masks are. Intersection over Union is used to measure the correctness of the segmentation maps.

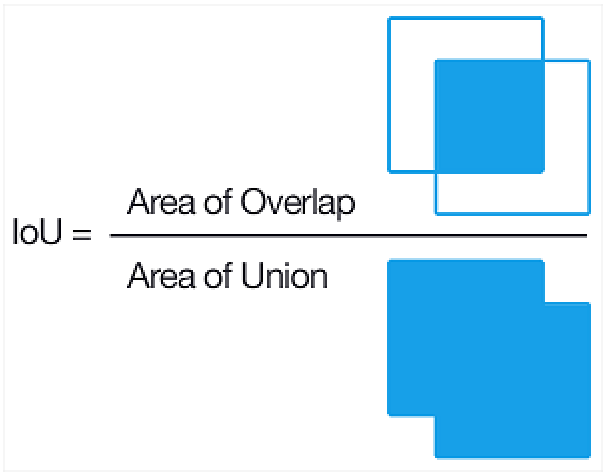


Figure 10: Visual Representation of Intersection over Union

3.6 Optimizers and Hyperparameters

1. The model was compiled using Adam as the optimizer:

   Combination of AdaGrad update and Momentum update and gives better performance.

2. The initial learning rate was set to 0.00001:

   Found out by Babysitting the Learning Process.

3. Dropout with a rate of 0.1:

   To prevent overfitting.

4. Loss Function:- Soft Dice Loss:

   Cross-Entropy loss function can cause problems if the classes have unbalanced representation.
5. Accuracy metric(Jaccard Index):

   Pixel accuracy cannot perform under class imbalance problems (especially when our classes are extremely imbalanced).
6. Activation Functions used were ELU and Sigmoid:

   We chose ELU over ReLU as it provides us with all benefits of ReLU, it doesn't die and provides output with means closer to zero.
7. Initializer used was He Normal [21]:

   The weights are initialized keeping in mind the size of the previous layer which helps in attaining a global minimum of the cost function faster and more efficiently than Xavier/Glorot Initialization.

3.7 Callbacks[9]

As we have used Keras library for training our model we have used following callbacks to train our system:

1. ModelCheckpoint: It monitors cross-validation loss and tries to save the weights of the model with the lowest value of cross-validation loss.
2. EarlyStopping: It helps us in monitoring the cross-validation loss and it would stop the training process if the cross-validation loss does not decrease after a certain number of epochs.
3. ReduceLROnPlateau: It monitors cross-validation loss and tries to reduce the learning rate if the cross-validation loss doesn’t decrease after a certain number of epochs.

CHAPTER 4: SYSTEM ANALYSIS & DESIGN

4.1 Project Flow Diagram

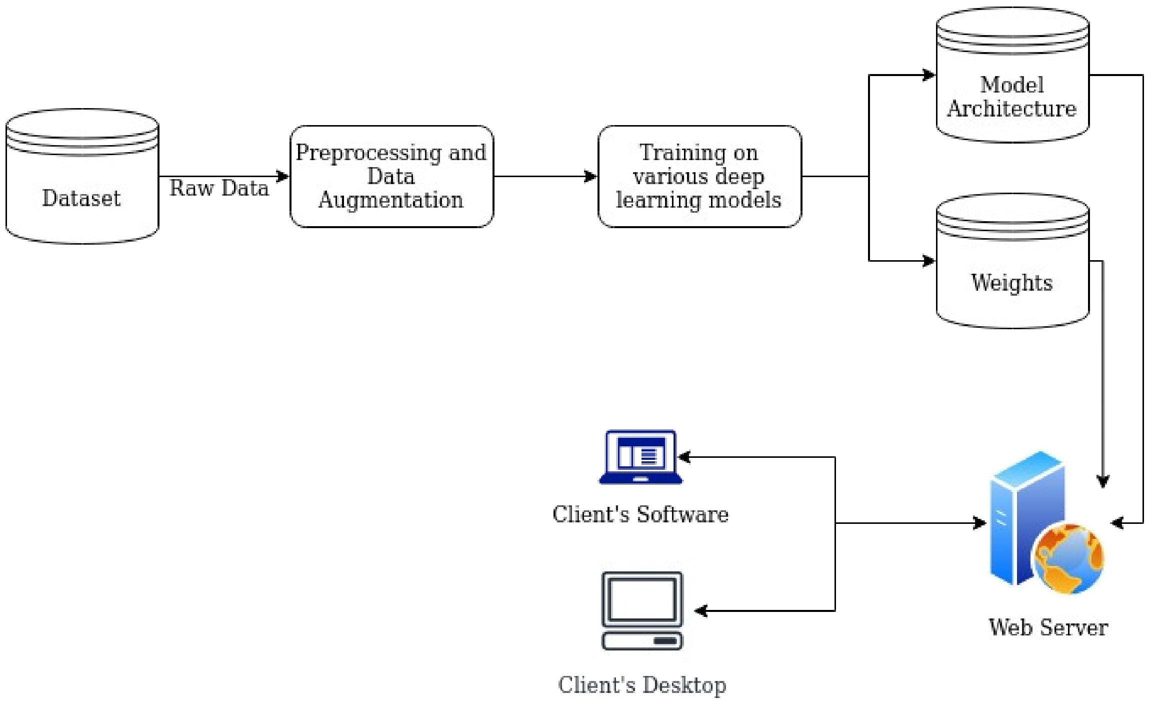


Figure 11: Data Flow Diagram

4.2 Requirement Specifications

4.2.1 Functional Requirements: Functional Requirements of all the present modules is listed below:

Map Segmentation module

This module must successfully take aerial images as input and segment it into various classes and create a segmented image file out of it.

QGIS support module

This module must successfully take a segmented image as an input and return a CSV file which would add vector layers of different segmented classes in QGIS.

Development Tracker module

This module must successfully perform a time series analysis on a set of segmented images by showing the percentage of change in selected classes.

LabelMe support module

This module must successfully return a JSON file which could be successfully loaded into LabelMe software.

REST API

All the data must be successfully sent through the network and received using HTTP method calls.

4.2.2 Non-Functional Requirements:

Availability

The software must be available during working days without any interruptions as an entire department will be dependent on our software.

Platform

The software must be platform-independent so that every authorised user of the institute can access it from any machine.

Adaptability

The software must be adaptable to various types of different datasets and it also must be such that it can change the current model to the latest model available.

Performance

The system must be responsible enough that it should not hang or be laggy during peak hours.

Security

Login ID - Any user who uses the system shall have a Login ID and Password.

Stability

The system outcome/output won’t change. Same output will be always for a given input.

Robustness

The system must be able to handle errors during its execution and also must deal with input errors.

4.3 Use Case Diagram

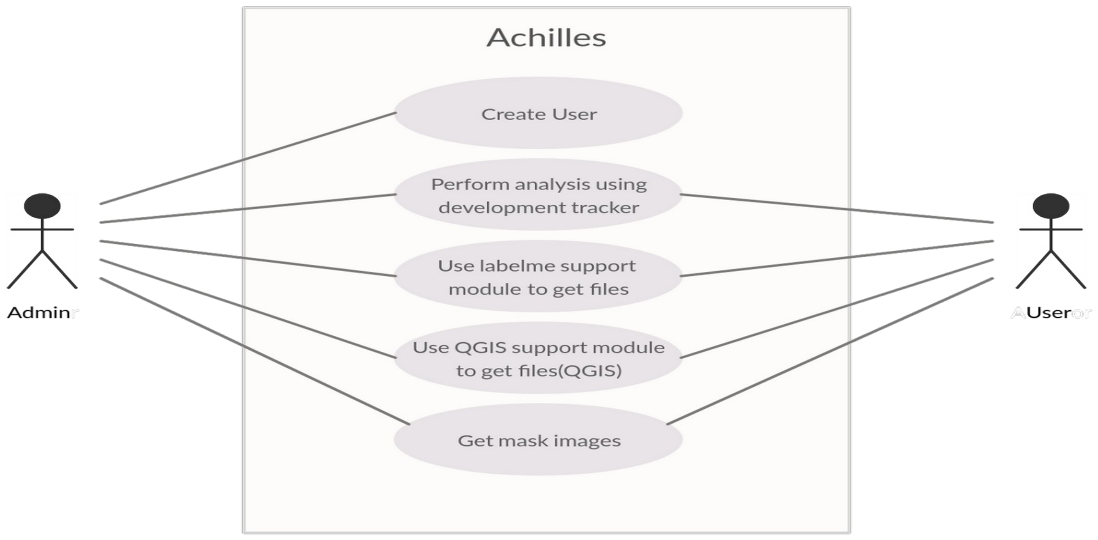

Achilles
Create User
Perform analysis using development tracker
Use labelme support module to get files
Use QGIS support module to get files(QGIS)
Get mask images
Admin
User

Figure 12: Use Case Diagram

4.4 Entity-Relationship Diagram

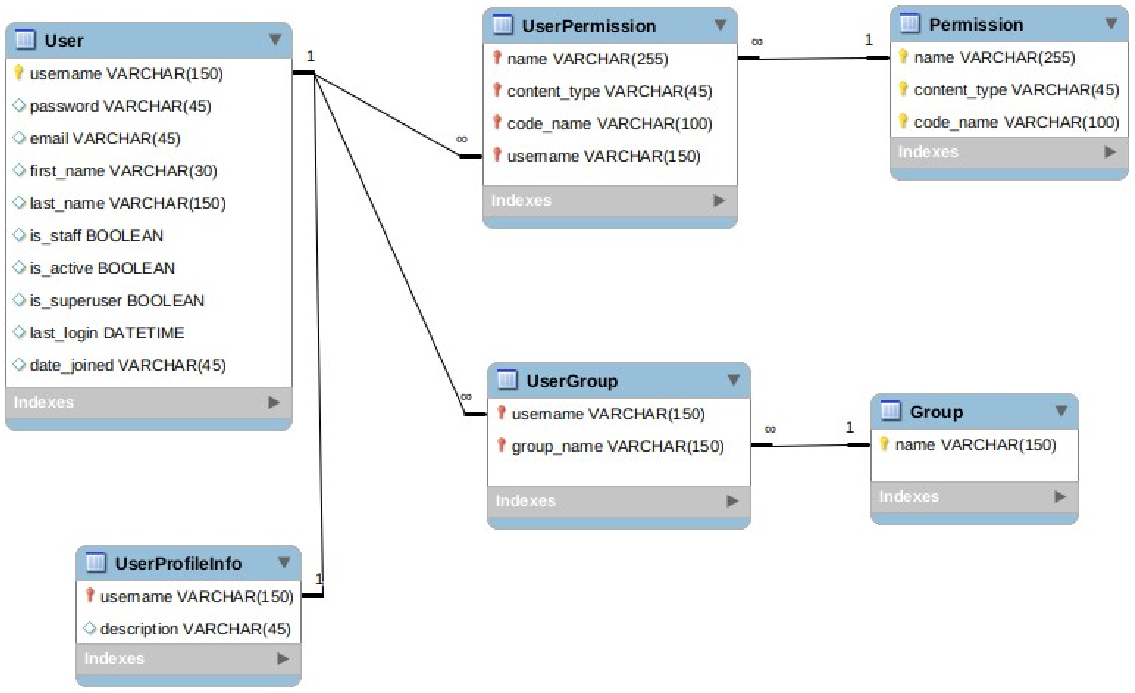


Figure 13: Entity-Relationship Diagram

4.5 Activity Diagram

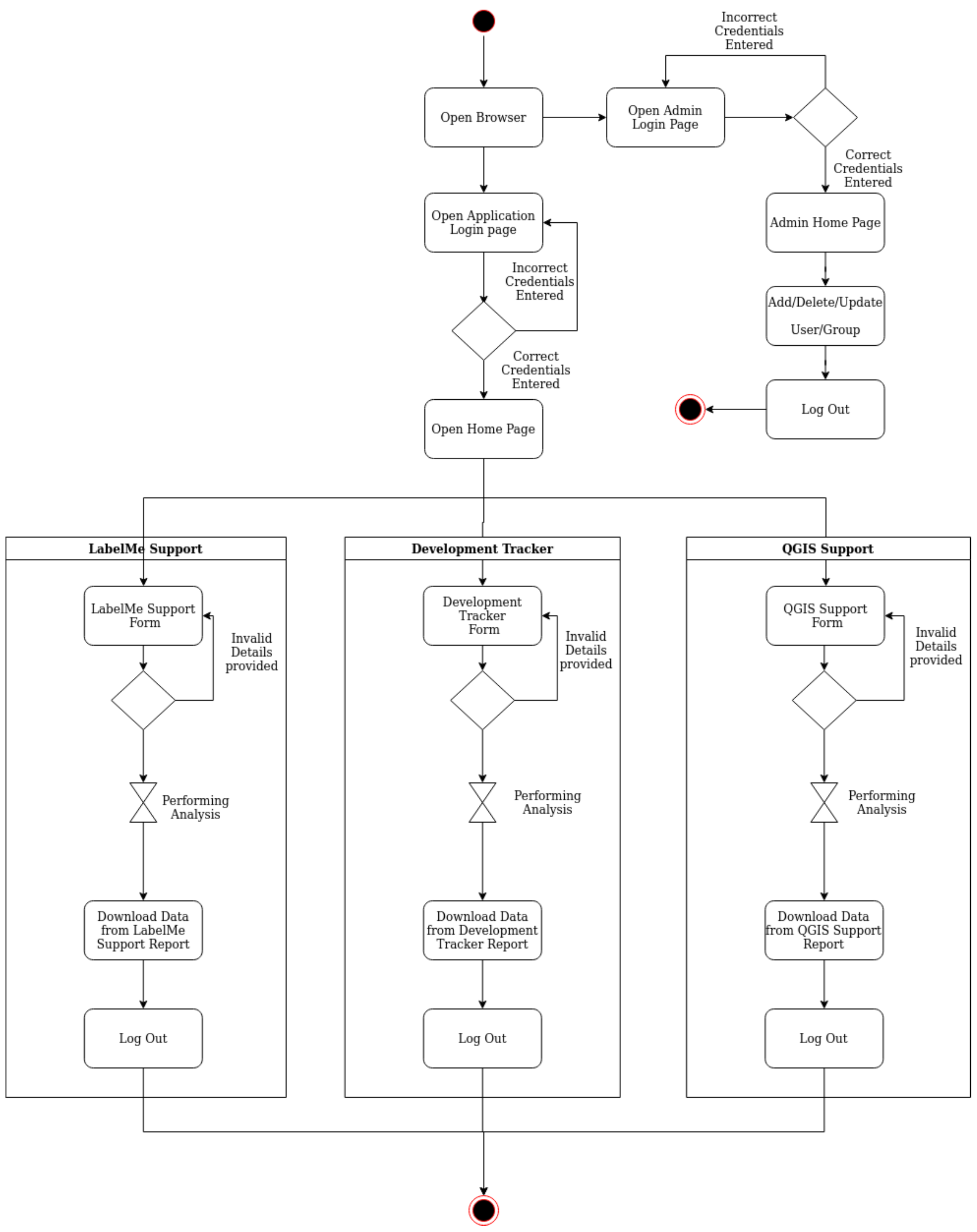


Figure 14: Activity Diagram

4.6 Testing Process

The testing document is attached as a separate file.

Chapter 5: RESULTS

5.1 Model Performance

In this section, we experimented with various models with different backbones on both Massachusetts Road and Building Dataset.

Model	mIoU	F1-Score
UNet	92.01%	99.01%
ResNet 101	93%	99.1%
UNet with Efficient Net b7 as a backbone	93.38%	98.81%
PSPNet	93.77%	98.7%
FPN with DenseNet 101 as a backbone	93.81%	98.83%

Table 1: mIoU values for various models on the road test set

5.2 Visualization Results

5.2.1 Roads:

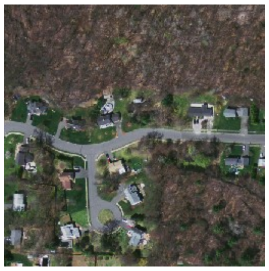
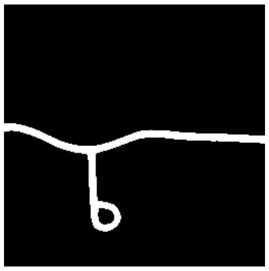
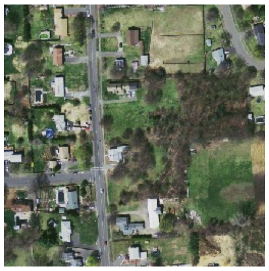
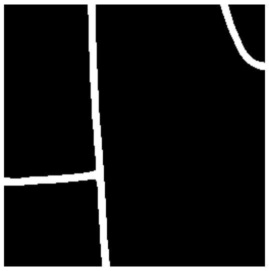
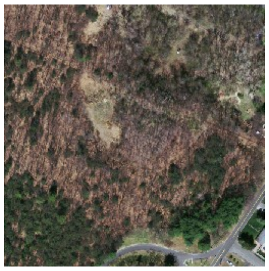
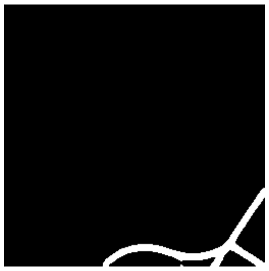
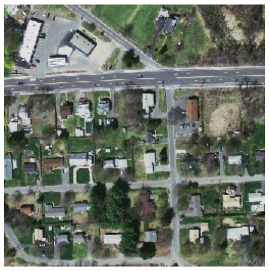
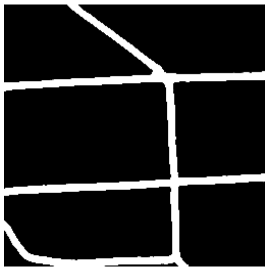
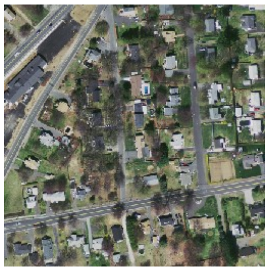
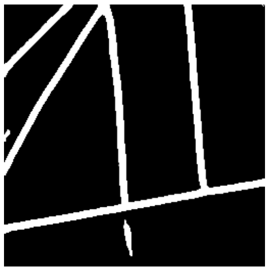
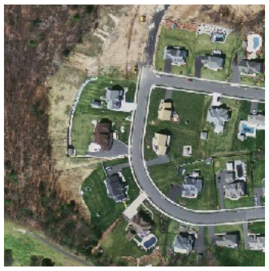
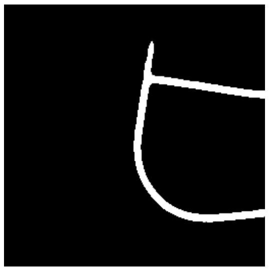
Image	**Mask**	**Image**	**Mask**
			
			
			

Table 2: Results for running our software on-road prediction, the bottom row shows minor errors of our system

5.2.2 Buildings:

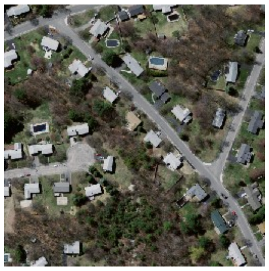
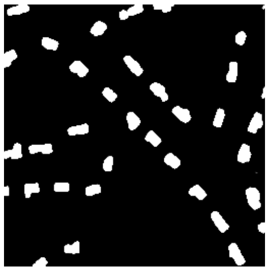
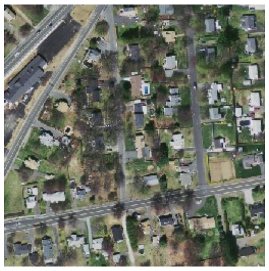
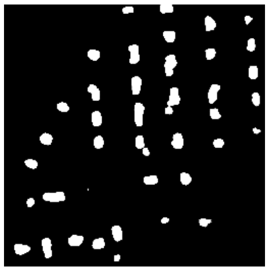
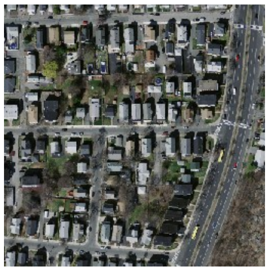
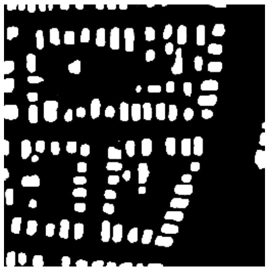
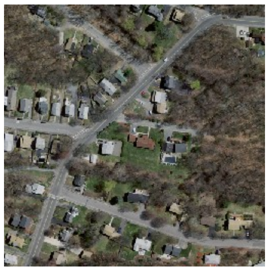
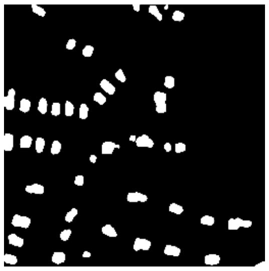
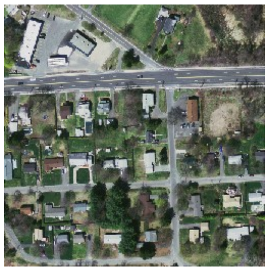
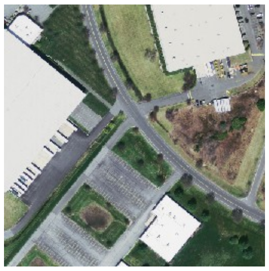
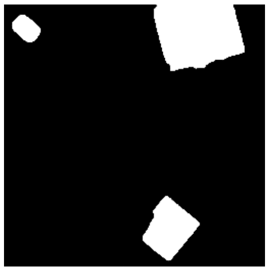
Image	Mask	Image	Mask
			
			
			

Table 3: Results for running our software on building prediction, the bottom row shows minor errors of our system

5.2.3 Cars:

Image	Mask	Image	Mask

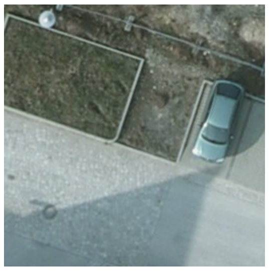
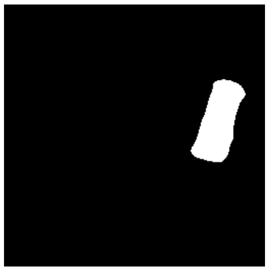
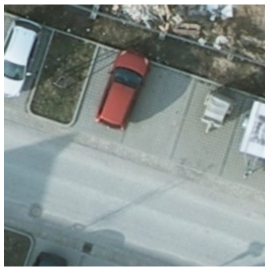
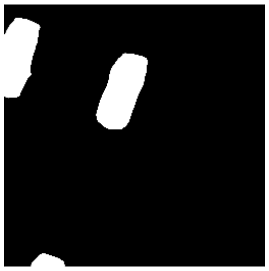
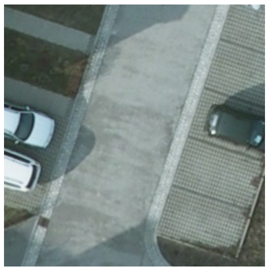
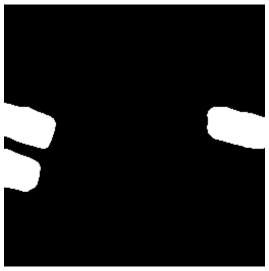
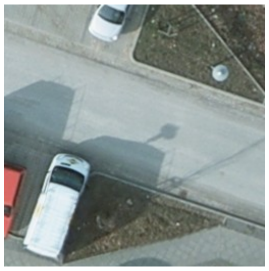
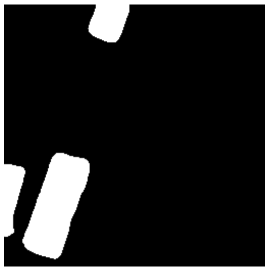
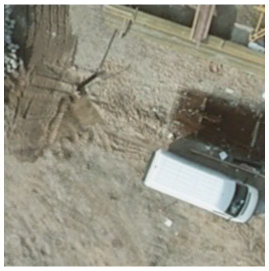
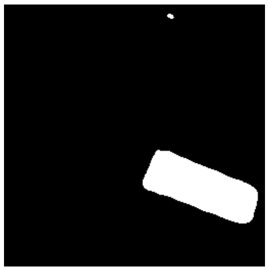
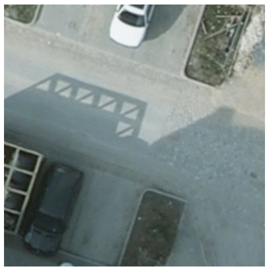
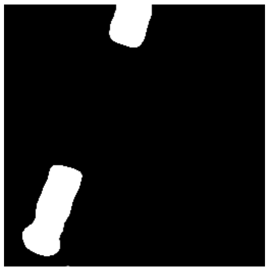
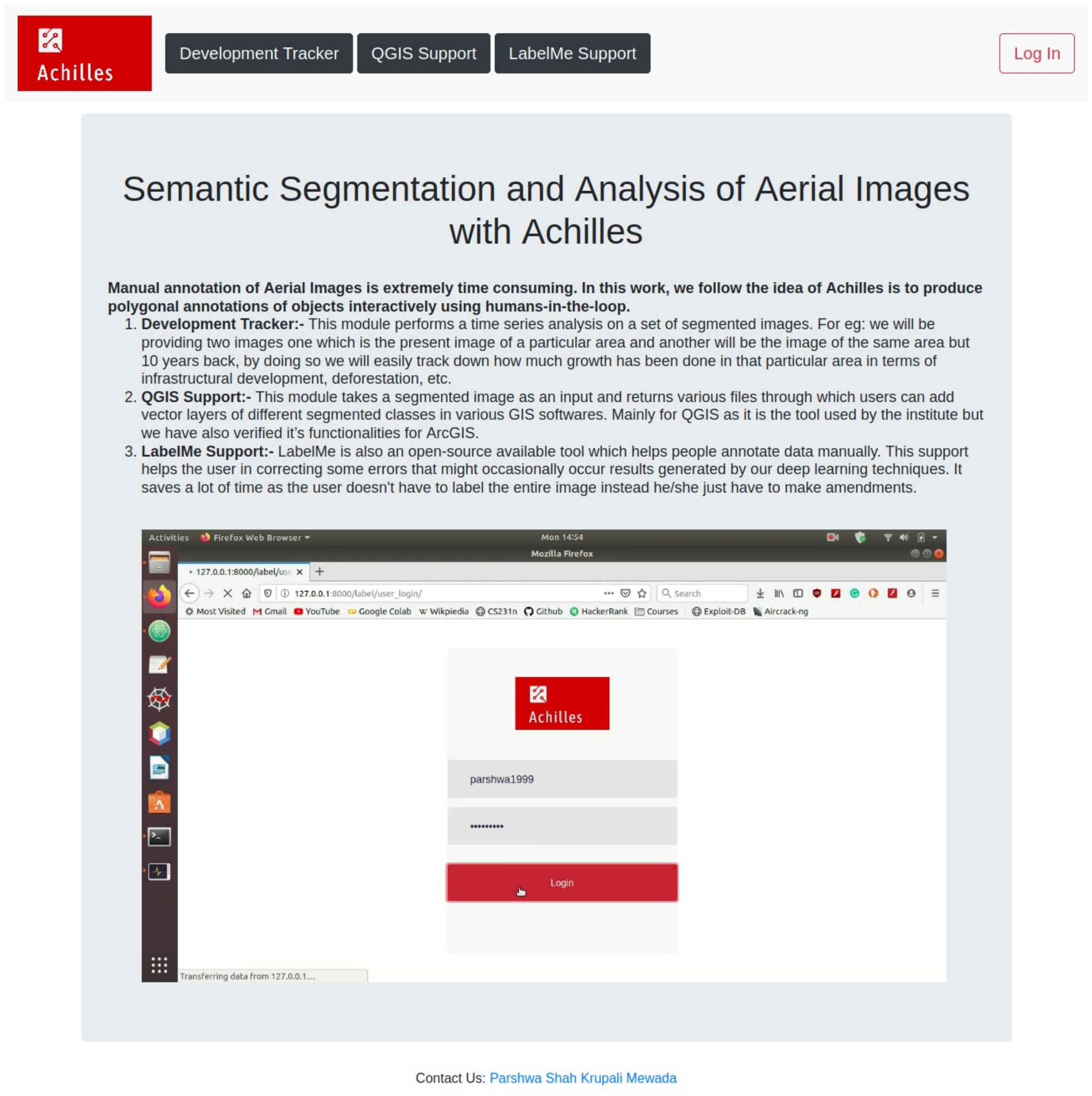
			
			
			

Table 4: Results for running our software on car prediction, the bottom row shows minor errors of our system

5.3 API Response



Achilles | Development Tracker | QGIS Support | LabelMe Support | Log In

Semantic Segmentation and Analysis of Aerial Images with Achilles

Manual annotation of Aerial Images is extremely time consuming. In this work, we follow the idea of Achilles is to produce polygonal annotations of objects interactively using humans-in-the-loop.

1. **Development Tracker:-** This module performs a time series analysis on a set of segmented images. For eg: we will be providing two images one which is the present image of a particular area and another will be the image of the same area but 10 years back, by doing so we will easily track down how much growth has been done in that particular area in terms of infrastructural development, deforestation, etc.
2. **QGIS Support:-** This module takes a segmented image as an input and returns various files through which users can add vector layers of different segmented classes in various GIS softwares. Mainly for QGIS as it is the tool used by the institute but we have also verified it's functionalities for ArcGIS.
3. **LabelMe Support:-** LabelMe is also an open-source available tool which helps people annotate data manually. This support helps the user in correcting some errors that might occasionally occur results generated by our deep learning techniques. It saves a lot of time as the user doesn't have to label the entire image instead he/she just have to make amendments.

Contact Us: Parshwa Shah Krupali Mewada

Figure 15: Image of the Index page

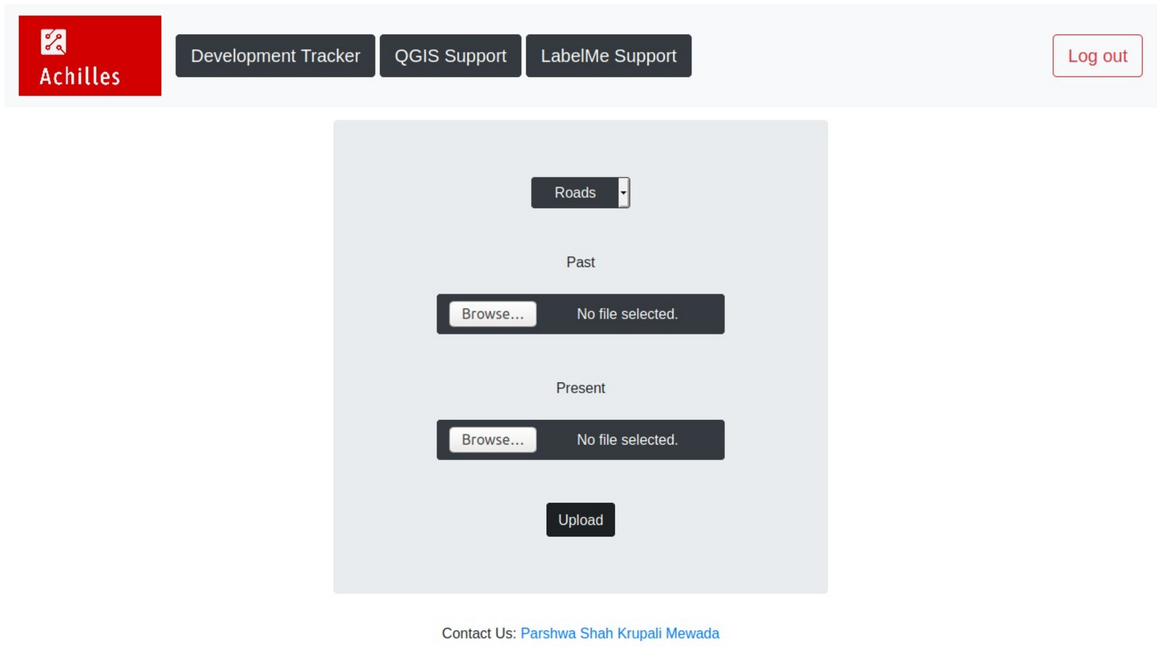
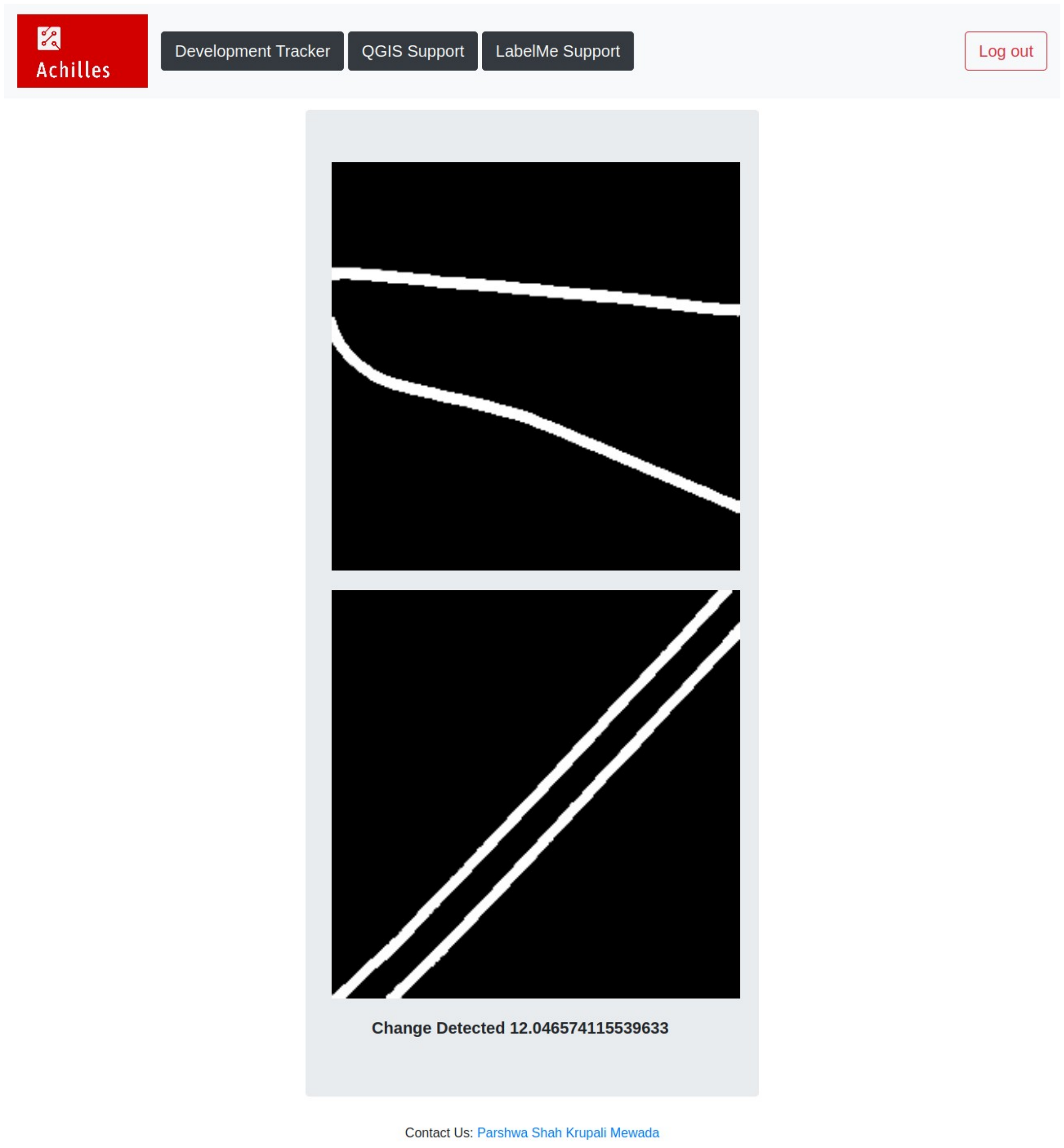


Figure 16: Development Tracker Form

This module performs a time series analysis on a set of segmented images. For eg: we will be providing two images one which is the present image of a particular area and another will be the image of the same area but 10 years back, by doing so we will easily track down how much growth has been done in that particular area in terms of infrastructural development, deforestation, etc,



Change Detected 12.046574115539633

Contact Us: Parshwa Shah Krupali Mewada

Figure 17: Development Tracker Report Response

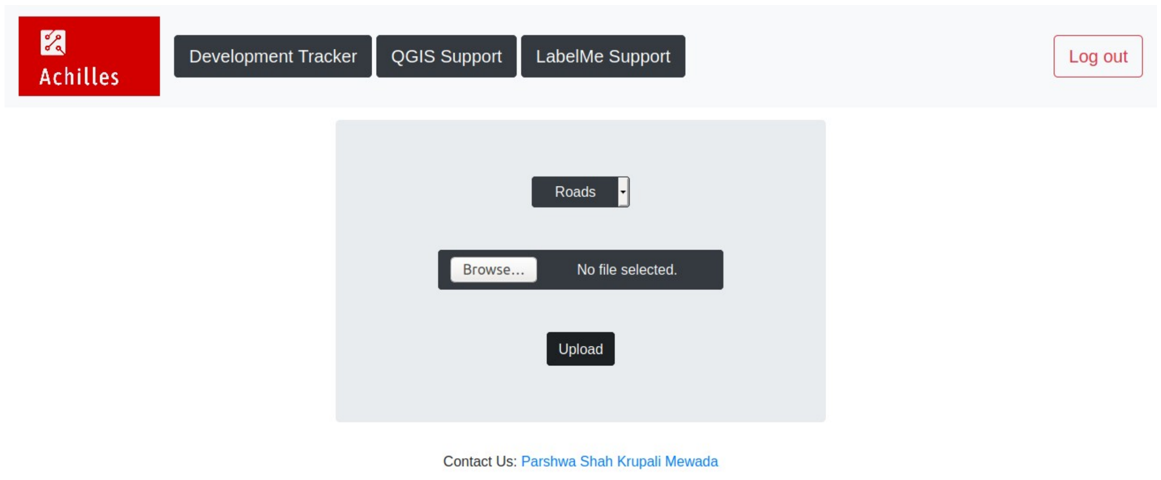


Figure 18: QGIS and LabelMe form

QGIS support: This module takes a segmented image as an input and returns various files through which users can add vector layers of different segmented classes in various GIS softwares. Mainly for QGIS as it is the tool used by the institute but we have also verified it’s functionalities for ArcGIS.

LabelMe support: LabelMe is also an open-source available tool which helps people annotate data manually. This support helps the user in correcting some errors that might occasionally occur results generated by our deep learning techniques. It saves a lot of time as the user doesn't have to label the entire image instead he/she just has to make amendments.

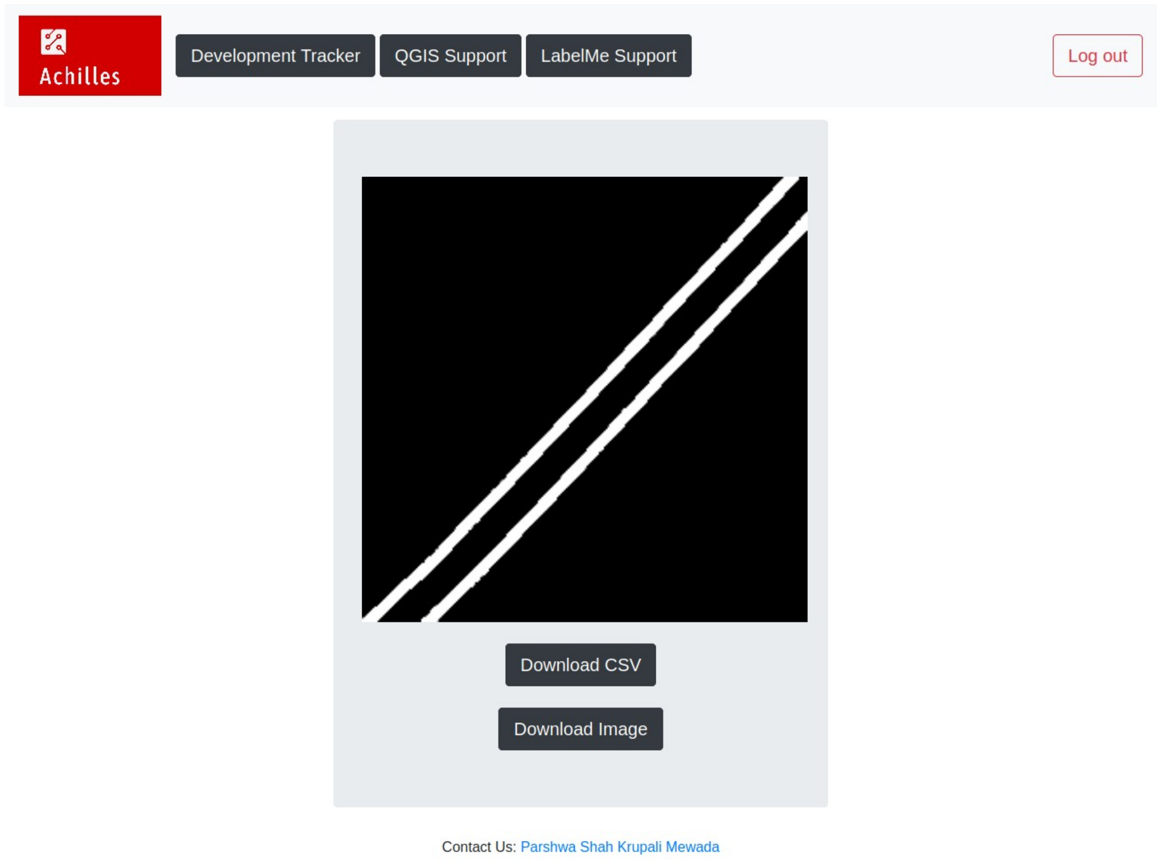
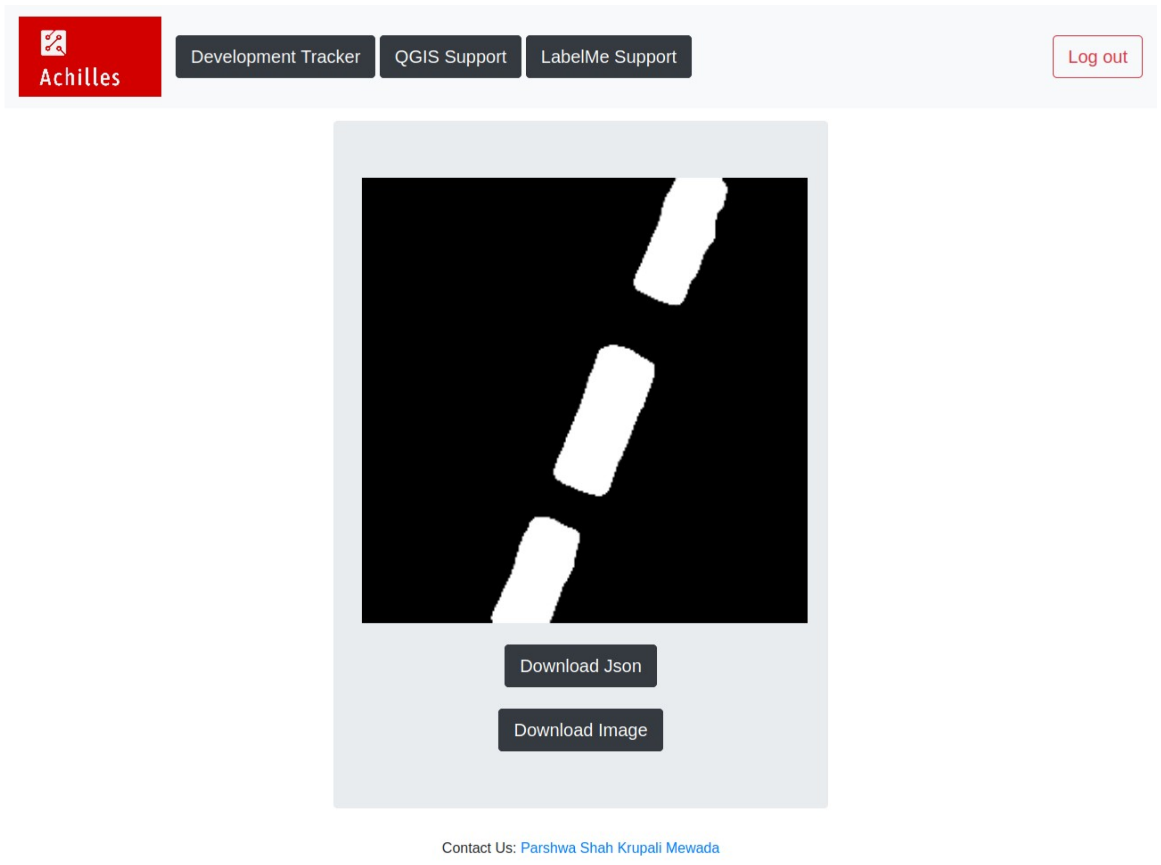


Figure 19: QGIS Support Response



Contact Us: Parshwa Shah Krupali Mewada

Figure 20: LabelMe Support Response

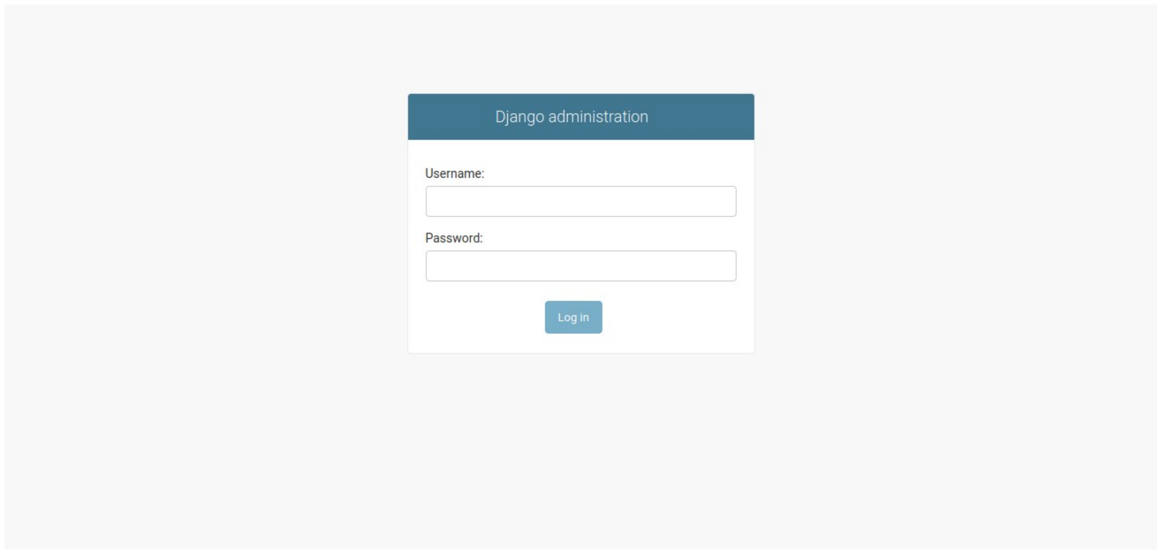
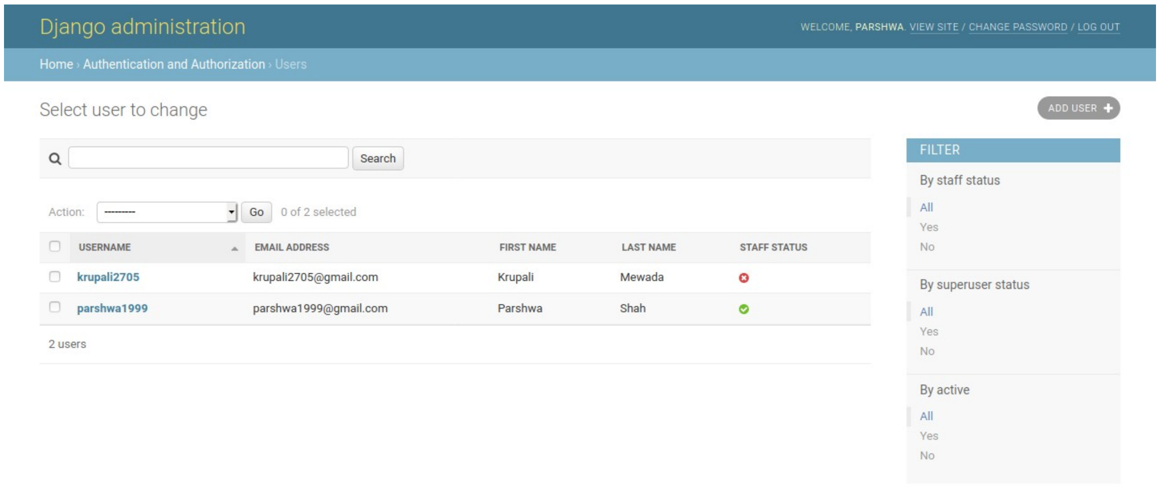


Figure 21: Admin Login Page



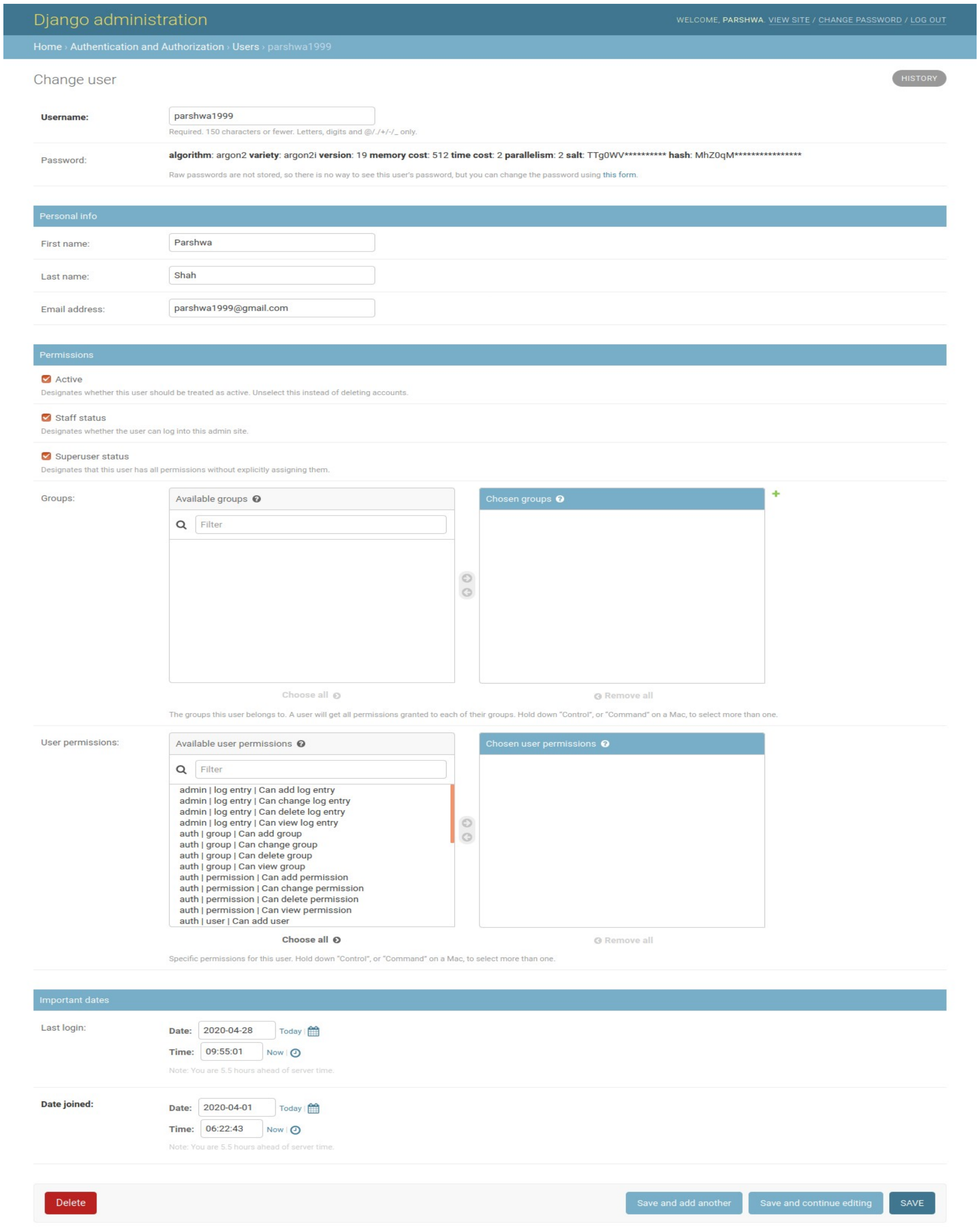
Select user to change

0 of 2 selected

USERNAME	EMAIL ADDRESS	FIRST NAME	LAST NAME	STAFF STATUS
krupali2705	krupali2705@gmail.com	Krupali	Mewada	✗
parshwa1999	parshwa1999@gmail.com	Parshwa	Shah	✓

2 users

Figure 22: Admin Access page



Django administration

WELCOME, **PARSHWA**. VIEW SITE / CHANGE PASSWORD / LOG OUT

Home › Authentication and Authorization › Users › parshwa1999

Change user

HISTORY

Username: parshwa1999

Required. 150 characters or fewer. Letters, digits and @/./+/-/_ only.

Password: **algorithm**: argon2 **variety**: argon2i **version**: 19 **memory cost**: 512 **time cost**: 2 **parallelism**: 2 **salt**: TTg0WV********** **hash**: MhZ0qM****************

Raw passwords are not stored, so there is no way to see this user's password, but you can change the password using this form.

Personal info

First name: Parshwa

Last name: Shah

Email address: parshwa1999@gmail.com

Permissions

✓ Active

Designates whether this user should be treated as active. Unselect this instead of deleting accounts.

✓ Staff status

Designates whether the user can log into this admin site.

✓ Superuser status

Designates that this user has all permissions without explicitly assigning them.

Groups:

Available groups

Filter

Chosen groups

Choose all

Remove all

The groups this user belongs to. A user will get all permissions granted to each of their groups. Hold down "Control", or "Command" on a Mac, to select more than one.

User permissions:

Available user permissions

Filter

admin | log entry | Can add log entry
admin | log entry | Can change log entry
admin | log entry | Can delete log entry
admin | log entry | Can view log entry
auth | group | Can add group
auth | group | Can change group
auth | group | Can delete group
auth | group | Can view group
auth | permission | Can add permission
auth | permission | Can change permission
auth | permission | Can delete permission
auth | permission | Can view permission
auth | user | Can add user

Chosen user permissions

Choose all

Remove all

Specific permissions for this user. Hold down "Control", or "Command" on a Mac, to select more than one.

Important dates

Last login:

Date: 2020-04-28 Today

Time: 09:55:01 Now

Note: You are 5.5 hours ahead of server time.

Date joined:

Date: 2020-04-01 Today

Time: 06:22:43 Now

Note: You are 5.5 hours ahead of server time.

Delete

Save and add another

Save and continue editing

SAVE

Figure 23: User Details view using Admin account

Chapter 6: CONCLUSION

In this project we have designed a system to automate the work of labelling various datasets manually and also the whole process of performing time series analysis. All of these functionalities were presented as a web app on Django platform. Also the system has added support of various applications used by the industry.Our application segments each map image into various segments through which we can perform time series analysis on images and track development in various parts of our country. Moreover, our time series analysis would help us in tracking various environmental changes such as deforestation, afforestation, urbanization, etc and infrastructural changes such as Rural development. Also, with the help of additional support files to load all the detected classes as the layers in QGIS and improve segmentation parts through LabelMe.

Future Work:

Deploying our entire system on a physical server at the institute accessible on intranet and performing further testing on it. Create a module in order to track user activities and link it to the current system present for project tracking. This module will mitigate the task of tracking individual employees' work by the institute.

Chapter 7: REFERENCES

[1] Shetty, S., Shetty, S., Gall, R., Lobo, S., Davis, V., & Sunith ShettyData Science. (2018, September 20). Why TensorFlow always tops machine learning and artificial intelligence tool surveys. Retrieved from https://hub.packtpub.com/tensorflow-always-tops-machine-learning-artificial-intelligence-tool-surveys/

[2]Keras. (2020, April 21). Retrieved from https://en.wikipedia.org/wiki/Keras

[3]Introduction to OpenCV-Python Tutorials. (n.d.). Retrieved from https://docs.opencv.org/master/d0/de3/tutorial_py_intro.html

[4]Uses of Django: Top 10 Uses of Django You Must Learn in Real World. (2020, April 21). Retrieved from https://www.educba.com/uses-of-django/

[5]Argon2. (2020, April 04). Retrieved from https://en.wikipedia.org/wiki/Argon2

[6]Python Imaging Library. (2020, April 10). Retrieved from https://en.wikipedia.org/wiki/Python_Imaging_Library

[7] Acuna, David, Ling, Huan, Kar, Fidler, & Sanja. (2018, March 26). Efficient Interactive Annotation of Segmentation Datasets with Polygon-RNN. Retrieved from https://arxiv.org/abs/1803.09693

[8](n.d.). Retrieved from https://roadmaps.csail.mit.edu/roadtracer/

[9]Paul, J. (2019, December 19). Using Deep Learning to Segment Roads in Aerial Images. Retrieved from https://towardsdatascience.com/road-segmentation-727fb41c51af

[10]Ronneberger, O., Fischer, P., & Brox, T. (2015, October). U-net: Convolutional networks for biomedical image segmentation. In International Conference on Medical image computing and computer-assisted intervention (pp. 234-241). Springer, Cham.

[11]CS231n: Convolutional Neural Networks for Visual Recognition. (n.d.). Retrieved from http://cs231n.stanford.edu/

[12]Zhao, H., Shi, J., Qi, X., Wang, X., & Jia, J. (2017). Pyramid scene parsing network. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 2881-2890).

[13] Sasgarit. (2019, October 16). Deep Semantic Segmentation of Natural and Medical Images: A Review. Retrieved from https://www.groundai.com/project/deep-semantic-segmentation-of-natural-and-medical-images-a-review/1

[14]Lin, T. Y., Dollár, P., Girshick, R., He, K., Hariharan, B., & Belongie, S. (2017). Feature pyramid networks for object detection. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 2117-2125).

[15]Tsang, S. (2019, March 20). Review: FPN-Feature Pyramid Network (Object Detection). Retrieved from https://towardsdatascience.com/review-fpn-feature-pyramid-network-object-detection-262fc7482610

[16]He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770-778).

[17]Tsang, S. (2019, March 20). Review: ResNet - Winner of ILSVRC 2015 (Image Classification, Localization, Detection). Retrieved from https://towardsdatascience.com/review-resnet-winner-of-ilsvrc-2015-image-classification-localization-detection-e39402bfa5d8

[18]Huang, G., Liu, Z., Weinberger, K. Q., & van der Maaten, L. Densely Connected Convolutional Networks. arXiv 2017. arXiv preprint arXiv:1608.06993.

[19]Tsang, S. (2019, March 20). Review: DenseNet - Dense Convolutional Network (Image Classification). Retrieved from https://towardsdatascience.com/review-densenet-image-classification-b6631a8ef803

[20]Jeremy Jordan. (2018, December 18). Evaluating image segmentation models. Retrieved from https://www.jeremyjordan.me/evaluating-image-segmentation-models/

[21]He, K., Zhang, X., Ren, S., & Sun, J. (2015). Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In Proceedings of the IEEE international conference on computer vision (pp. 1026-1034).