

**Ahmedabad
University**

Map Segmentation

Parshwa Shah (1641068)

Krupali Mewada (1501056)

Project Details

We are currently doing our internship at BISAG (Bhaskaracharya Institute for Space Applications and Geoinformatics)

On-campus mentor: Dr Kuntal Patel

Off-campus mentor: Mr Vishal Patel (BISAG)

Group size: 2

Group members:

Parshwa Shah (1641068)

Krupali Mewada (201501056)

Problem Statement

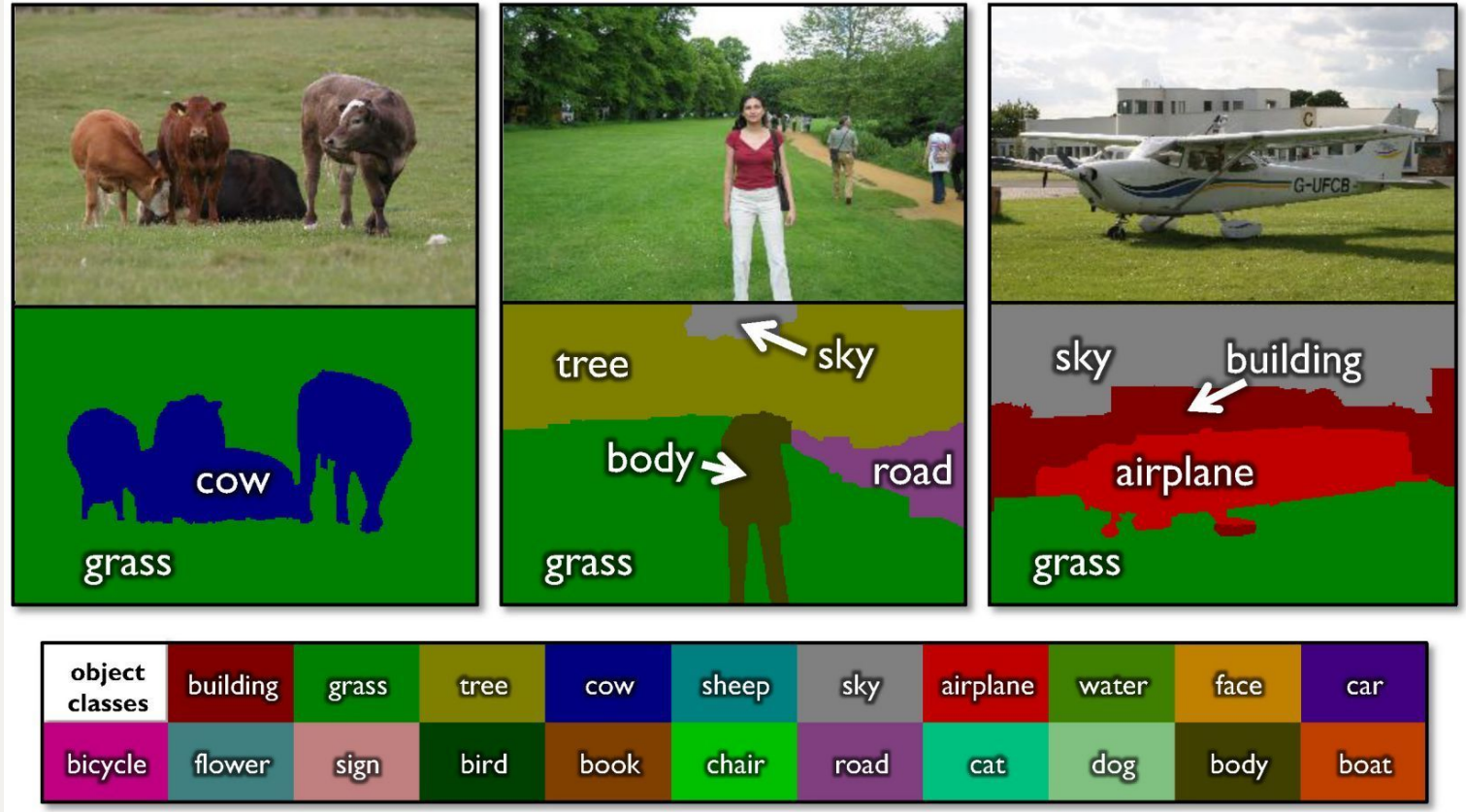
Everyday Geospatial Data Storages are deluged with millions of optical overhead imagery captured from airborne or space-borne platforms.

Due to such a large amount of time series analysis on images and tracking development in various parts of our country becomes a brobdingnagian task for GIS institutes.

Solution:- Machine vision techniques must be employed if we want to make any use of the available data.

Semantic Segmentation

- Label every pixel !
- Simultaneous detection and segmentation.



1. SELECTING DATASET

Dataset

- 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 dataset contained 1171 aerial images, along with their respective maps. They are 1500 x 1500 in dimension and are in .tiff format.
- ISPRS Potsdam 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.

Massachusetts Roads and Building Dataset



Image



Mask

2. PREPROCESSING DATA

Preprocessing

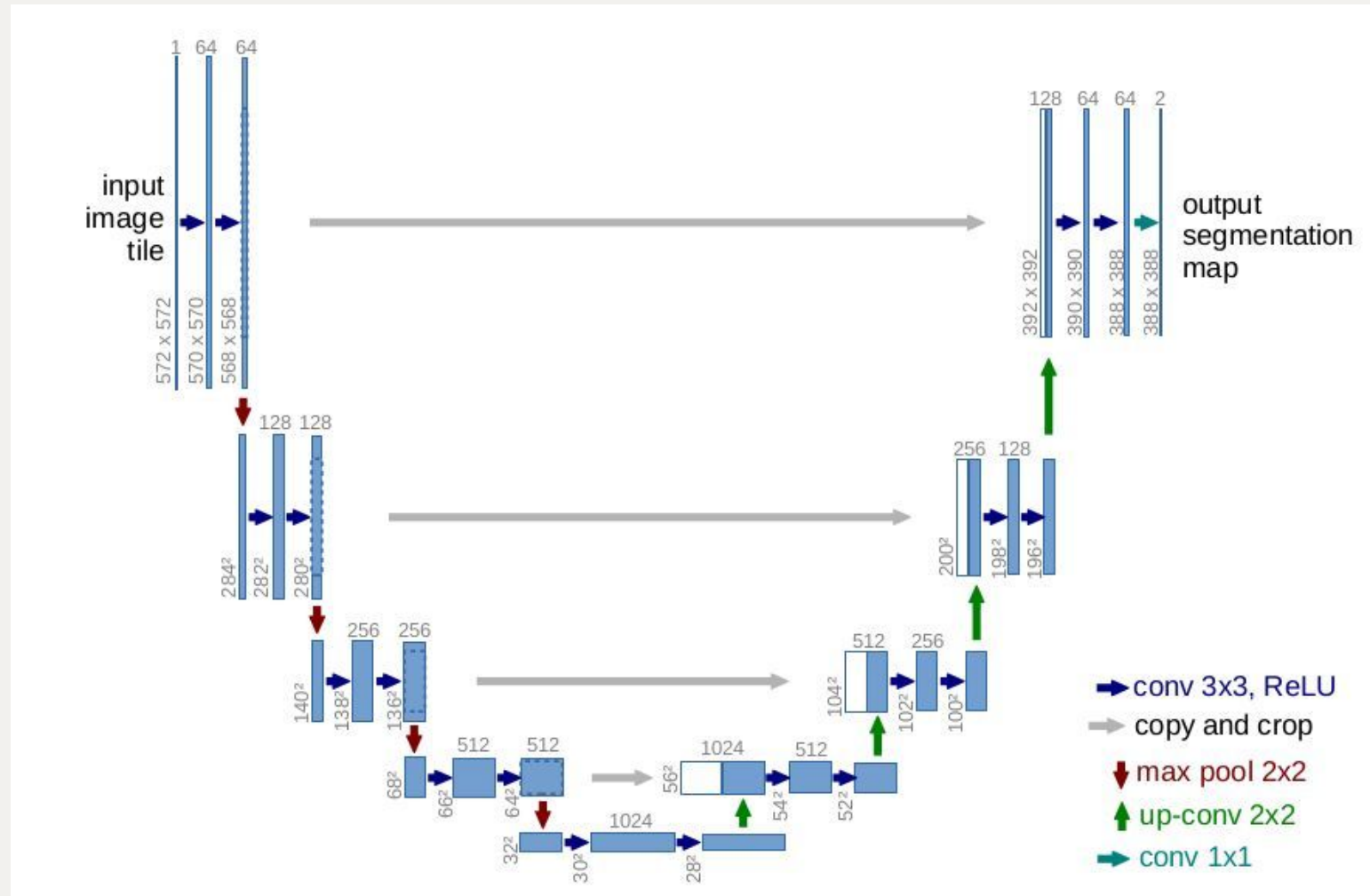
1. **.tiff to .pnf:-** First of all we converted all the .tiff images to .png images with three colour bands (RGB) and with resolution of 1.2m.
2. **Hand-picking:-** There are few images in the dataset where a big chunk of the aerial images are missing. Since this can throw the model off, we manually removed them.
3. **Cropping instead of resizing:-** Training our model on large images is not only resource-intensive but is bound to take a lot of time as well. Resizing images to lower dimensions can be an answer, but resizing regardless of the interpolation method we end up losing information.
4. **Augmentation:-** We have augmented our data (~ 7x times) by rotating and mirroring both the mages and masks.

Preprocessing

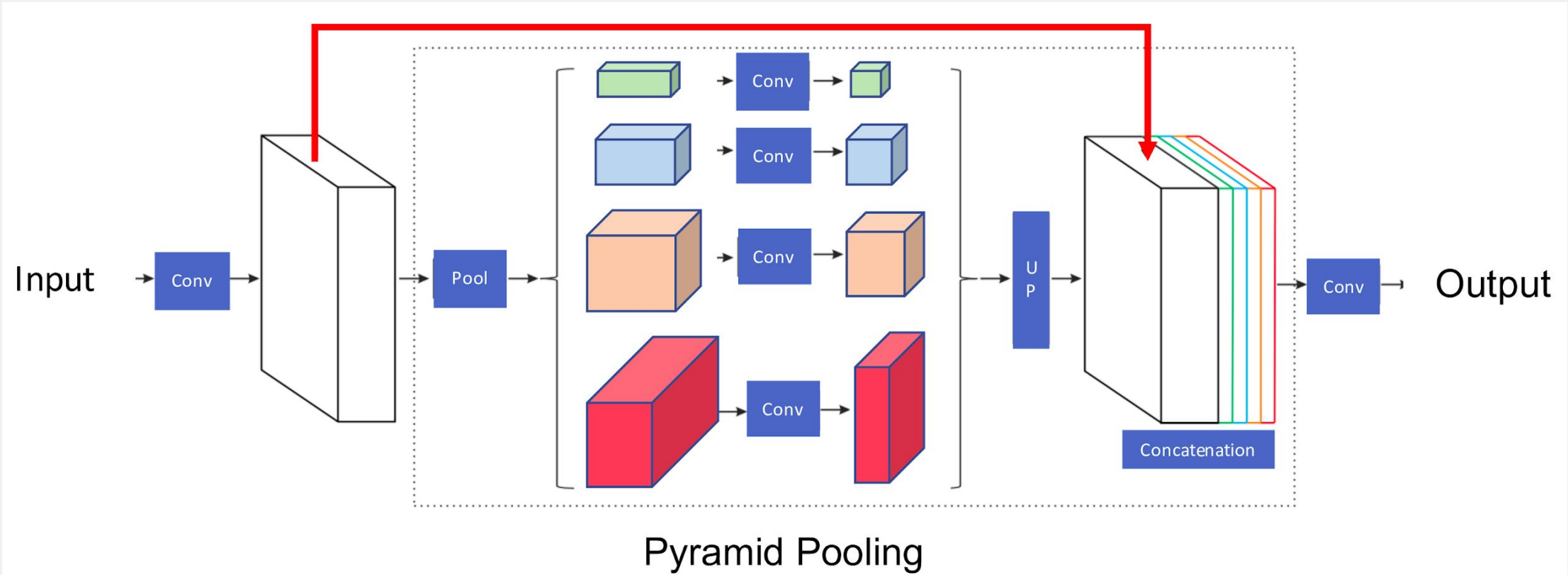
5. **Thresholding and binarization the labels:-** We have converted our masks into grayscale and converted all the pixel values either to 0 or to 255 with a threshold value of 128. Now we divided all the masks by 255 and normalized the masks ending up with only two values - 0 and 1.
6. **Conversion to .h5py:-** Supplying images to the model from system during training (using ImageDataGenerator) ends up consuming extra time. So for training the model on our system we have packaged all the images and masks files into two separate .h5py files and loaded them onto the RAM. Doing so speed up the training process.

3. NEURAL MODELLING

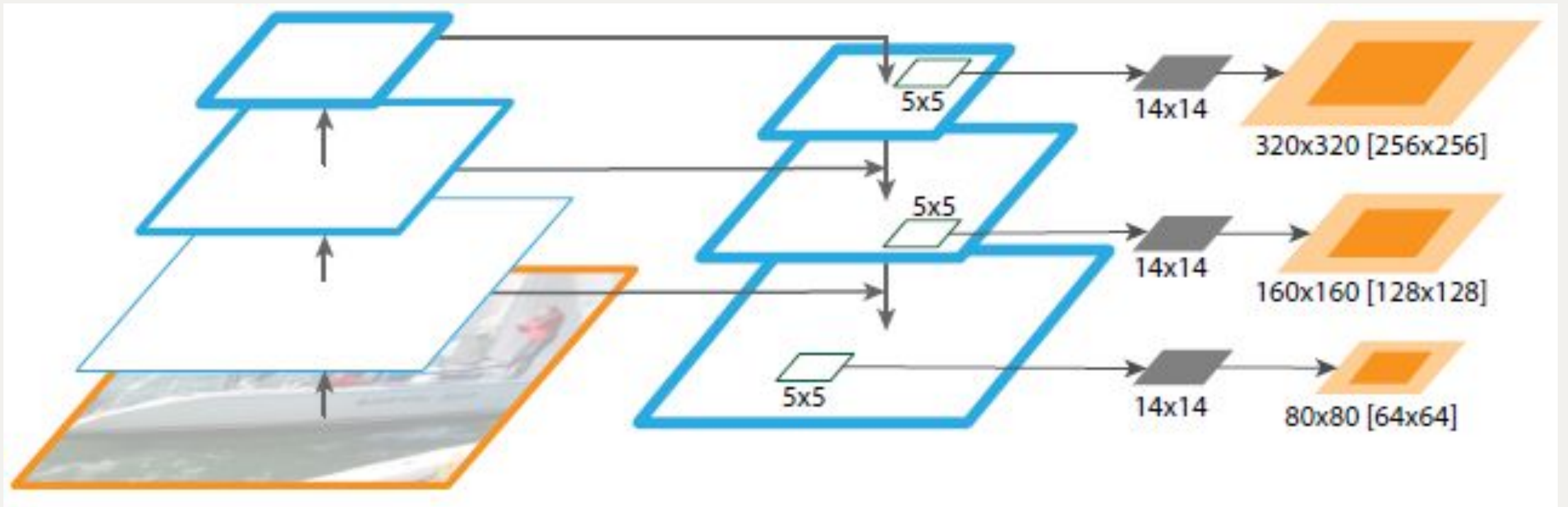
Fully Convolutional Network: U-net



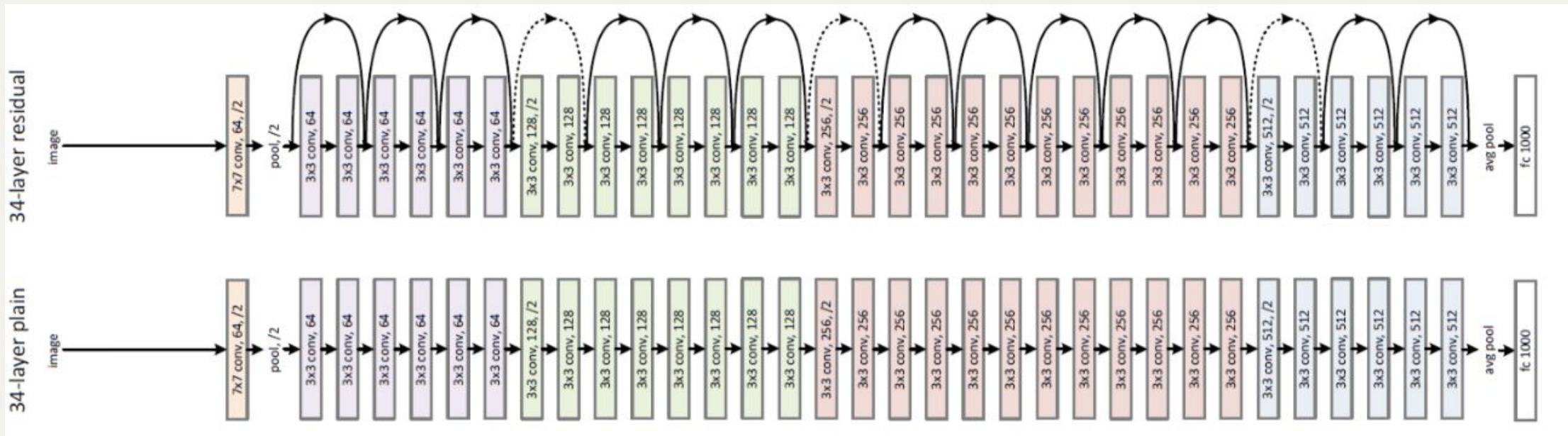
Pyramid Scene Parsing Network



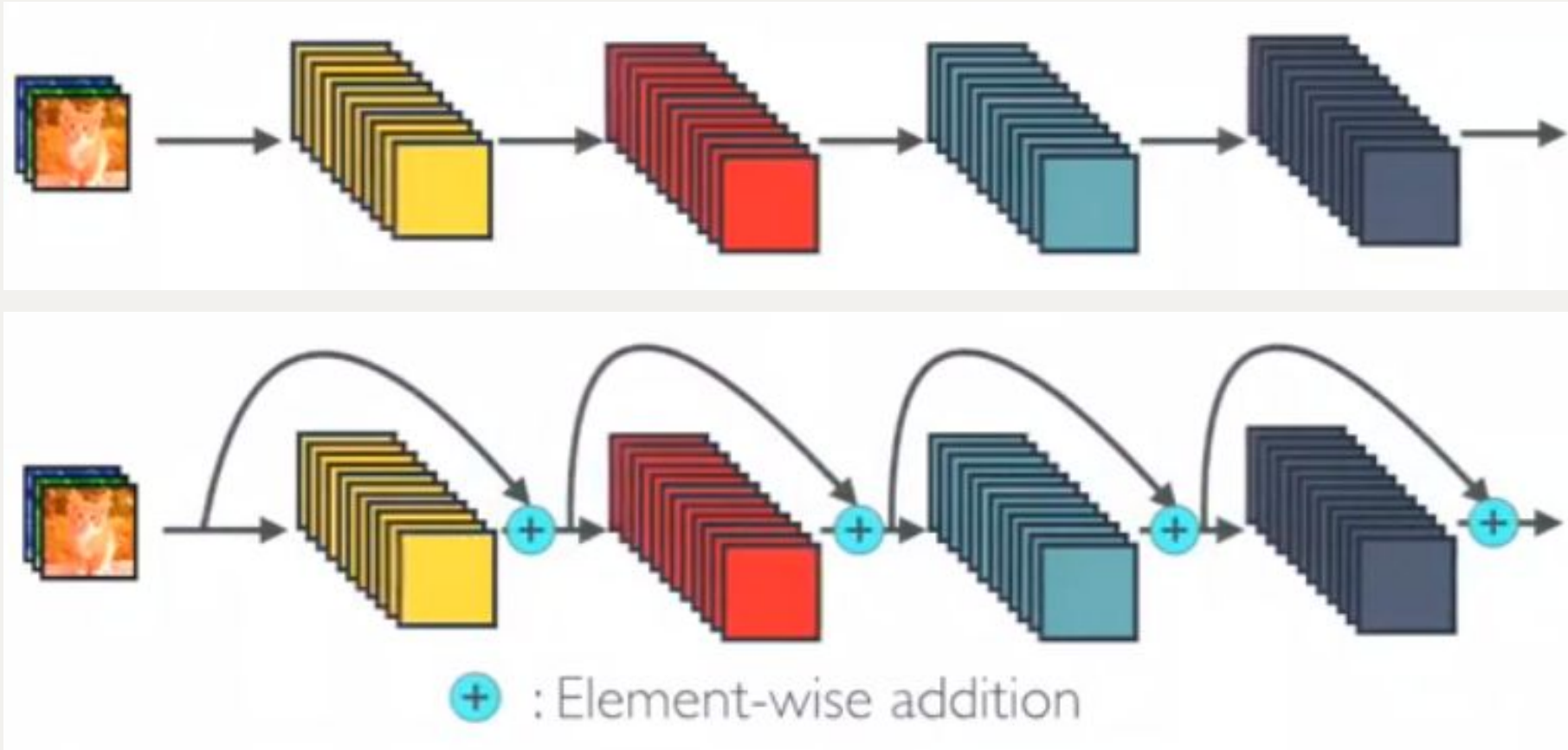
Feature Pyramid Network



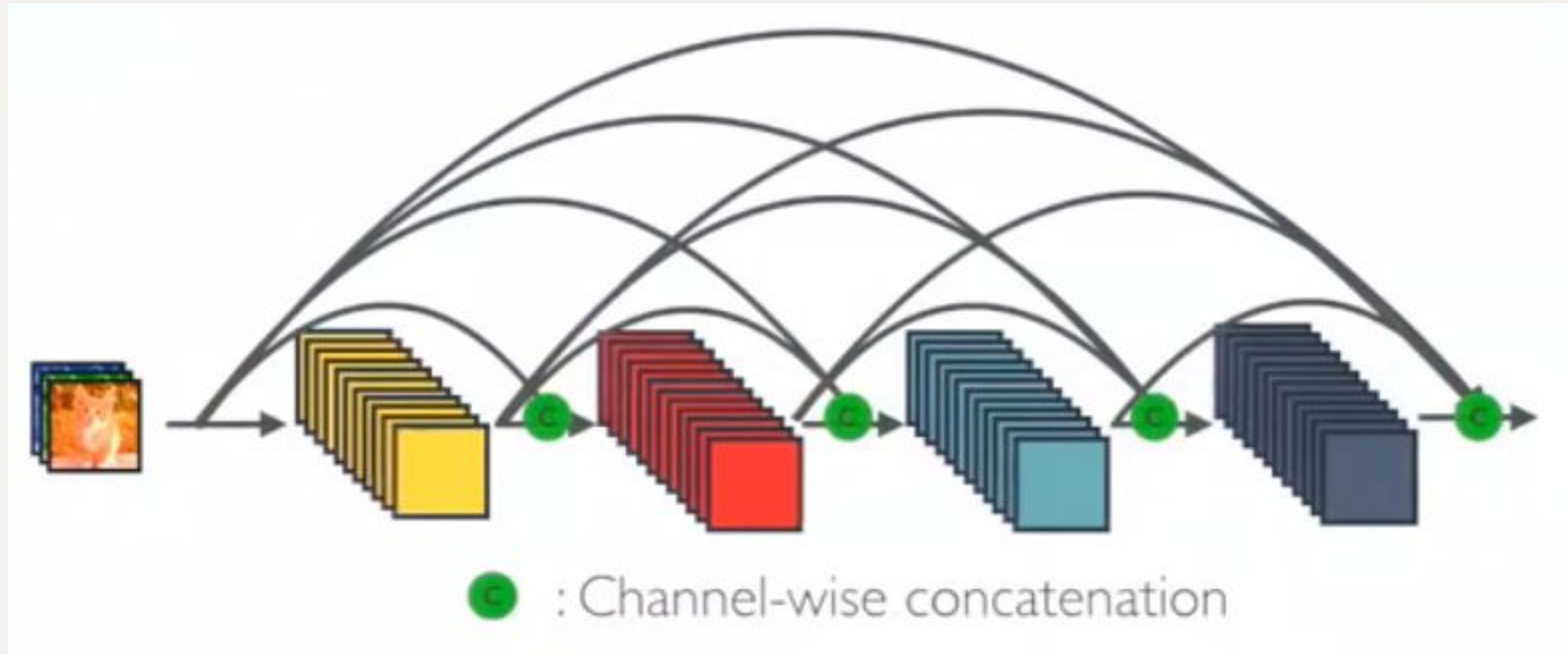
Residual Network (ResNet)



Densely Connected Convolutional Networks



Densely Connected Convolutional Networks



Why U-net ?

- Pooling or strided convolutions is detrimental to for semantic segmentation as spatial information is lost which is avoided in U-net.
- 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.
- Such a network can be trained end-to-end from very few images. Moreover, the network is fast. Segmentation of a 512x512 image takes less than a second on a recent GPU.

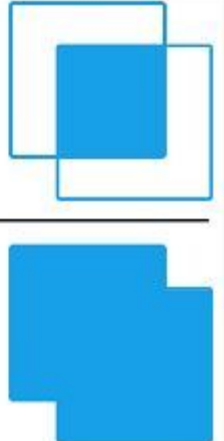
Loss Function

- 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).
- Dice Coefficient is the measure of overlap between the predicted sample and the ground truth sample, and this value ranges between 0 and 1. Where 0 represents no overlap and 1 represents complete overlap.

$$\frac{2 * |X \cap Y|}{|X| + |Y|}$$

Loss Function and Accuracy Metric

- Smooth Dice Loss is simply $1 - \text{Dice Coefficient}$, this is done to create a minimizable Loss Function. [1]
- By adding smooth (default value 1) to both the numerator and denominator, we ensure that a division by zero never occurs.
- We will be using the Jaccard Index, aka Intersection over Union, to tell us how accurate the generated maps are.
- As the name suggests, Intersection over Union is the measure of the correctness of the segmentation maps.


$$\text{IoU} = \frac{\text{Area of Overlap}}{\text{Area of Union}}$$

Optimizers and Hyperparameters

- The model was compiled using Adam as the optimizer.
 - Combination of AdaGrad update and Momentum update and gives better performance.
- Initial learning rate was set to 0.00001
 - Found out by Babysitting the Learning Process.
- Dropout with a rate of 0.1
 - To prevent overfitting.
- Loss Function:- Soft Dice Loss
 - Cross Entropy loss function can cause problems if the classes have unbalanced representation.

Optimizers and Hyperparameters

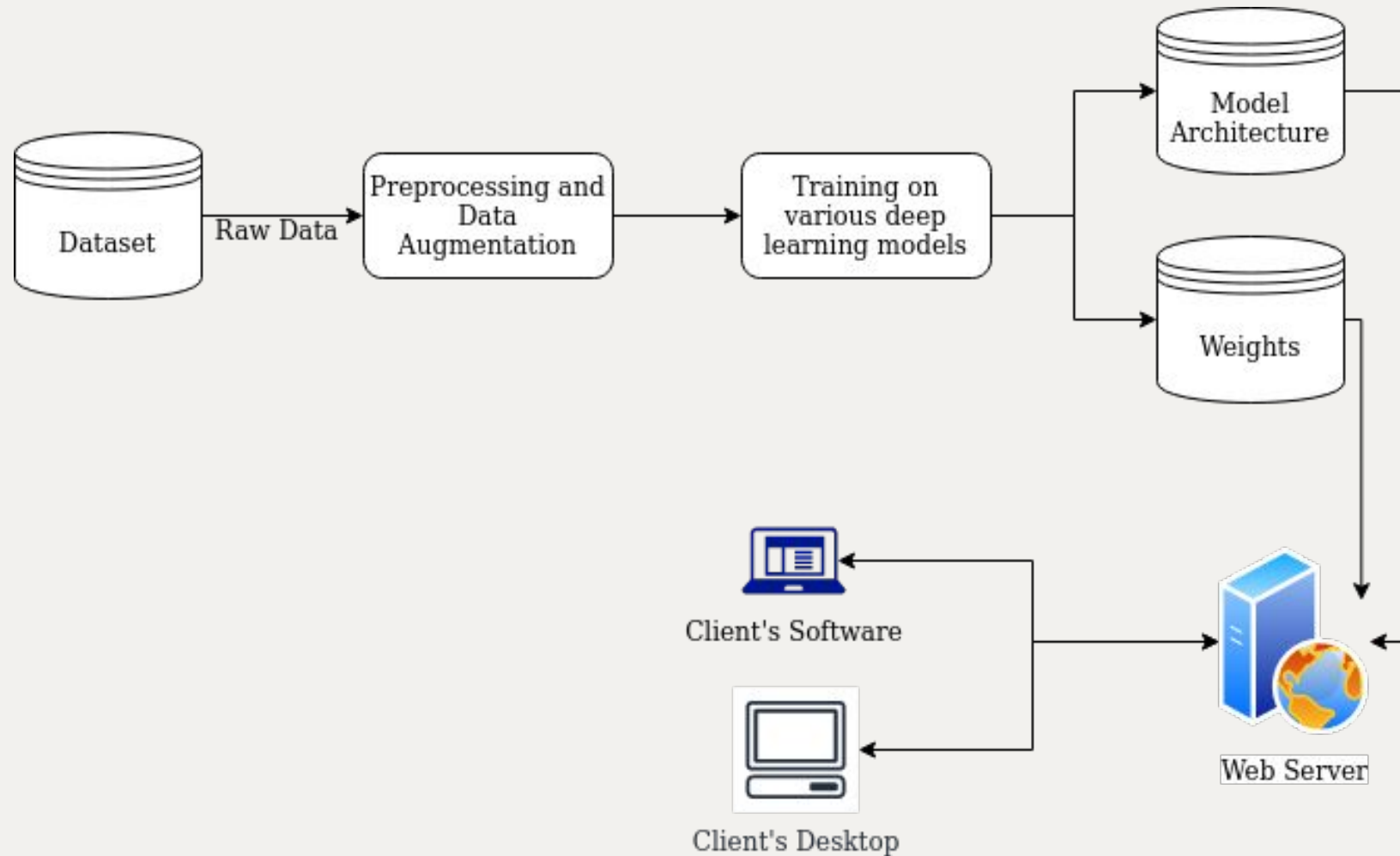
- Accuracy metric:- Jaccard Index
 - Pixel accuracy cannot perform under class imbalance problems (especially when our classes are extremely imbalanced).
- Activation Functions used were ELU and Sigmoid
 - We chose ELU over ReLU as it provides us all benefits of ReLU, it doesn't die and provides output with means closer to zero.
- Initializer used was He Normal (He-et-al) ^[2]
 - 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.

Callbacks

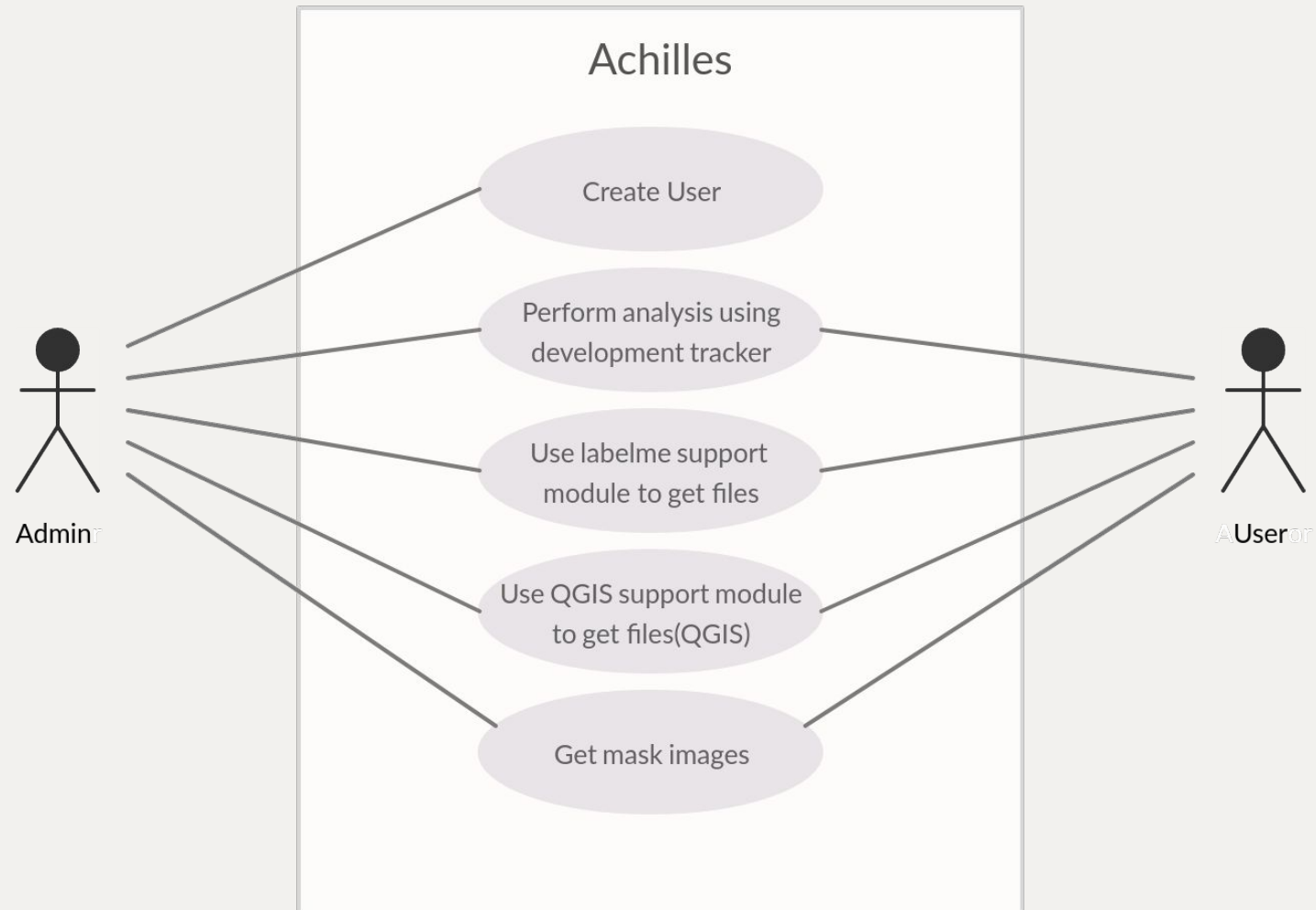
1. **ModelCheckpoint:** Monitors validation loss and saves the weights of the model with the lowest validation loss.
2. **EarlyStopping:** Monitors the validation loss and kills the training process if the validation loss does not increase after a certain number of epochs.
3. **ReduceLROnPlateau:** Monitors Validation loss and reduces the learning rate if the validation loss doesn't go lower after a certain number of epochs.
4. **TensorBoard Colab:** is a special version of Tensorboard tailored to work on Google Colab. We can monitor the accuracy and other metrics during the training process.

4. SYSTEM ANALYSIS & DESIGN

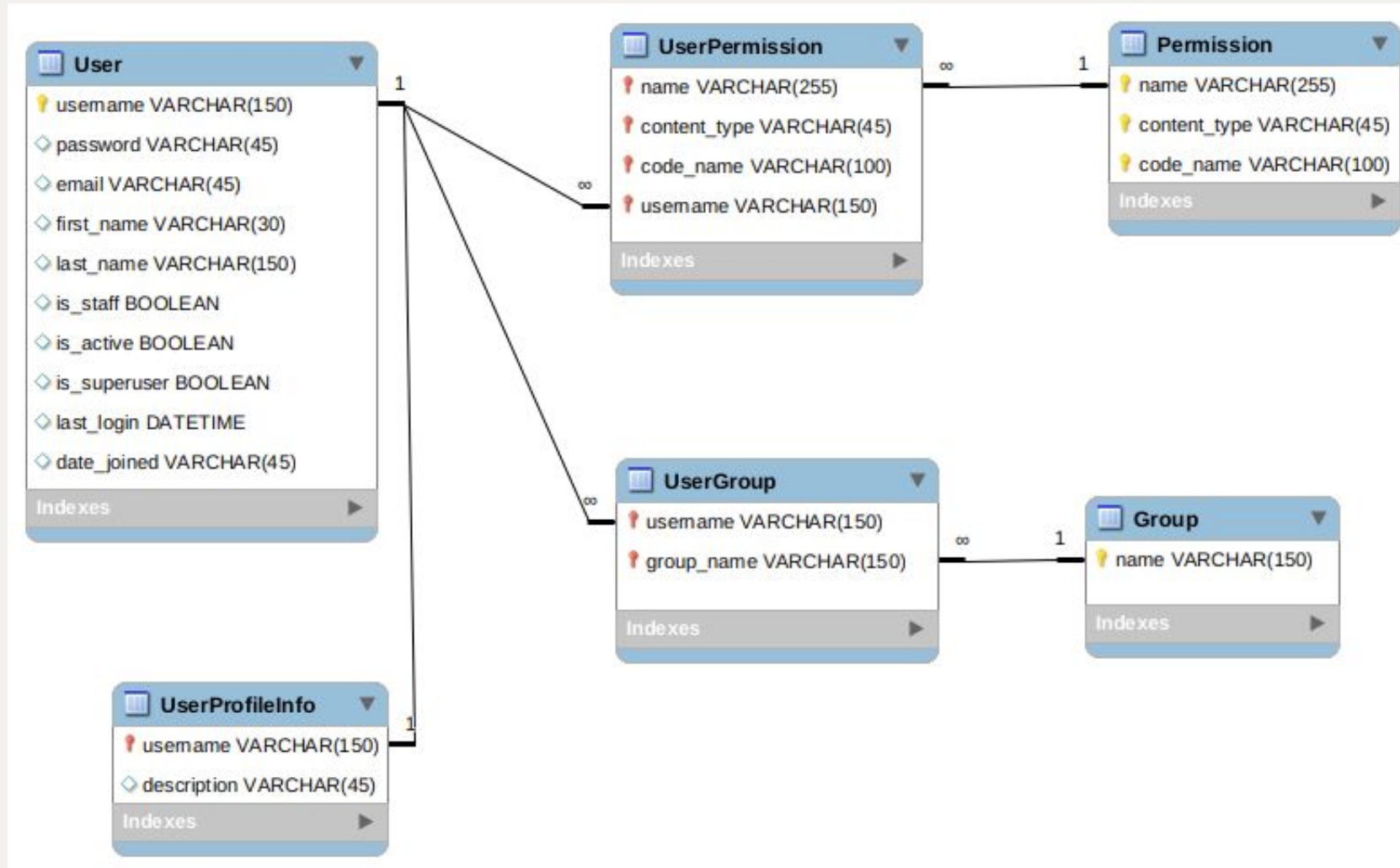
Project Flow Diagram



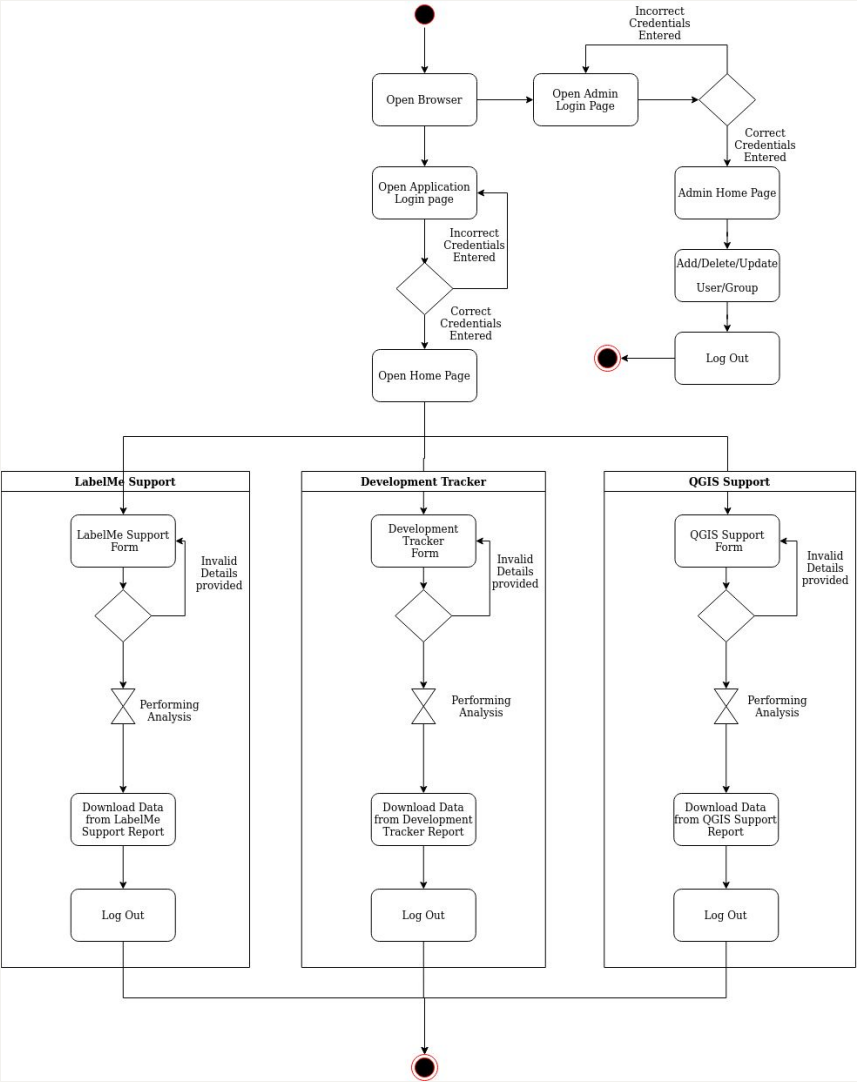
Use Case Diagram



Entity-Relationship Diagram



Activity Diagram



Testing Process

TEST CASE

Test Case ID	T-07	Test Case Description	Test the LabelMe Support Form		
Created By	Krupali	Reviewed By	Parshwa		

Tester's Name	Krupali	Date Tested	07-04-2020	Test Case (Pass/Fail)	Pass
---------------	---------	-------------	------------	-----------------------	------

S #	Prerequisites:
1	Access to Chrome Browser
2	Access to Firefox Browser
3	
4	

S #	Test Data
1	Can we select a class
2	Can we upload an image
3	
4	

Test Scenario	Verify whether can we access the LabelMe Support website
---------------	--

Step #	Step Details	Expected Results	Actual Results	Pass / Fail / Not executed / Suspended
1	Navigate to http://127.0.0.1:8000/lab-el/development_tracker/	Site should open	As Expected	Pass
2	Enter Form	Details can be entered	As Expected	Pass
3	Click Upload	Data is uploaded	As Expected	Pass
4	Navigate to http://127.0.0.1:8000/lab-el/development_tracker_response/	Site should open with correct output and contents can be downloaded as expected	As Expected	Pass

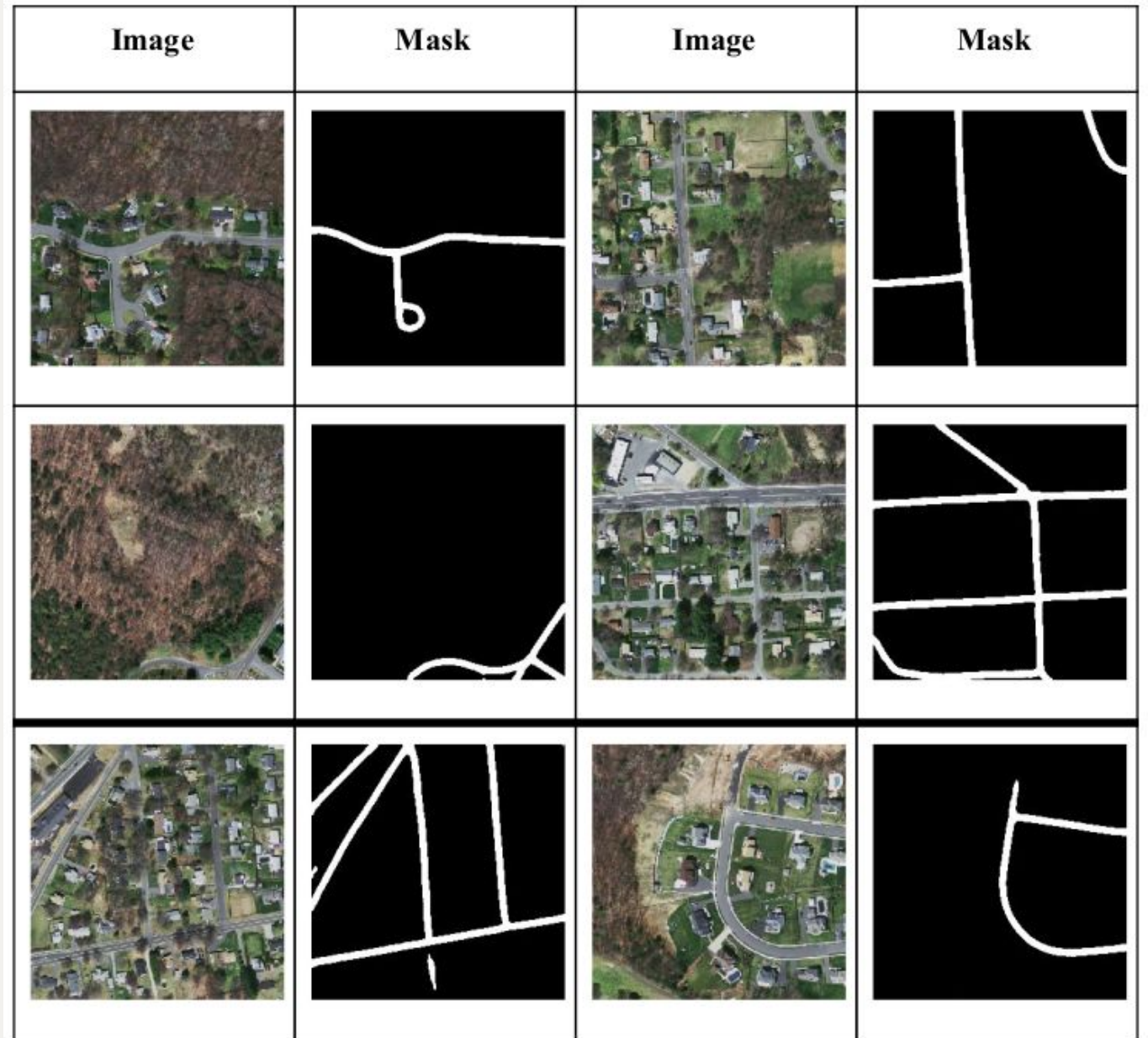
5. RESULTS

Model Performance

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%

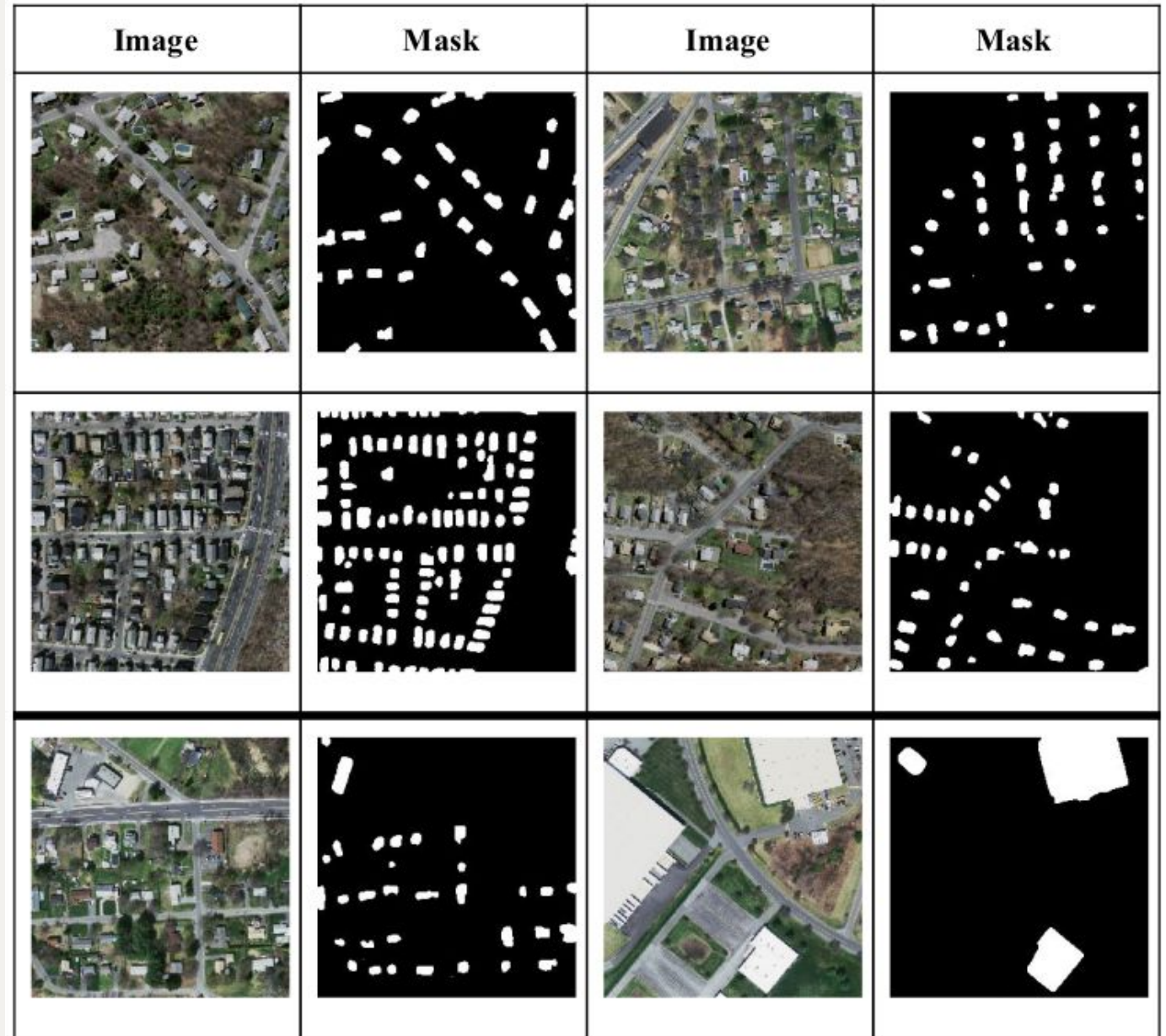
Road Segmentation

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



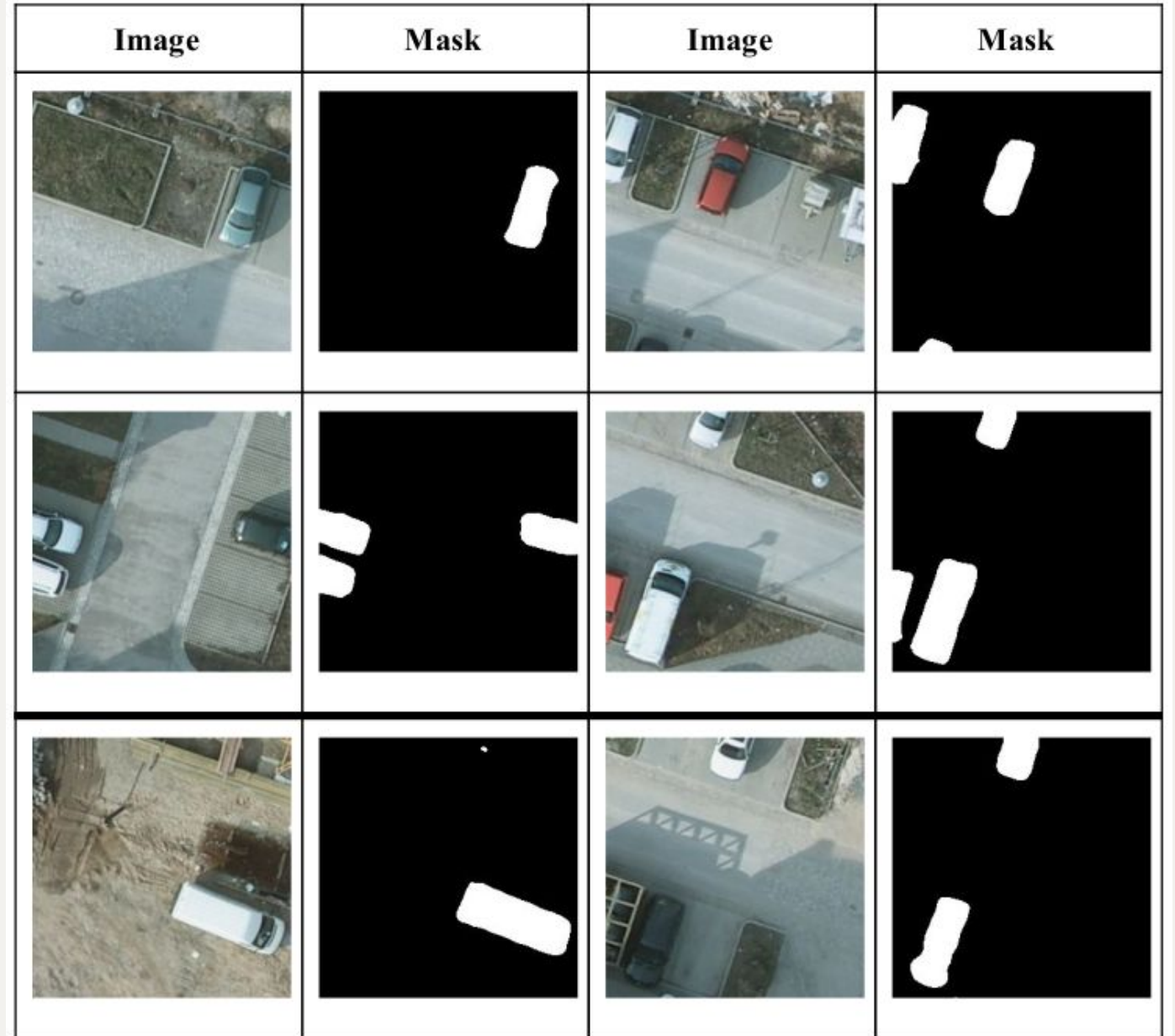
Building Segmentation

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



Car Segmentation

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



Activities Firefox Web Browser Mon 14:54 Achilles - Mozilla Firefox

Achilles

127.0.0.1:8000/label/

Most Visited Gmail YouTube Google Colab W Wikipiedia CS231n Github HackerRank Courses Exploit-DB Aircrack-ng

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 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.

Thank You