

Satellite Image Segmentation using Deep Learning Techniques

Shwetarani Shwetarani

Department of Applied Data Science, San Jose State University

DATA 270: Data Analytics Processes

Dr. Edurado Chan

December 10, 2021

Abstract

In this article we describe an end to end process of implementing satellite segmentation applications using deep learning techniques. We implement both binary and multiclass classification of the segmented images. The project describes the datasets needed for the application and the sources from where the datasets were collected from like Kaggle and public databases like Inria and Spacenet. Once the dataset is collected, it is required to filter the unwanted samples and perform initial cleanup. Since the dataset is mainly images and the masks, the cleanup process involves image resizing and data type conversions. Several augmentation techniques are used on the cleaned-up images to introduce variances of the model during training. The normalized images are then used to train three deep learning models independently. The models selected were Fully Convolutional Neural Network, UNET and LinkNet. The performance of these models were compared based on several loss and accuracy metrics and the best model was selected for the inference in the final deployment. U-NET is performing better than other modules with an accuracy score of 0.96.

Satellite image segmentation has a lot of applications. Automating the satellite image segmentation process allows us to offload a lot of manual work to the computers and this allows us to develop important applications like forest fire management, study global warming and other environmental changes. Apart from these there are various commercial applications like roof area analysis for the solar panel industry, live traffic monitoring and efficient re-routing in software maps used in navigation.

Keywords: Satellite, Segmentation, Deep Learning, Datasets, Fully Convolutional Neural Network, UNET, LinkNet, Augmentation.

1. Introduction

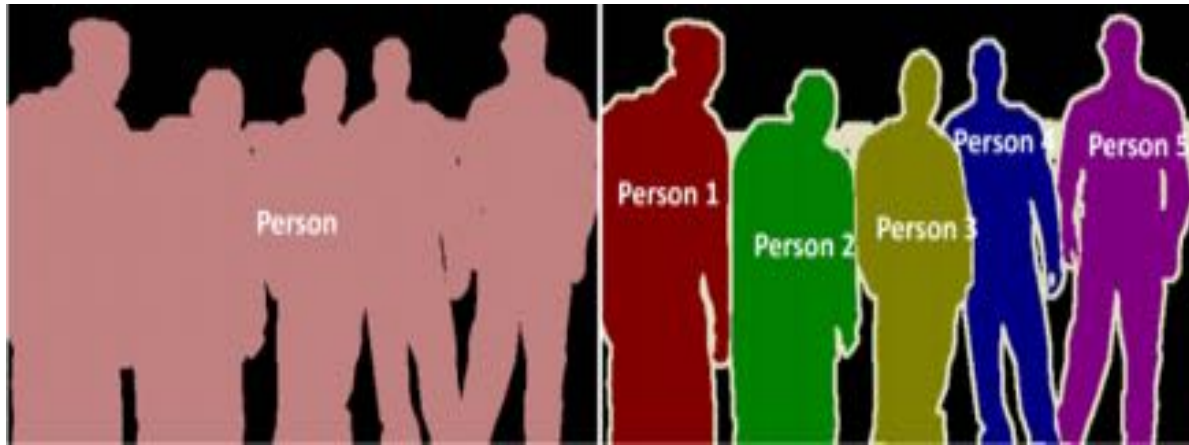
1.1 Project Background and Executive Summary

Whenever we see a painting of nature, our brain can identify what that painting contains (Waterfall, Mountains, and Trees). Nowadays it is possible to do the same thing for machines. The development in Computer Vision has changed the way we process images. Image is nothing but a collection of different pixels. There is an inherent pattern stored in the image depending on the objects present in an image. Those inherent patterns are in turn derived from the features present in an image. Computer vision uses Convolutional Neural Networks (CNN) to recognize the features in an image. Object Detection (OD) and Segmentation are the applications of Computer Vision. Object Detection detects objects of different classes and forms a bounding box to each class in an image. Object Detection is agnostic to the shape of the object in an image.

Image Segmentation is a category of Computer Vision, Image Segmentation will group together all the pixels which have similar attributes and mask them with different colors (Sharma, 2021). Segmentation is sensitive to the shape of the object in an image. There are two types of Segmentations, Semantic Segmentation and Instance Segmentation. Semantic Segmentation is the process of identifying and generating a mask of different classes of the objects in the image. As shown in Figure 1, the model identifies a class person in pink and the background in black. Instance segmentation goes one step further and separates instances within a class. As shown in Figure 1, the model identifies instances of a class of people of different colors and labels them as person 1, person 2, person3, and so on (Sharma, 2021).

Figure 1

An Example of Semantic and Instance Segmentation



Note. Figure shows Semantic and Instance image segmentation. Adapted from *Image*

Segmentation | Types Of Image Segmentation by Sharma, P. (2021).

(<https://www.analyticsvidhya.com/blog/2019/04/introduction-image-segmentation-techniques-python/>). Copyright 2021 by Analytics Vidhya.

1.1.1 Motivation and Needs of Satellite Image Segmentation

Some of the advantages of image segmentation are locating objects in satellite images, Traffic Control Systems, Self-driving cars, and biomedical image segmentation (cancer cell detection). In this project, we are considering one of the advantages of image segmentation, locating objects in satellite images using deep learning models.

Satellite image segmentation plays a huge role in automatically detecting sudden changes in the environment for example if there is a fire in a remote area real-time satellite image segmentation can be used to detect fires and notify, analyze traffics, monitor environmental changes, find how much open roof space is available for the solar panel industry, landscape analysis like urban vs rural, and also forest landscapes. Satellite image segmentation can be used for research purposes like oceanography, effects of global warming such as melting ice in the

Arctic, and the Antarctic. Satellite images cover a vast geographical area. It generates a lot of data, a number of images generated within the area, and also images are generated in various resolutions.

Analyzing and making sense of these images with the human involved will be very tedious and exhaustive. So this project is about automating some of those tasks or processes using image segmentation techniques with deep learning models. This project is not about an end to end automation using Deep Learning models but instead can be treated as the first level of automation where it can do Semantic Segmentation of the satellite images and classify them as Roads, Trees, Buildings, Ocean, and so on. This automation will help to reduce some of the manual work.

1.1.2 Objective and Approach

The main goal or objective of this project is to classify the objects in the satellite images by generating the masks for those objects. As shown in Figure 2, we are segmenting and assigning different colors to each class in the satellite image (Buildings, Roads, Trees, Crops, and so on) (Nowaczynski, 2021). Our model should be able to segment both binary and multiclass satellite images. The number of mask categories is the same as the number of objects detected in the given model. A simple binary classifier would classify only two classes of objects. The training datasets will consist of images and labels. The labels are nothing but images with only the masks for the objects that need to be recognized. All other objects that do not need to be recognized will fall into a single category of the mask (background mask or class).

Solving the Satellite Image Segmentation problem using semantic segmentation technique where satellite images are segmented by classifying each pixel of the object. To do this

process, the project uses some of the state-of-the-art deep learning models like Fully Convolutional Networks (FCN), U-NET, and LinkNet.

Figure 2

Images illustrate what satellite image segmentation looks like after building a model.



Note. Image shows before and after segmentation results of satellite images. Adapted from *Cookie and Privacy Settings. Deepsense.Ai* by Nowaczynski, A. (2021).

(<https://deepsense.ai/deep-learning-for-satellite-imagery-via-image-segmentation>). Copyright 2021 by Deepsenseby.AI

1.2 Project Requirements

1.2.1 Dataset requirement

After deciding the project objective, we decided to collect satellite images. While researching satellite image data we found several existing resources which have both image and labelled data. So we have decided to make use of those existing resources like Kaggle datasets containing Semantic Segmentation Of Aerial Images, Satellite Images Of Water Bodies, and Satellite Building Semantic Segmentation images. Second existing sources are publicly available

Databases (Inria and SpaceNet). The Inria database contains satellite images of buildings from various cities. The SpaceNet database contains images of buildings and roads networks. These existing resources collected satellite images from Sentinel-2, Worldview-2, and Landsat 8 satellites. Additionally collecting some satellite images by using MapBox API for open street map (OSM). OSM is an open-source mapping platform where we can add features like trees or roads by editing maps (“Mapbox Satellite”, n.d.). A Python API called DevelopmentSeed is used to scrape a dataset of satellite images. To make our models more generalized for multiclass classification, we also collected some raw satellite images from NASA earth Data. The images of these datasets are from different Geo locations so we are also thinking of classifying and categorizing satellite images based on their locations.

For Image annotation, some of our datasets already have pre generated images and their masks images and some do not. For those who do not have mask images, we are using the Groundwork tool for image annotation purposes. By using the Groundwork tool we can manually classify the image classes like Tree, Buildings, and Roads and assign different colors to each class to get the masked satellite images (“Data labeling, n.d.). We have decided to use RGB satellite images of size $256 * 256$ to train all the models. Satellite images in the collected datasets are of different sizes so we need some preprocessing steps to convert them to $256 * 256$ images. After collecting all the satellite images, we are manually testing and measuring every set and making sure our resources meet the data requirement of this project which is we should have satellite images of all class types (Tree, Buildings, Roads, Water, and so on).

1.2.2 Functional and AI-powered requirements (Deep learning models)

To train the models on satellite images, we first need to store those images. We are making use of Google Cloud Platform. We are storing and organizing the Satellite images by

using Google Storage buckets and Bigquery service. We are using Deep Learning virtual machines and Google DataFlow for preprocessing and modeling tasks. Our whole project is done on the python programming language by using a platform called jupyter Lab. In this project, we are using Deep Learning Models like FCN, U-NET, and LinkNet. These models are tested and measured based on some evaluation metrics like Accuracy, F1 score, IOU, Confusion Matrix and so on. The AI models used to achieve our project goal are listed below:

- FCN (Fully Convolutional Neural Network), is a fully convolutional network but without skip connections.
- U-NET, is similar to FCN and also a variation of a fully convolutional network with skip connection from encoder convolutional to decoder convolutions.
- LinkNet is similar to U-NET and we can train the model without increasing the significant number of parameters.

1.3 Project Deliverables

The deliverables of this project includes detailed quality and quantity measures to achieve the project goal. The deliverables of this project will be a detailed description of the research paper on the satellite image segmentation problem and reporting of the research paper which includes measurable performance results of the models proposed for the problem. After doing an existing literature survey about the satellite image segmentation problem, we have decided to collect satellite images from various existing sources and also collect raw data to make our models more generalized so that our models are able to segment multiclass objects in the image (Forest, Water, Land, and so on). These resources have collected satellite images from Sentinel-2, Worldview-2, and Landsat 8 satellites. We are also providing some of the sample collected datasets as deliverables of this project. These collected satellite images are

preprocessed before training our Deep Learning models to achieve semantic satellite image segmentation.

Following this approach, our project includes multiple Deep Learning Models. The performance of these models are compared based on the different evaluation metrics like Precision, Confusion Matrix, Accuracy Score, F1 Score, Recall, ROC - AOC curve, Sorensen-Dice coefficient (DSC), and Intersection Over Union (IOU). These performance measures of each model are compared with other models' performance scores. Based on these comparison scores we are selected the best model to solve satellite image segmentation problems.

To give Visual representation of the Data Preprocessing and Models Evaluation results, in this project we are using several bar plots , line plots, and some image plots. These plots show distribution of multiclassses present in the images, statistics of satellite images, the results of data transformation and different bands in the image datasets, and the performance results comparison of different models. Finally, Prototype with UI, Simple binary segmentation classifier using best model implemented on TensorFlow or Pytorch. The deliverables of this project also includes PowerPoint Presentation slides, The detailed Coding Files which are organized based on Data Preprocessing and Modeling steps, and the Report which includes sample data, Visualizations, and Evaluation snippets of this project.

1.4 Technology and Solution Survey

Several Machine Learning algorithms are proposed to solve the Satellite Image Segmentation problem. Some papers have used traditional ML models to segment the satellite images and some recent papers are based on Deep Learning algorithms. In this phase we are

explaining the solution survey of different technologies to solve the targeted problem and also comparing the proposed model in this project with the existing solutions.

Deep Learning technologies are far more advanced nowadays and are the state of the art models for image segmentation and also in general for computer vision tasks. Before DNN came into existence, segmentation or classification methods based on feature extraction from the images were popular. The feature is a form of information that captures a unique aspect of the image. Set of features are then used in solving computational tasks in computer vision and image processing. Different types of features were used in image segmentation like Pixel color, Local Binary Patterns (LBP), Histogram of Oriented Gradients (HOG), and so on (Xiaolong Liu¹ et al., 2018). In most of the traditionally used machine learning models, a great amount of model performance depended on manual feature extraction.

Apart from feature-based classification, a more popular traditional approach used in image segmentation was unsupervised clustering methods like K-means clustering. Initially, k centroids are located randomly, then the pixels are clustered using the K centroids by attributing them to the nearest centroid. The number of centroids is also equal to the number of classes used in the segmentation. For image segmentation, initially feature space allocates some cluster of pixels with its centers. In the first step each pixel is allocated to the nearest clusters. In the next step, the new clusters are computed. Until convergence these above steps are repeatedly processed. The basis of segmentation in K-means clustering is pixel intensities. Each pixel randomly assigns to some random class and another pixel reiterates so that the center of different classes is recalculated and again pixel groups affected to the new classes on the basis of nearest centroid. This process continuously repeats until convergence occurs. After doing the K-means process for segmentation and classification the results will be stored for each image. Although

the K-means algorithm gives a good accuracy score for satellite image segmentation, the disadvantage is we can only train the same type of data at a time by using K-means (Sathya & Malathi, 2011).

For classification and segmentation of Land cover satellite images, Support Vector Machine (SVM) is used. SVM is a supervised machine learning technique. Support Vector Machine (SVM) is a linear model which is used for both regression and classification tasks. SVM performs classification tasks by drawing a hyperplane so that one type of the categories (classes) are on one side and other types of categories are on the other sides. There could be multiple hyperplanes, SVM tries to find the one that best separates the two categories. The distance between the closest data points and the hyperplane is called the margin distance and the points on that margin are called support vectors. By using this hyperplane SVM finds the correct class of the training sets (Bernard et al., 2012). The author Bernard et al. (2012), used SVM classifier for pixel wise image classification based on classification results of the segmentation of the satellite images performed by using clustering. Then they combine these results based on majority votes. By using clustering methods in segmentation they are successfully able to define the adaptive neighbor for every pixel in the image. The author was able to segment the remote sensing images with good performance by using the SVM algorithm. The disadvantage of this method is that small spatial structures are wrongly segmented to the different class while considering spatial information from the nearest neighbors in a classifier.

Random Forest Classifier method also used for segmentation and classification of satellite images. Random Forest classifier is a collection of multiple random trees and much less sensitive to the training data. RF is an ensemble classification and advanced version of the bagging method. Random Forest method is used to create a new training dataset with choosing

important features from the original dataset (Akar & Güngör, 2012). Random Forest technology is very robust and is against the model overfitting because of its grown tree never pruned characteristic. Initially the N and M parameters are set to the RF algorithm, where N is the number of nodes to split and M is the number of variables to split each node. The tree split is stopped when the Gini index value becomes zero. Random Forest method also works good for image classification and segmentation, if we give N value as 1000 the RF forms 1000 different classes for each pixel in the image. For instance if that particular pixel is classified 900 trees as Forest and 100 trees as Water then the classifier chooses the Forest as the class of that pixel based on majority votes. RF classifiers give better performance scores than the SVM method but the disadvantage of RF is the performance of the classifier totally depends on the user defined parameters like N and M (Akar & Güngör, 2012).

Recently the era of Deep Learning models has started. The deep learning models on the other hand do a much better job in automatically extracting the abstract features from the images and inherently handle most of the computer vision tasks (Yusuf Artan, 2011). The author Long et al. (2015), proposed an architecture called Fully Convolutional Network (FCN). FCN is also a Convolutional Neural Network but instead of a dense network it has samplers. FCN is an encoder decoder type architecture and can handle arbitrary sized images. It has Convolutions, ReLU activation, and max Pooling process. The main advantage of these Convolutional models is that extraction of image features is automatic. The encoder network projects a lower dimension of the image and extracts features from it. The decoder will reconstruct the image again to give a segmented image as output (Yoshihara et al., 2018). The papers proposed on this method got around 90% accuracy result for satellite image segmentation.

Encoder-Decoder CNN architecture called SegNet. The SegNet is a significantly smaller model compared to FCNs and has a memory/computation advantage over the FCNs and other state-of-the-art models. The model looks very similar to FCN but uses the max-pooling indices to upsample the images in the decoder stages instead of learning to upsample in the FCNs (Vijay Badrinarayanan et al, 2016).

U-NET is another Convolution method, which is also a Fully Convolutional Network. The author Ronneberger et al. (2015), originally proposed U-NET architecture for the purpose of image segmentation in medical applications. U-NET architecture consists of Convolutions, ReLU activation, and max Pooling process in addition to it has skip connections. The main advantage of this network is skip connections between the encoder and decoder network. These skip connections help to get localization information that makes semantic segmentation accurate without losing any information of the input image in U-NET. Another advantage of U-NET architecture is that we can train this model by using less training data (Chhor et al., 2017). The papers who solved satellite image segmentation problems by using this method achieved an accuracy score of 96%.

Author Chaurasia and Culurciello (2017), proposed another convolutional method called LinkNet. LinkNet also works similar to the U-NET architecture but there are only four encoder and decoder blocks on the LinkNet architecture. The main advantage of LinkNet architecture is that we can train the model without increasing the significant number of parameters. Because of this advantage we can achieve segmentation processes in real time by using the LinkNet model. Satellite image segmentation problem was solved by using this method and resulted in 93% accuracy.

As a result, Deep Learning models have much higher performance and are easy to implement compared to the older solutions where we used to extract features of images manually which is actually time taking and tough to solve. So in this project we have decided to use Deep Learning methods like FCN, U-NET, and LinkNet for semantic segmentation of the satellite images. The reason behind choosing Deep Learning models are these models follow automatic feature extraction technique, time efficient while training the models, they can handle all type of datasets so that we can multi classify images and do segmentation on them, and the input image size is arbitrary not fixed size for these Deep Learning Models.

1.5 Literature survey of existing Research

Existing research papers tell about the model used for solving the problem and the real-world applications of performing segmentation on satellite images. Detail about the state of the art model that is widely used for image segmentation. Some of them talk about other state-of-the-art models that could be used for image segmentation. Potential applications of automatically detecting buildings from satellite images such as monitoring movements of populations and finding available space for implanting solar panels on roofs. There are many papers which talk about satellite image segmentation by using Deep Learning Models. They develop many versions of Convolutional Neural Networks like FCN, FCN8, FCN16, U-NET, and LinkNet to segment the satellite images.

In 2016, there was a publication on semantic segmentation for satellites using Fully Convolutional Neural Network. The author Muruganandham (2016), trained the satellite images on a series of FCN models like FCN-no skip, FCN-8, FCN-16. He has analyzed the results of these three versions of FCN models in his paper. The author mainly concentrates on segmenting roads on the satellite images which is nothing but binary classification problem. To train these

models he has used two cities datasets one is from Massachusetts and another is from Prague city satellite images. For data preparation, the images are of size $1500 * 1500$ pixels are converted into $500 * 500$. They have used an augmentation data transformation process to increase the number of satellite images. For data annotation purposes, they have used the MapBox tool. They have used 4440, 294, 84 sets of images for training, validation and testing purposes. The author has used several performance metrics such as Confusion matrix, Recall, Precision, TPR, FPR, Intersection Over Union (IOU), and F1 score to measure the models (Muruganandham, 2016). FCN models with skip connections performed better than no skip in this paper. The author was able to achieve an IOU score of 88% and TPR of 79% with the models.

Using FCN-8 LULC Segmentation of RGB Satellite Image paper was proposed in 2020. The author Nayem et al. (2020) used RGB satellite images to train the FCN-8 model and they have attempted the multiclass segmentation problem in this paper. GID dataset contains 7168×6720 size 150 images which are converted into $224 * 224$ for training purposes. They also have ground truth images with this dataset. They have prepared data such as targeted class represented in blue color and other class is in red like this they prepared satellite images for all types of classes. They have followed nine augmentation techniques to increase the variations and the number of images since they only had 150 images in their dataset. They have split the dataset as training and testing sets, forest class has training sets of 25 and test sets of 6 images, farmland class has training sets of 119 and test sets of 12 images, and water class has training sets of 63 and test sets of 9 images (Nayem et al., 2020). The FCN-8 model trained on these images and uses the vgg-16 weights to segment the satellite images. To evaluate the results of each class the author has used evaluation metrics such as F1 score, IOU, and average Accuracy score. They have successfully achieved an average accuracy score of 91% and the IOU of 0.80 with the

FCN-8 model. Surprisingly their model was able to give better results compared to recognition results (Nayem et al., 2020).

Using U-NET, a paper was published for Building Detection on Aerial Images and LinkNet Neural Networks was proposed in the year 2019. U-NET was introduced for Biomedical image segmentation. This model cleared the large consensus where we thought that a successful deep network requires a thousand annotated training samples. This model uses available annotated samples. The first half of the network architecture consists of a contracting path that learns or captures the important features and on the second half, there is an expanding path that uses these learnt or captured features to precisely detect and localize the objects in the image. The author trained such networks end to end by using very few images. By using this architecture, they successfully achieved segmenting $512 * 512$ images which took them to run less than a second (Olaf Ronneberger et al., 2015).

The same concept was applied to a satellite image segmentation task in the paper. The author compared U-NET with the LinkNet model in his paper. To train the models they have used $1000 * 1000$ sized Planet and Inria database satellite images. The Inria database uses images from Sentinel-2 satellite. The Planet database uses images from the Worldview-8 satellite. The author focused on binary classification of building or not building in his paper. To measure the performance of these two models, the author used Accuracy score and Sorensen-Dice coefficient which gives quality of segmentation results. The U-NET model gives a 96.31% accuracy score and the LinkNet model gives an accuracy of 95%. They have achieved high accuracy with these two models to segment high resolution satellite images as buildings or not (Ivanovsky et al., 2019). The author considered U-NET as the best model compared to LinkNet in his paper because of its less training time with less number of input training images.

U-NET is considered the most efficient in terms of segmentation classification accuracy. U-NET consists of this skip connection encoder convolutional to decoder convolutions so, at each decoding stage, the input to the decoder consists of a corresponding encoded image appended with the output image of the previous decoder. As a result, the decoder convolution gets more information about the original image that is required to reconstruct or generate the segmentation mask. This also makes the model very efficient in a way that it requires very few images to fully train the model to get an expected accuracy.

The Author PRIIT ULMAS1 and INNAR LIIV (2020), proposed a paper to segment satellite images by using U-NET for Land Cover Classification. They have trained satellite images on the U-NET model. Automating the process of generating the land cover map by using a modified U-Net architecture. This is necessary to monitor environmental changes and also regional planning. The current process of generating land cover maps is very manual and labor-intensive. By using the advancement in satellite imaging technology and machine learning this process could be simplified and made more efficient (PRIIT ULMAS1 & INNAR LIIV, 2020). To achieve this the author used semantic segmentation technique in his paper. The author has used two datasets to train the model; one is the BigEarthNet dataset proposed recently in 2019. BigEarthNet dataset includes satellite images from Sentinel-2 satellite. The second one is the CORINE land cover map dataset which is also from Sentinel-2 satellite images of Estonia (PRIIT ULMAS1 & INNAR LIIV, 2020). In this project they are classifying and segmenting multiclass satellite images. For data preparation purposes they have neglected the cloud covered satellite images and selected good resolution ones to train the model. For training and testing sets, they split the datasets into 80 : 20 ratio of RGB images. For classification purposes, they have used ResNet and for segmentation they have used the U-NET method. To measure the

performance of models they have used metrics like Recall, Precision, Intersection Over Union (IOU) and F1 score. The author successfully achieved a F1 score of 0.749 for classification model and high IOU score of segmentation of multi class on satellite images such as forests, buildings, and water bodies (PRIIT ULMAS1 & INNAR LIIV, 2020).

1.5.1 Proposed Approach

After doing extensive research on the satellite image segmentation papers, To solve the semantic satellite image segmentation for multi classifications, we are using FCN, U-NET, and LinkNet models in this project. We have decided to use satellite images from existing data sources like Kaggle and databases (Inria and SpaceNet), In addition to this we are generating raw data from MapBox API and also collected some other data from NASA earth to make our model more generalized. We have done some data preprocessing steps based on exploratory data analysis (EDA) on satellite images like image resizing to RGB 256 * 256, delete unwanted satellite images from the repository, cloud cleaning on satellite images, data transformations like augmentation and normalization of the images. Our existing resources have already masked images but to create labels for raw satellite images we are using Groundwork tool for data annotation purpose. To test the datasets for training and testing, we are following two methods one is cross validation method and another one is region based splitting for instance Newyork satellite images go for training sets and Chicago images go for testing sets. The models are trained on these images and evaluated based on evaluation metrics like F1 score, Accuracy, Confusion matrix, Sorensen-Dice coefficient, IOU, and ROC curve. The highest performing model will be selected as a best model among FCN, U-NET, and LinkNet. The detailed explanation of our research is covered in upcoming chapters.

2. Data and Project Management Plan

2.1 Data Management Plan

Data management plan (DMP) depicts how much and what data we will collect or generate and how to manage the collected data. It should describe good strategies and methods to access, analyze and store the collected data and also depict the mechanism we are using to share and preserve that data throughout the whole research. Basically, DMP covers the life cycle of the data phases (“Data management plans”, n.d.).

Keeping the objective of the project in mind, we are collecting and generating satellite images from many different sources like Kaggle and online databases. From kaggle, three datasets are chosen for this project. The first dataset is, Semantic Segmentation Of Aerial Imagery dataset, it is an open-access satellite image dataset. The total volume of the dataset is 72 images grouped into six larger tiles. This dataset contains Road, Vegetation, Water, Building, Land, and unlabeled classes. This dataset is maintained by the dataset owner and the license for this dataset is CCO: Public Domain (“Semantic segmentation of aerial imagery, 2020).

The second dataset is the Satellite Images Of Water Bodies, which is created by Escobar (2020), this dataset uses Sentinel-2 satellite to capture the water body images. Black and white masks are generated by calculating NWDI (Normalized Water Difference Index). To detect and measure vegetation and water bodies NWDI is normally used, for greater threshold it is detected as water bodies otherwise it is vegetation (Escobar, 2020). The dataset is maintained by the dataset owner and the license for this dataset is CC By-NC-SA 4.0.

The final Kaggle dataset used in this project is Satellite Building Semantic Segmentation Data. The Dataset was created by Wang (2020) and it contains satellite images of buildings. This

dataset is maintained by the dataset owner and the license for this dataset is Attribution 4.0 international (CC By 4.0).

Apart from the kaggle datasets, there are several databases available for satellite aerial images. In this project, we are using the Inria database and the SpaceNet database. The Inria database was created by Maggiori et al. (2017), this database contains 180 color satellite images, having coverage of 810 km², which is further separated into 405 km² for training data and 405 km² for testing data. Images of this dataset have a resolution of 1000 * 1000 pixels where each pixel covers an area of 0.3m. The data is classified into the building and not-building segments. The dataset also contains images from various cities.

Conventional techniques used to split the datasets into training and testing subsets involve mostly random splitting. For this particular dataset, a slightly different approach is used by Maggiori et al. (2017), where the splitting of the dataset is based on geographical locations. For instance, the model is trained on the images of San Francisco city but it is evaluated and tested in other cities. The Author Maggiori et al. (2017) states this kind of splitting technique will allow for more generalization for the model.

According to Van Etten et al. (2018), the SpaceNet database is publicly available at the SpaceNet on Amazon Web Services. This dataset includes satellite images of six large cities. Worldview-2 and Worldview-3 satellites are used to collect Eight-channel photos with different spatial resolutions. The database contains subsets that are further separated by the type of images tagged. For example, the database comprises two subsets, one set is of 3011 km² and the second one is 5555 km² for building detection (Ivanovsky et al., 2019).

We are also collecting some satellite images by using MapBox API for open street maps (OSM). OSM is an open-source mapping platform where we can add features like trees or roads

by editing maps. A Python API called DevelopmentSeed is used to scrape a dataset of satellite images.

The datasets we have chosen for this project are open source in nature. The sources for the datasets are from Kaggle, the SpaceNet database and the Inria database. All the datasets are public and available for free download. Collectively, the total volume of the datasets is about five GB. The dataset mostly consists of various image sizes ranging from 1000 * 1000 to 256 * 256 but during data preparation, the idea is to scale or crop the images to 256* 256 standard image size. Even though some datasets are multichannel colored images, the preprocessing steps convert them to RGB images where each pixel is an eight-bit unsigned integer.

Datasets are managed by creating directories based on geographical types. The directory structure also takes into account the versions of the datasets. We implement techniques to check for the uniqueness of the data by defining the uniqueness score and threshold. The uniqueness check is nothing but the pixel-level comparison between any two images. If the score is lesser than the threshold we discard one of the two images, thereby filtering the datasets to contain only unique images. We also perform other kinds of sanity checks, like pixel value validation, basic image processing etc, to make sure the dataset contains only valid images.

For documentation and metadata, datasets are managed by creating directories based on the geographical types. Initially, the datasets are stored in Google cloud storage. We maintain the record in the big query database, with several attributes like image serial Id, corresponding label serial Id, image category it belongs to (like ocean, forest, and building), the label resolution of the image, and so on. Information like resolution, image Serial Id, label Serial Id, date of the image, who created the image, and the image category like the ocean, forest, and building could be used to query a subset of images or group them in any combinations. This provides a more

automated and systematic approach to store and access the images at any time during the project cycle or in future.

As far as the licensing and sharing is concerned, the Kaggle dataset has a license that provides consent for the share and copy the material in any format and adapt, transform and build upon the material for any purpose. The other two datasets are publicly available and open-source datasets.

Since we are storing the datasets in the Google cloud, the cloud services will be responsible for backup and recovery. Including the ML model and the datasets, the total required storage capacity will come to around 10 GB and as per Google cloud calculator, it will cost around two dollars per month to rent that much storage space over a six months period of time so total storage cost will be \$12. Security of the data is handled by Google cloud authentication and services which require us to create an account with a secure username and password to access the storage services. The storage space is basically a shared space in the cloud and can be accessed by the collaborators securely if the access is granted.

For the Kaggle datasets which are multicolored datasets, we would like to keep these datasets for future model updates and research purposes. Currently, we are using image size of $256 * 256$ pixels for training the model. For future research and study, we would like to preserve Inria datasets original images size, which is $1000 * 1000$ pixels, that will have a spatial resolution of $0.3m / \text{pixel}$. Default sharing method for other users would be to provide access to the Google storage space. If the sharing is across the network, the dataset should be compressed and serialized for efficient transmission.

After the project is completed with the final quality analysis and evaluation for both the training and inference model, the dataset, ML model and other software could be made available

for users by providing read access using the google cloud services so that they can not edit, update or mess around the data. The user should request access to the dataset by filling up a form with all the required details so that the user can be verified and tracked. Once the access form is reviewed and verified, we could allow the users to securely access or download the dataset.

Since this research is done by me, I will be responsible for implementing the Data management plan (DMP) and each data management activity like data capture, metadata production, storage/backup, data archiving, and data sharing. To deliver this plan, the Google cloud platform services and security policies are required.

2.2 Project Development Methodology

Models like Waterfall, Kanban Scrum, Research and Development (R&D), and Cross-Industry Standard Process for Data Mining (CRISP-DM) are popular project management methodologies. Scrum and Kanban are the subsets of agile methodologies, which are iterative and incremental development. The CRISP-DM model is based on both waterfall and agile methodologies. CRISP-DM Waterfall represents the horizontal slicing so the team must submit one final report. However, CRISP-DM Agile is based on vertical slicing where the team can go through all the phases and come back again to the bottom phase if any changes are required (“CRISP-DM”, 2021).

The advantages of the CRISP-DM model are, Generalize-able: Although it was developed for Data mining projects, most of the projects start with business understanding, collecting data, and preparing data. So we can use this for Data science projects also. Common sense: It identifies the phases and does several iterations, Adopt-able: We can create a CRISP-DM model without having skills or taking extra training for it. Right start, Strong finish,

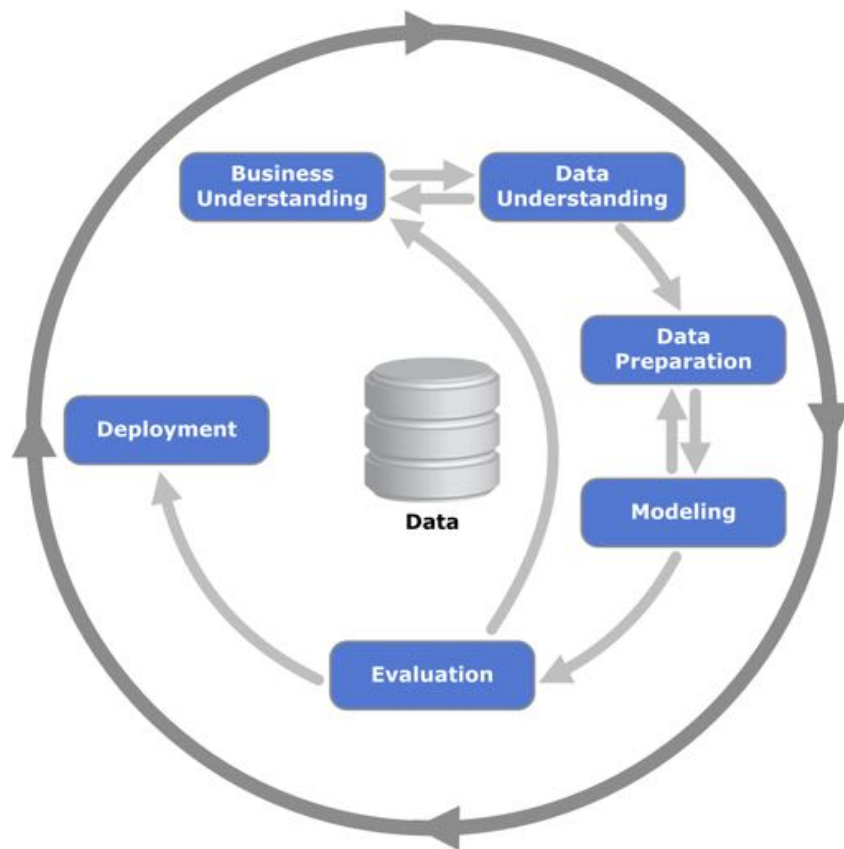
and Flexible: After finishing the project also we can cycle through the steps for changes (‘‘CRISP-DM, 2021).

Keeping all the advantages and its agile methodology in mind, this project follows the Cross-Industry Standard Process for Data Mining (CRISP-DM) model which was published in 1999. CRISP-DM is the robust and standard model for Data mining and Data science projects.

According to Wirth and Hipp (2000), It is a hierarchical model that has four levels of abstractions namely, Phases, Generic Tasks, Specialized Tasks, and Process Instances from general to specific. As shown in Figure 3, the lifecycle of a project is divided into six phases, which describes the life cycle of Data Science with having dependencies on the other phase’s outputs and the outer circle.

Figure 3

Different phases of the CRISP-DM model



Note. CRISP-DM model with six different phases. Adapted from “Integrating Crisp DM Methodology for a Business Using Tableau Visualization”, by A. Parate, 2020, (<http://dx.doi.org/10.13140/RG.2.2.36619.31520>). Copyright 2020 by ResearchGate.

2.2.1 Business Understanding

This phase answers the question, what does a business need? It determines the objectives and the requirements of the Project. According to Wirth and Hipp (2000), Business understanding is based on understanding the project objective and the requirements on the basis of business perspective. Keeping project objectives and business requirements in mind, creating an overall project plan to reach the goal.

In the Business Understanding phase, we are looking for the Satellite image segmentation project demands like automatically detecting sudden changes in the environment, landscape analysis, and research purposes like oceanography. After finding the demands, the next step is to determine the goal of the project which is to classify the objects by generating the masks for those objects by using deep learning models. Finally, analyzing how to measure project progress.

2.2.2 Data Understanding

This phase deals with identifying and collecting the data to achieve a project goal. After identifying and collecting the data, the main phase is to understand the data clearly and answer whether the data is relevant to the project requirement or not. Once the data is decided completely we have the task to store that data for this we need to decide the hardware to locate it. Based on the business requirements we have to decide risk assessments like unavailability and accuracy of the data. As shown in Figure 3, Business Requirement and Data Understanding phases are interconnected which means data understanding and collecting the specific type of data is totally dependent on Business understanding.

Collecting satellite images using Kaggle datasets, SpaceNet and Inria databases, a python API called development seed is used to scrap a dataset from satellite images, and also using Mapbox API for open street Map (OSM) for the purpose of collecting data. Once the data is collected and stored, the dataset is classified based on landscape data (urban landscape with buildings and Forest landscape data) and ocean images. Finally, analyzing the risks like real time satellite images unavailability and issues with image resolution.

2.2.3 Data preparation

After collecting the data from different resources, our next job is to prepare that data according to our project goal. This phase includes the preparations like cleaning, construct, format, storing, and splitting that data into a train, testing, and validating data.

First of all, we are cleaning the data by selecting specific satellite images and conducting image preprocessing like scaling, cropping, and normalization. Secondly, store the cleaned data using google cloud storage which includes serialization, deserialization, and image compression and restoration techniques. Finally, splitting the data into train, test, and validate data for further process.

2.2.4 Modeling

Once the data is prepared, our next step is to select models based on different modeling techniques. Software like language and framework requirements for them are to be decided and installed based on the selected models and create those models.

As shown in Figure 3, we can see there is a link between the Data preparation and Models phases, which means while creating a model if we realize any data problems, we can go to the Data preparation phase and refix it (Wirth & Hipp, 2000).

We are using UNET (Ronneberger, et al., 2015), FCN (Long, et al., 2015), and LinkNet models in this project. Based on the models installing software like Python programming language (NumPy, Matplotlib, Seaborn, and Scikit) and frameworks (TensorFlow or PyTorch). After having all the resources, our next step is to create models and train the models using training satellite images.

2.2.5 Evaluation

After models are trained using the train data in the Modeling phase, We need to test and validate the model using test data. This phase also contains a set up of evaluation metrics (F1 score, IOU, Roc curve, and so on) model performance comparison, analyze error for improvements, and feedback and update the models.

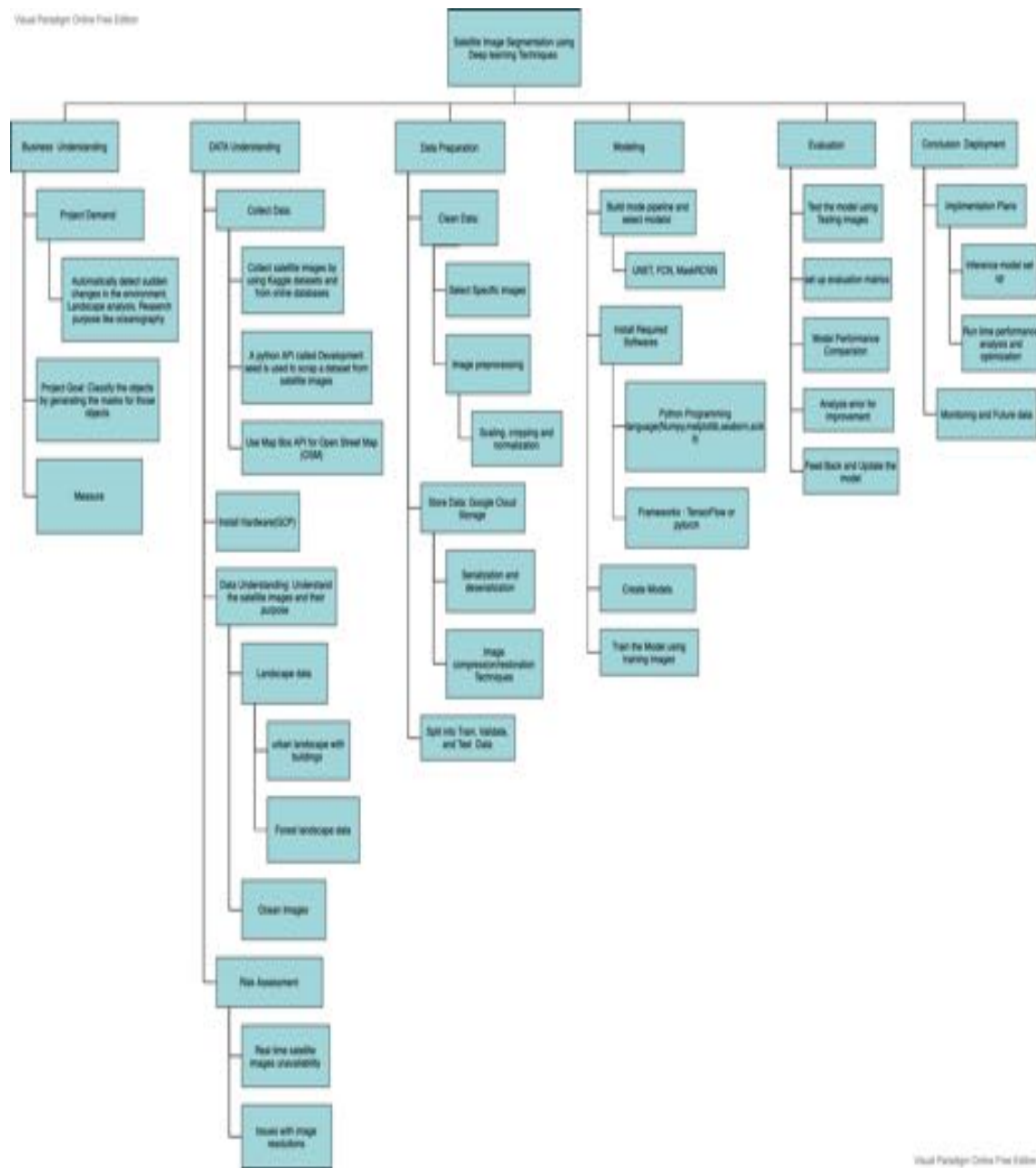
2.2.6 Deployment

After creating models we need to present them in a way that the customer can use them, that is where Deployment comes into the picture. Implementation plans contain inference model setup and runtime performance analysis and optimization. This phase also includes documenting all milestones in the form of reports and seeing if any future work can be done on the proposed system.

2.3 Project Organization Plan

Organizing and planning the project using a Work Breakdown Structure (WBS). As shown in Figure 4, Work Breakdown Structure represents the hierarchical representation of the project phases, deliverables, and work packages. A good Work Breakdown Structure should be well defined, easily estimated, manageable, measurable, and flexible (“What is Work Breakdown Structure?, n.d.).

As shown in Figure 4, This project breakdown is based on the CRISP-DM model. The project is divided into six main tasks or milestones, which are Business Understanding and requirements, Data Understanding, Data Preparation, Modeling, Evaluation, and Deliverable. All these six phases are again subdivided into sub tasks or components which help us to maintain and achieve project goals easily.

Figure 4*Work Breakdown Structure of the satellite image segmentation project*

Note. Satellite image segmentation project work breakdown structure with milestones, deliverables, components, and work packages. It also shows the dependencies between each milestone.

The Business Understanding task is divided into several sub tasks like Project Demand, Project goal, and Measurement tasks. The Project Demand task captures critical applications that can be built from this project and their demand in the real world scenario. Some of the applications relevant to this project are automatically detecting sudden changes in the environment, landscape analysis, and research purposes like oceanography. The Project Goal task determines the objective of the project based on the demands in the real world scenario that is to classify the objects by generating the masks for those objects by using Deep Learning techniques. The Measurement task tracks the project progress by keeping a tab on all the tasks that are in-progress or completed and provides a quantitative assessment of the project status.

The Data Understanding task is an important segment of the project which deals with datasets which in turn are very critical for deciding the ML model and project execution. It includes tasks like Collect Data, Install Hardware, Data understanding, and Risk assessment. The Collect Data subtask deals with collecting or generating satellite images from various sources like Kaggle, and online Databases. This task also includes artificially generating simulated data by using Mapbox API for OpenStreetMap (OSM). The Install Hardware task is about setting up the hardware for the project, this includes setting up the computer instances and the storage in the cloud services. After setting up the storage services the dataset collected could be stored for further execution. Once the data is collected and stored in the storage it is now ready for the next task called Data Understanding. In this task, the collected datasets are categorized based on landscape recovery like urban, forest and ocean images. Also it is necessary to understand the resolutions, image types, and other image attributes of the dataset. This information is required in creating the ML model and defining the appropriate loss functions for the model. This information also places an important role in defining the optimization strategies used for

performing weight updates in the model. The Risk Assessment task deals with identifying the risks associated with the collection of the dataset and their future availability. Some of the high risk aspects are the real time availability and issues with image resolutions.

The Data Preparation task deals with Data Cleaning, Storing the prepared data, and Splitting the dataset into train, test, and validate data. The Data Cleaning subtask involves removal of duplicate images, performance sanity check to check the validity of the images and discarding the invalid. It also involves preprocessing the images by performing scaling, cropping, and normalization. This step is crucial for model performance and accuracy and if performed properly will reduce the training time significantly. The Stored Data task is about efficient storage and retrieval of the dataset. This involves processes like serialization and deserialization that are important parts of the model input pipeline. The efficiency of the storage and retrieval could be further enhanced by using state of the art image compression and restoration techniques. The final task is about splitting the dataset into training, testing, and validation subsets. Depending upon the dataset, it could be either random splitting or involve some kind of fursit based on the geographical area to increase the generalization capacity of the model.

The Modeling task deals with the definition of the applicable ML models for the project. The current project used three different models like UNET, FCN, and LinkNet. Some of the subtasks involve installing all the software required to set up the environment and install the frameworks like Tensor flow and Pytorch including the supporting tools in python. Once the models are created and software is set up the next task is about running the model along with the input pipeline to perform training.

Once the model is trained, The Evaluation task runs the validation dataset on the trained model to obtain the evaluation performance. There are several evaluation metrics used in image segmentation like input and label image, pixel comparison and MeanIoU (mean intersection over union). The performance of the trained model is evaluated using these metrics by running the model on validation and the image subsets. If the model is lower than the defined threshold, the model is again trained considering this as the feedback by slightly updating the model hyper parameters. This loop of model performance feedback and model update process is continued till the model attains sufficient performance.

The Conclusion and Deployment task includes Implementation plans to integrate the model with the application of interest. This also involves the setting up the inference model from the training. The inference model is executed in the user data where its runtime performance is continuously monitored.

2.4 Project Resource Requirements and Plan

Resource management plays a huge role in project management. Hansen (2018september) said in his blog, if we know what we need to complete the project then we can plan for those resources in an efficient way. Advantages of proper resource management are it avoids unforeseen hiccups, prevents burnouts, builds transparency, and measures efficiency (Hansen, 2018september).

2.4.1 Hardware requirements

For this project, we use UNET and other FCN models to train for image segmentation. In general, these models are very compute-intensive and can not be run only on CPU servers. We require hybrid computational resources that involve GPUs and state-of-the-art Accelerators. For Accelerators Google Cloud compute engine provides the option of using either Graphics

Processing Unit (GPU) or Tensorflow Processing Unit (TPU) clusters. For GPUs, we use NVIDIA® Tesla® A100: 40 GB of HBM2e, 6,912 CUDA cores, 432 Tensor Cores, PCIe 4.0. For TPU we use TPU v3 16nm clock speed 940 MHz 32 GB HBM. The CPU core consists of Processors, 2 AMD EPYC™ 7003 series CPU, AMD EPYC 7313: 16 cores, 32 threads, 3.0 GHz base, 3.70 GHz boost, 128 MB cache.

2.4.2 Software requirements

This project uses Google Cloud GPU instance which is added to a virtual machine (VM), Python 3.9.5, Python ML libraries like Numpy (version 1.19.2) to perform mathematical operations on arrays, Pandas (version 1.3.3) for data science/ data analysis tasks which is build on the top of Numpy package, Matplotlib (version 3.4.3) for data visualization, Seaborn (version 0.11.0) for statistical graphics, Scikit-learn (version 0.20) is a robust library for machine learning which provides tools for statistical modeling, Tensor flow(version 2.6.0) is an open-source artificial intelligence/deep learning library used to create neural networks, Keras (version 2.4.0) for developing and evaluating deep learning models, Scikit-image or Skimage (version 0.18.0) for image preprocessing, Nvidia GPU driver and TPU driver CUDA, ODV4 to enable GPU and TPU programming

2.4.3 Tools and License

Tableau desktop for data visualization. QGIS (version: 3.10) for data pre-processing, Label Image, for data annotation, Voila (Convert jupyter notebook to a dashboard), Geemap (interactive mapping for Google earth engine). Need License for Anaconda navigator for jupyter notebook, or Google Colab, Google earth engine (GEE), Permission from U.S. Geological Survey (USGS), and Sentinel Hub.

As shown in Table 1, we are using resources like Google cloud computing which include Google cloud GPU instance (GPU: NVIDIA® Tesla® A100: 40 GB of HBM2e, 6,912 CUDA cores, 432 Tensor Cores, PCIe 4.0.) and Google cloud TPU (TPU: TPU v3 16nm clock speed 940 MHz 32 GB HBM.) instance instead of CPUs for computing the models because the models used in this project are very compute intensive. These GPU and TPU instances are available for free. For Google cloud storage resources, we are choosing a storage capacity of 10 GB per month to store all image datasets and software models. As per the Google cloud cost calculator, the current plan used in this project for computer instances and storage services costs around two dollars per month, and this subscription is continued over a period of six months which is assumed to be the project duration. Another free tier resource Heroku is used in this project for deploying Google cloud models, we can deploy web apps on Google cloud storage by using Heroku.

Table 1*Project Resources, Specification, Cost, and Justification*

S.no	Resources	Specifications	Costs	Justification
1	Google cloud computing resources	Google cloud GPU instance, Google cloud TPU instance	Free tier	Models are very compute intensive, so we are using either GPU or TPU clusters instead of CPUs. GPU: NVIDIA® Tesla® A100: 40 GB of HBM2e, 6,912 CUDA cores, 432 Tensor Cores, PCIe 4.0. TPU: TPU v3 16nm clock speed 940 MHz 32 GB HBM.
2	Google cloud storage resources	10 GB per month to store all images and software models.	Two dollars per month over a period of six months (\$12)	Including the software model and the datasets as per the Google cloud cost calculator, It costs around two dollars per month, and this subscription is continued over a period of six months which is assumed to be the project duration.
3	Deploying a google cloud models	Heroku	Free tier	We can deploy web apps by using Heroku on google cloud storage.

Note. Table which describes the resources used in the project with their specifications, costs and justification to use them.

2.5 Project Schedule

Scheduling project contains, list of activities, deliverables, milestones, and dependencies of the tasks on one another of the projects. This leads to assigning the tasks to teammates, track those tasks, and achieve project deliverables timely.

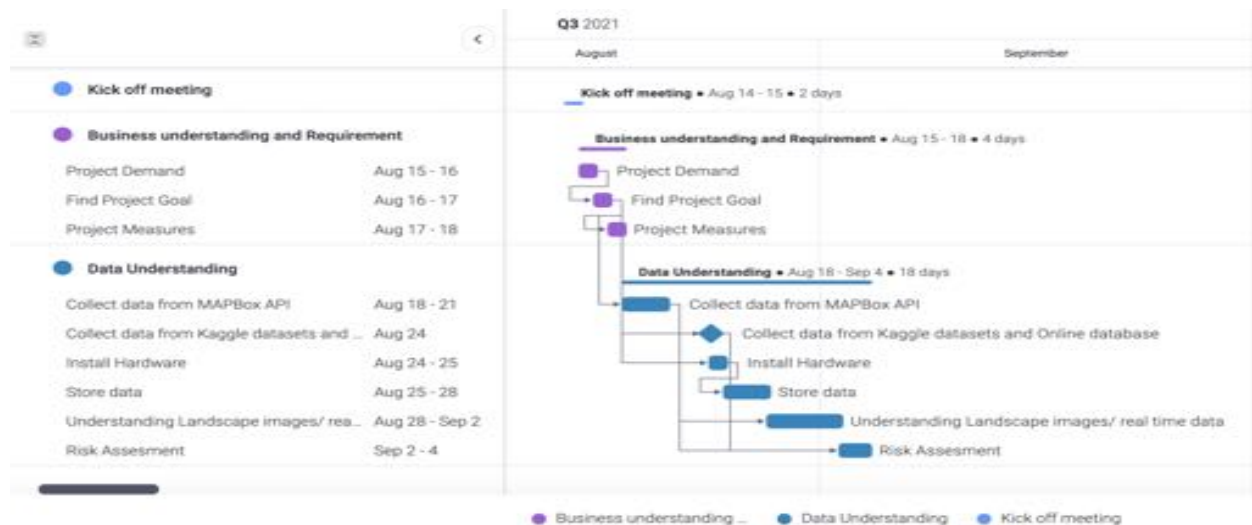
2.5.1 Gantt chart

A Gantt chart is used for planning and managing the project schedule. It is a horizontal bar representation of all the tasks in each phase with a project plan over a period of time. As shown in Figure 5, the Gantt chart represents all the tasks, timelines, and who is responsible for that task. It is a hierarchical approach of a task breakdown with information like the beginning and end of the task, each task duration, assignee of each task, task dependencies over one another, what are the important tasks and deadlines, and finally the total project duration (“Gantt Charts: Definitions, Features, & Uses”, n.d.).

In this project, we are developing a Gantt chart by considering the CRISP-DM model six phases Business understanding, Data understanding, Data preparation, Modeling, Evaluation, and Deployment as the main tasks and subdividing those tasks into subtasks with a timeline, assignee, and the dependency of that task on one another. As shown in Figure 5, The main tasks like the Business understanding phase is taking four days to complete, Data understanding is taking 18 days to complete. As shown in Figure 6, Data preparation is taking 10 days to complete, as shown in Figure 6, Create model is taking 16 days to complete. As shown in Figure 7, Evaluation is taking 16 days, and as shown in Figure 6 and Figure 9, Deployment is taking 13 days to complete. The main task duration is calculated by grouping up the duration of the subtasks under it. From the main tasks duration we can calculate the total period of project completion time which is 77 days. All the subtasks under these main tasks depend on one another.

Figure 5

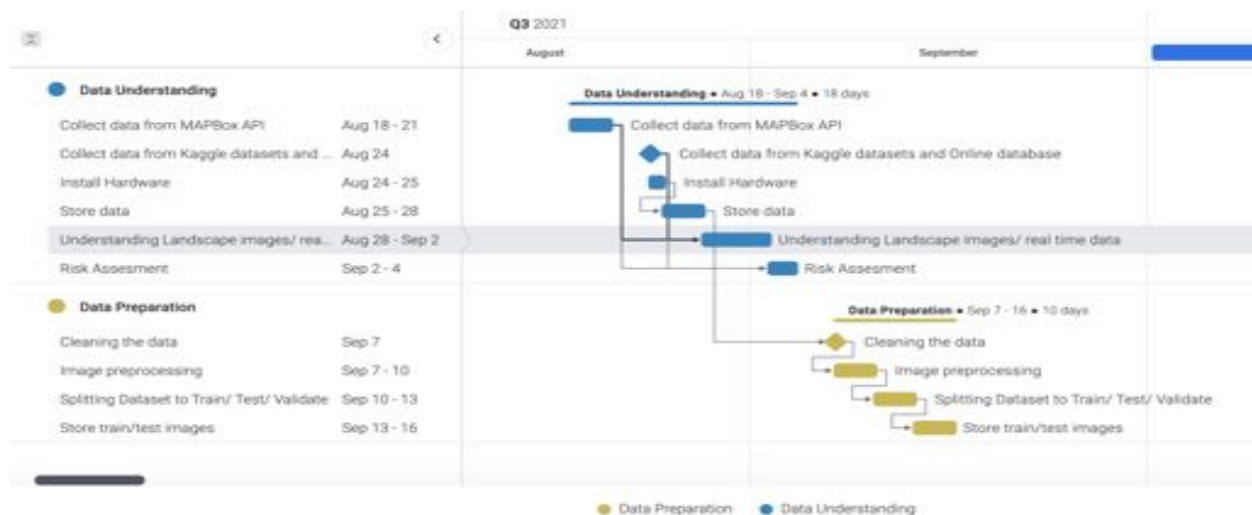
Gantt chart of Tasks Business understanding and requirement and Data Understanding



Note. Gantt chart showing dependencies and time duration of the phases Business understanding and Data Understanding.

Figure 6

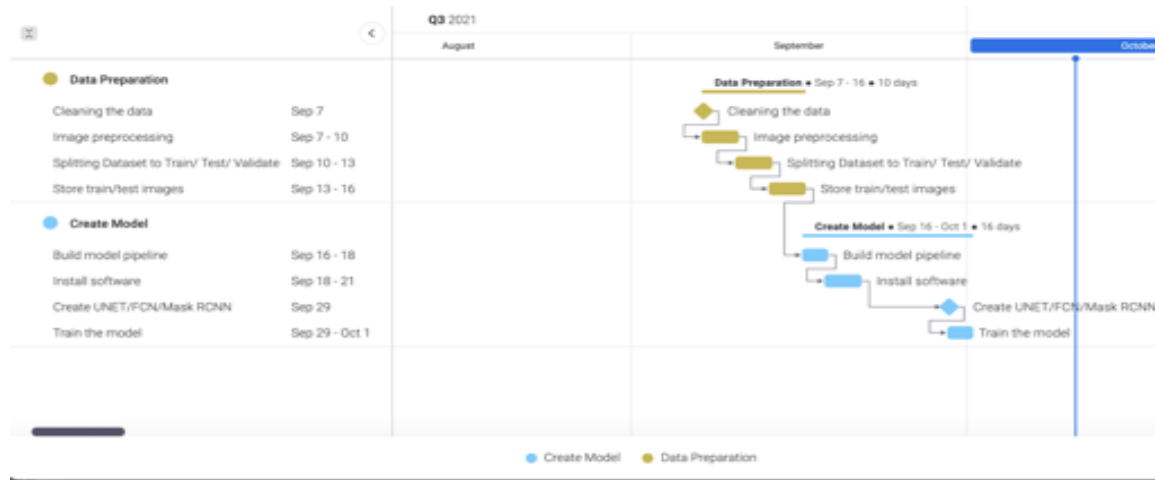
Gantt chart of the tasks Data understanding and Data preparation



Note. Gantt chart showing dependencies and time duration of the phases Data understanding and Data Preparation.

Figure 7

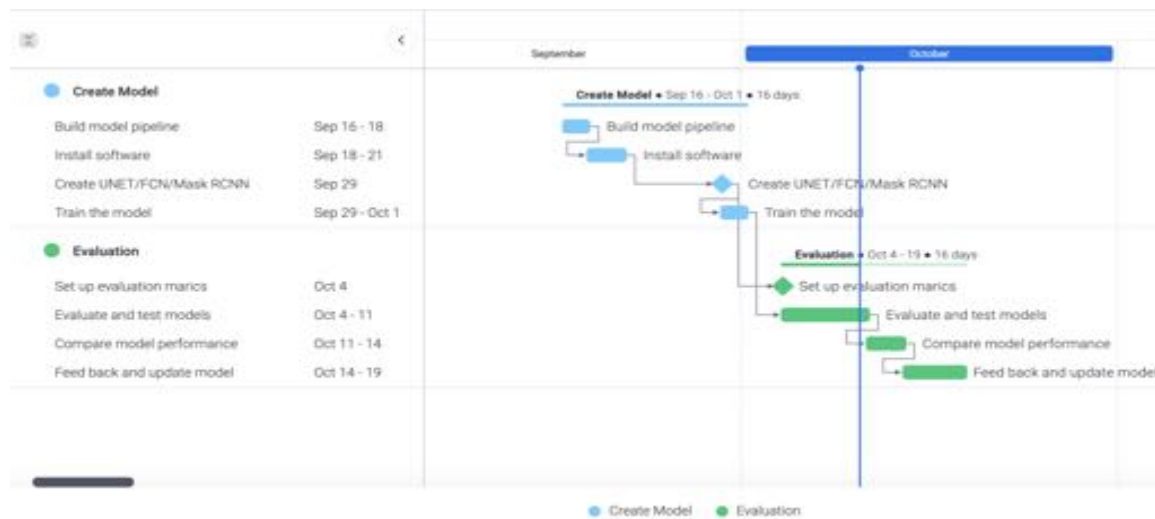
Gantt chart of the tasks Data preparation and Models



Note. Gantt chart showing dependencies and time duration of the phases Data Preparation and Create Model.

Figure 8

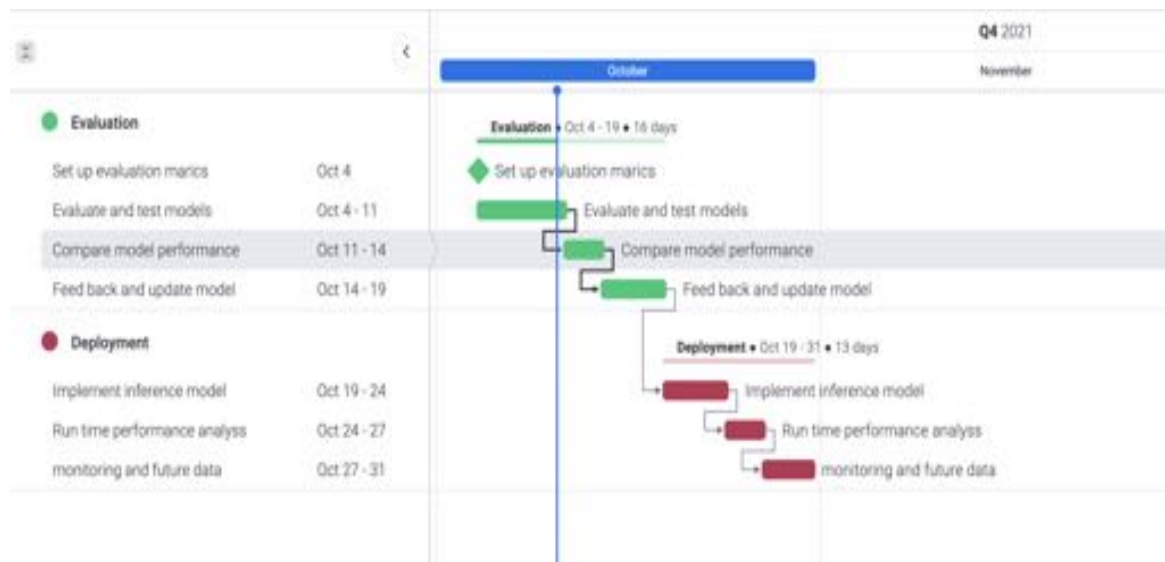
Gantt chart of the tasks Create Models and Evaluation



Note. Gantt chart showing dependencies and time duration of the phases Create Model and Evaluation.

Figure 9

Gantt chart of the tasks Evaluation and Deployment



Note. Gantt chart showing dependencies and time duration of the phases Evaluation and Deployment.

2.5.2 Pert chart

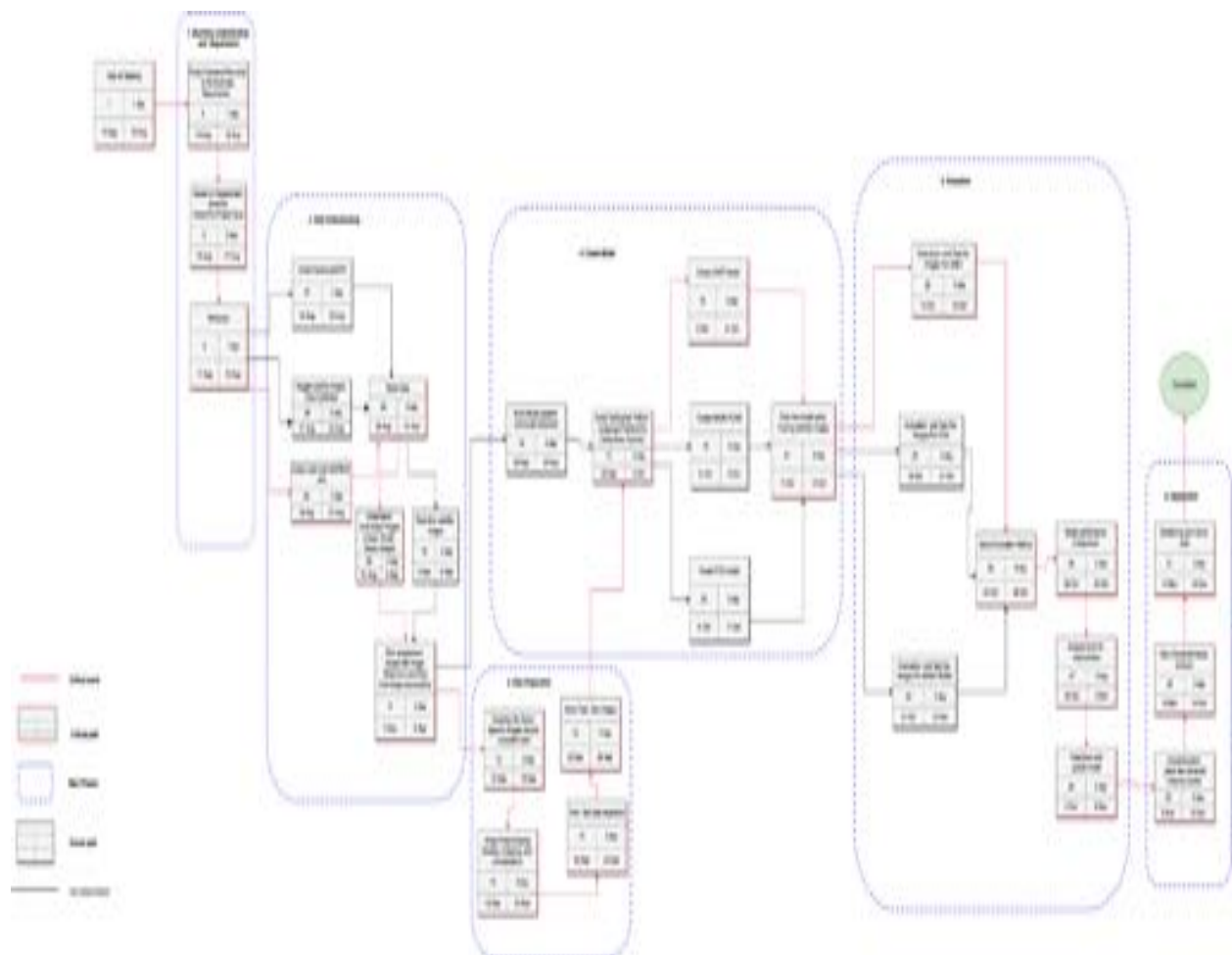
Program Evaluation and Review Technique (PERT) is a type of chart we can use for project scheduling. Critical Path Method variation (CPM) is a method that we can include in the pert chart for efficient project management.

According to the author Boushaala (2014), the PERT chart with CPM is a major tool for project management. The critical path is the network, which shows the longest path for the relations of activities towards the final activity with time. Basically Critical paths are the ones which are used to find the completion time of the project. We can also find the earliest start time and end time of the project by using CPM/PERT. We can analyze the timing of every activity based on the infinite resources it is taking to complete by using a critical path.

As shown in Figure 10, we are using the PERT/CPM chart to schedule the satellite image segmentation project. The schedule of this project starts with the kick-off meeting and we are scheduling all the subtasks in the work breakdown structure with the number of days to complete those tasks (estimated time duration). All the tasks are dependent on one another. For instance, As shown in Figure 5, creating ML models is dependent on the setup software environment. We are also defining the critical path for this project which is in the red arrow, and the critical red-bordered rectangular box represents the critical task. The black arrow and black rectangular box represent the normal events path and normal tasks which are not taking too much time to complete. The path which is shown in the red line is taking a minimum of 59 days to complete and the path in the black line is taking 23 days to complete. We are considering a path with 59 days as a critical path, we can consider that the tasks in that path are very crucial tasks to complete this project since they are taking a long time to complete. Basically this critical path calculation determines the project completion time.

Figure 10

PERT chart performing project analysis with individual tasks and dependencies.



Note. Pert chart with all milestones and the dependencies between them. This Pert chart is based on the Critical path analysis, which is the longest time taking path. Red arrows and red bordered blocks represent the critical events of this project.

3. Data Engineering

3.1 Data Process

Data process is a systematic step to gather the qualitative and quantitative data from different sources and prepare that data for training, validation and test datasets to achieve the project objective. Collecting the right data for Machine Learning models is a difficult task in a project.

3.1.1 Data Collection process and steps

Before deciding the final datasets for our project we are following some methods to collect qualitative data for training the Machine Learning models.

3.1.1.1 Background Study. In this step, we have done some research on the data. We identified what kind of data is required for the project to reach objective. We need various kinds of satellite images like urban, rural, roads, buildings, ocean, and vegetation satellite images. In this we described the top level aspects of the project like whether they are going for supervised and unsupervised training. We chose supervised training that means the dataset that has the images and labels.

3.1.1.2 Literature Survey. After doing research on background study, we wanted to study some existing research on the satellite image segmentation to understand what type of data they have used to build a model. The author Ivanovsky et al. (2019), has used Inria and SpaceNet databases to collect and train the Machine Learning model. SpaceNet train and test data is categorized based on the geographical location and is collected by using Worldview-2 and Worldview-3 satellites. He is considering only satellite images of buildings to detect or segment the buildings in the image using UNET architecture. Our project is not only based on detecting buildings but also includes all kinds of satellite images. Some other papers were using Kaggle

datasets to train the model. Those images are collected by sentinel-2 satellite. From these existing researches we got an idea to collect datasets for our project.

3.1.1.3 Explore Additional Resources. We got some ideas to collect satellite images of buildings. Exploring additional resources to collect a good quantity of data with all types of satellite images. After doing a research on collecting the data, we found some existing kaggle datasets which have generalized satellite images but the quality of that data is very less we wanted to train our model on a good amount of data so keeping the Kaggle datasets and some data from database which we have studied from the literature survey, we explored the NASA earth data. NASA provides satellite images in the form of Global Imagery Browse Services (GIBS). These satellite images are freely available for the public. There are 900 images in NASA earth data and we can also access historical and real time images. While exploring some other resources we found recent research on the Mapbox framework. We have decided to make use of this framework to create artificial satellite images by using Mapbox API.

3.1.1.4 Analysis and Dataset Pruning. Now we have a good amount of data to train the model. Before moving to modeling we need to analyze or explore the data. Data exploration leads to prune the unwanted data and keep the satellite images that are required for this project. Our data sources have all kinds of satellite images, we are grouping the images based on their classes so that it will be easy for us to track the images in the future.

After studying all the methods to collect datasets, we have decided to use two types of satellite image datasets to achieve satellite image segmentation for our project. We shortlisted the following type of satellite images:

1. Land cover datasets, which cover urban and rural areas' satellite images such as buildings, roads, and trees.

2. Ocean datasets, which cover ocean and vegetation area satellite images and help in research areas like oceanography.

Based on the above types of datasets, we are segregating all these types of satellite images into different folders and restoring them. It may help us prepare training, test and validation datasets. We are splitting training, test, and validation sets based on three techniques. The first one is Random Splitting, where we randomly choose images to form training, testing and validation subsets based on a fixed ratio. The second one is called K-Fold Cross Validation technique, and the third one is called Region Based Splitting. The first technique is straightforward, the other two techniques are explained in more detail in the data preparation stage.

3.2 Data Collection

We are collecting the relevant data from different sources to achieve the project goal. Data collection is an important step in Machine Learning. Our main agenda of this project is to segment all kinds of satellite images like ocean satellite images, landscape satellite images which include urban and rural. So based on our goal, we have decided to collect satellite images. After conducting research on collecting satellite images, we have decided to collect data from various open sources like Kaggle and available satellite Databases (Inria and SpaceNet). Apart from these sources we are also collecting our own data.

3.2.1 Existing Data Sources

Collected Kaggle datasets contain Semantic Segmentation Of Aerial Images, Satellite Images Of Water Bodies, and Satellite Building Semantic Segmentation images. Semantic Segmentation Of Aerial Image dataset is an open source satellite image dataset. Which contains Road, Vegetation, Water, Building, Land, and unlabeled classes of satellite images. The size of

this dataset is 72 images grouped into six larger tiles. There are 2841 source images and the same number of label images in the dataset. The images are colored and the labels are black and white. Each image is of resolution $565 * 568 * 3$ pixels where each pixel holds a uint8 value. The average image size is about 868.56 KB and the total dataset is about 287 MB (“Semantic segmentation of aerial imagery, 2020).

Satellite Images Of Water Bodies, this dataset is collected by using Sentinel-2 satellite. As shown in Figure 12, the dataset contains satellite images of water bodies created for segmenting water bodies and vegetation. The images are colored and the labels are black and white. Each image is of resolution $1479 * 2149$ pixels where each pixel holds a uint8 value. The average image size is about 194.63 KB and the total dataset is about 34 MB. Black and white masks are generated by calculating NWDI (Normalized Water Difference Index) (Escobar, 2020).

Land Cover Classification dataset from deep globe challenge, the author Ashwath (2020), has collected this data and stored it in kaggle. As shown in Figure 14, the dataset contains satellite images of water bodies, forest, roads, water, and agricultural land. The images are colored and the labels are masked with respect to their classes. Each image is of resolution $2448 * 2448$ pixels of 803 images. The total dataset is about 3GB. This dataset contains satellite images from Las Vegas, Paris, Shanghai, and Khartoum cities (Demir et al., 2018).

The final Kaggle dataset used in this project is Satellite Building Semantic Segmentation Data. As shown in Figure 12, this dataset contains satellite images of buildings. This dataset contains image and corresponding segmentation labels for the images. There are 6030 source images and the same number of label images in the dataset. The images are colored and the labels are black and white. Each image is of resolution $256 * 256 * 3$ pixels where each pixel

holds a float value. The average image size is about 128 KB and the total dataset is about 921 MB (Wang, 2020).

Other than Kaggle datasets, we are collecting data from available databases such as Inria and SpaceNet. As shown in Figure 14, the Inria database contains satellite images of buildings from various cities. Images of this dataset have a resolution of 1000 * 1000 pixels where each pixel covers an area of 0.3m. The author of this database Maggiori et al. (2017), splits the dataset based on geographical location. For instance, the model is trained on the images of San Francisco city but it is evaluated and tested on other cities. The Author Maggiori et al. (2017) states this kind of splitting technique will allow for more generalization for the model.

The SpaceNet database contains images of buildings and roads networks collected by using Worldview-2 and Worldview-3 satellites. These satellites collected Eight-channel photos with different spatial resolutions. This dataset contains several subsets which are subdivided based on tagged objects. Dataset contains two subsets, one set is of 3011 km² and the second one is 5555 km² for building detection (Ivanovsky et al., 2019). As shown in Figure 15, the sample satellite building image looks in the SpaceNet database. We can download data from this dataset by using the Amazon Web Services (AWS) portal.

3.2.2 Raw Data Collection

To make the model more generic, we would like to train it on all kinds of satellite images. As a result we collect and make our own dataset. For this we use the NASA earth data. NASA provides satellite images in the form of Global Imagery Browse Services (GIBS). These satellite images are freely available for the public. There are about 900 images that are available in full resolution and are optimized for their color and aspect ratio. The other advantage is that the NASA earth data contains historical data over a period of years, which is very useful to study the

geographical changes that have occurred. These images can be accessed using the GIBS API. GIBS API can be used on the images to also generate masks which is useful to get the labels that are necessary for training the model. This is an ongoing active service from NASA which means that it is possible to get current satellite images (Bagwell, n.d.).

We are also collecting data from the satellite Landsat8. Landsat Data Continuity Mission (LDCM) was launched in 2013. The Landsat8 satellite is the one which is the recently launched Landsat satellite. In this project we are using vignette landsat dataset. The vignette landsat is of the shape $2158 * 1941$ and contains seven bands with 41,88,678 pixels. To download this dataset we are using a python inbuilt library called earth and getting the data by using a method called `get_data`. The datatype of this dataset is `int16` (“Landsat 8”, n.d.).

According to new research, the Mapbox framework consists of two classes of images and data called Mapbox Satellite and Mapbox Street. Mapbox Satellite consists of satellite images of a base map at various levels of resolutions. Mapbox Street consists of vector data of streets and interconnectivities between the locations. The combined Mapbox Satellite Street layer consists of satellite images of various locations with routes, buildings, and trees. This layer is used in many gaming infrastructure to generate artificial maps. Similar things could be done in our project to generate artificial satellite images to train the model on a specific kind of satellite images. Since these are artificial or simulated images, we can generate any amount of images using the Mapbox API at any resolution (“Mapbox Satellite, n.d.). For this project we generate around 25,000 colored images that capture urban landscape with a resolution of $256 * 256 * 3$. As shown in Table 2, we have collected satellite data from various kaggle sources with sufficient size to train our model. As shown in Table 3, we have collected satellite data from various other sources with sufficient size to train our model.

Table 2

Summary of Kaggle Data Sources and their Parameters

Name of the data	Data source	Data type	Size	Time stamp
Semantic Segmentation Of Aerial Image dataset	Kaggle	unit8	Average image size is about 868.56 KB and the total dataset is about 287 MB	2020-05-29 to 2020-05-29
Satellite Images Of Water Bodies	Kaggle collected by using Sentinel-2 satellite	unit8	The average image size is about 194.63 KB and the total dataset is about 34 MB.	2020-05-17 to 2020-05-17
Satellite Building Semantic Segmentation Data	Kaggle	float32	The average image size is about 128 KB and the total dataset is about 921 MB	2020-11-06 to 2020-11-24
Land cover dataset	Kaggle DeepGlobe	float32	2448 * 2448 pixels of 803 images. 3GB	2017-01-02 to 2018-06-28

Note. Summary of kaggle data sources and their parameters like size, data type, and time stamp.

Table 3*Summary of Data Sources and their Parameters*

Name of the data	Data source	Data type	Size	Time stamp
Satellite Images Of Buildings	Inria Database	float32	180 color image tiles of size 5000×5000	2017
Satellite Images Of Buildings and Roads Networks	SpaceNet Database collected by using Worldview-2 and Worldview-3 satellites.	float32	67,000 square km of very high-resolution imagery.	2018
Vignette Landsat dataset	This dataset is collected from Landsat 8.	int16	Contain images of size 2158 * 1941	2017-02-21 to 2021-02-21
Raw Satellite Images (all kinds)	NASA earth data (GIBS)	float32	900 images, 1 MB each	Up To Date
Artificially Generated Satellite Images	MapBox API	float32	25000 images, 120 KB each	2021-08-21 to 2021-09-02

Note. Summary of data sources and their parameters like size, data type, and time stamp.

3.2.3 Dataset Samples

We have shown some of the sample images that represent a broad category of the images in the dataset like urban landscape, water bodies, buildings for roof area analysis. As shown in Figure 11, sample satellite image contains the buildings and trees objects which is from the Kaggle dataset. As shown in Figure 12, we can see the sample image of water bodies with objects like water and vegetation which is also from the Kaggle dataset. As shown in Figure 13, we can see the sample images of Land cover satellite images which are also from the Kaggle

dataset. As shown in Figure 14, the images are from the Inria database with labelled images of them. As shown in Figure 15, the sample images are from the SpaceNet database.

Figure 11

Sample satellite image of buildings and trees



Note. Sample satellite image of buildings and trees. Adapted from *Satellite buildings semantic segmentation data*, by H. Wang, 2020 (<https://www.kaggle.com/hyyyrwang/buildings-dataset>).
CC By 4.0.

Figure 12

Sample satellite images of water bodies.

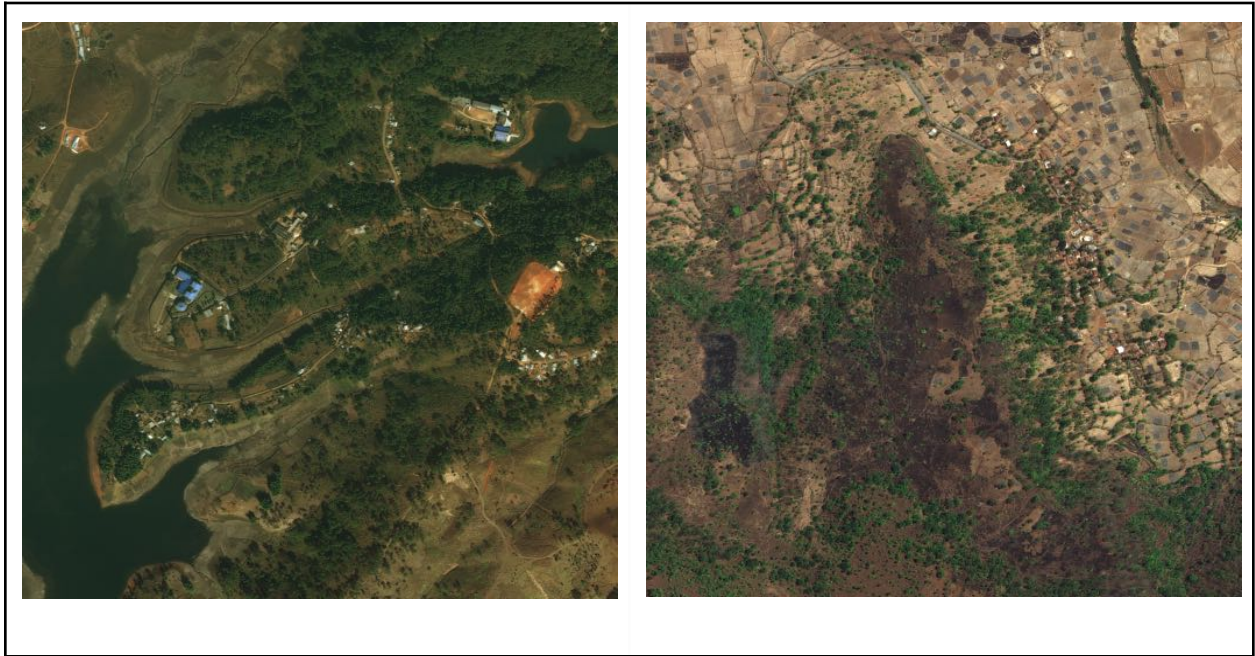


Note. Sample satellite image of water bodies and vegetation. Adapted from *Satellite images of water bodies*, by F. Escobar, 2020

(<https://www.kaggle.com/franciscoescobar/satellite-images-of-water-bodies>). CC By-NC-SA 4.0

Figure 13

Land Cover Kaggle Dataset Sample

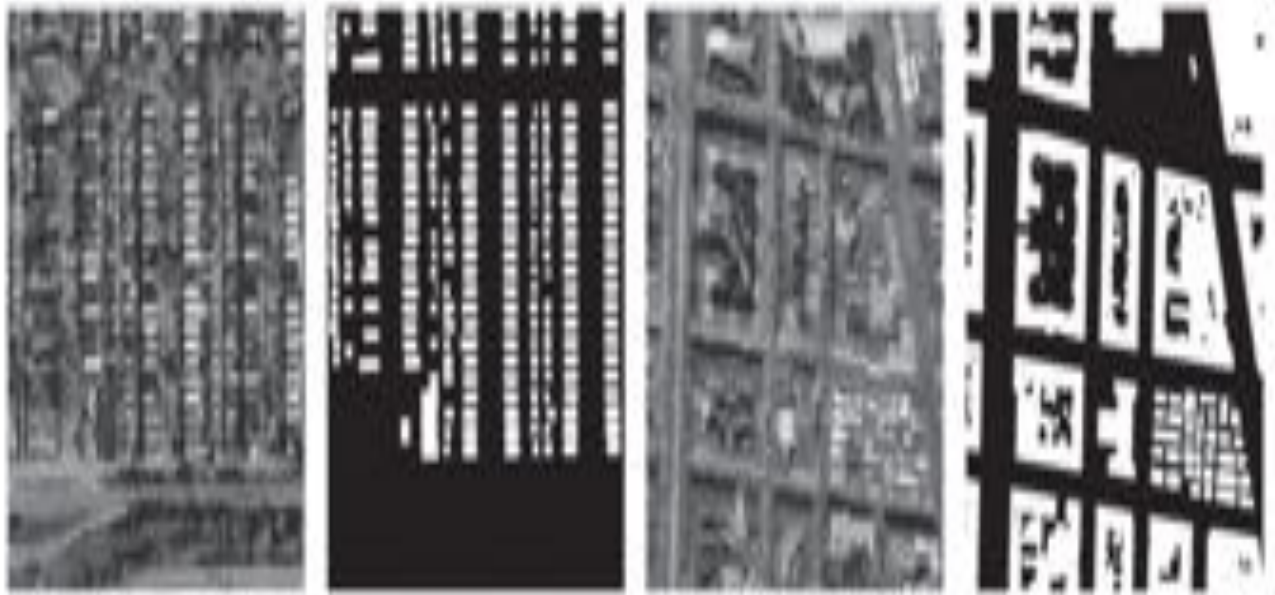


Note. Sample satellite image from land cover satellite image. Adapted from *Deep Globe Land Cover Classification Dataset*, by B. Ashwath, 2020

(<https://www.kaggle.com/balraj98/deepglobe-land-cover-classification-dataset/metadata>)

Figure 14

Sample satellite images of Inria database.



Note. Sample satellite images of Inria database. Adapted from “Building detection on aerial images using u-net neural networks”, by Ivanovsky et al., 2019, pp. 116-122 (<https://doi.org/10.23919/fruct.2019.8711930>). Copyright 2019 by 24th Conference of Open Innovations Association (FRUCT).

Figure 15

Sample satellite images of SpaceNet database.



Note. Sample satellite images of Inria database. Adapted from “Building detection on aerial images using u-net neural networks”, by Ivanovsky et al., 2019, pp. 116-122 (<https://doi.org/10.23919/fruct.2019.8711930>). Copyright 2019 by 24th Conference of Open Innovations Association (FRUCT).

3.3 Data Pre-processing

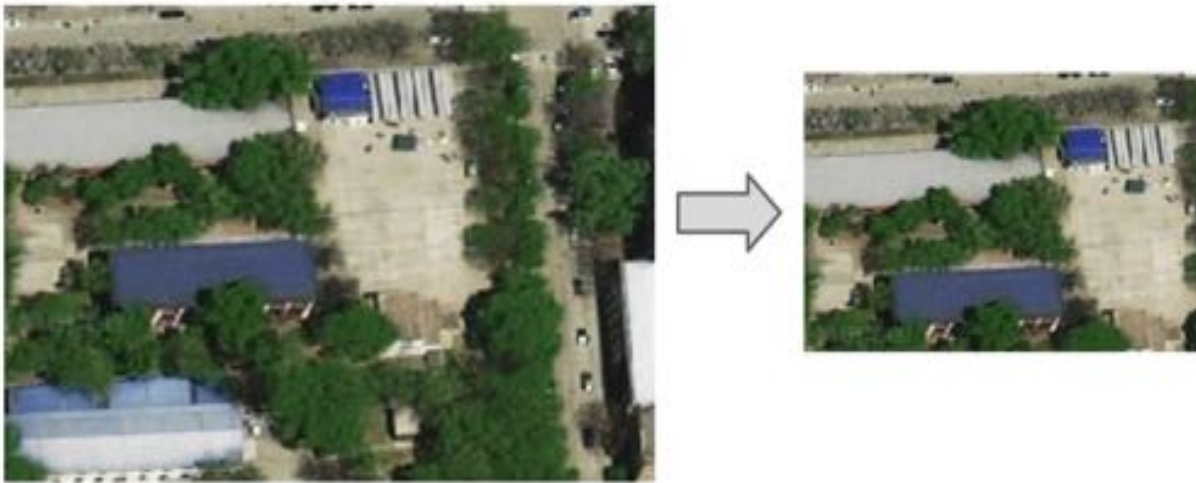
3.3.1 Data Exploration, Cleaning, and Validation

Exploratory Data Analytics (EDA) is an important first step in Data Analysis. Which helps us to identify the different patterns and outliers in the data. In this project we are exploring and preprocessing the raw satellite images which are collected from various sources by visualizing them. To do that we are installing an open source python package EarthPy to visualize the spatial data. We are also using Matplotlib to plot graphs. For exploration purposes we are taking sample data from our stored repository, and reading them in the jupyter notebook.

3.3.1.1 General EDA and Cleaning. Before we even do any kind of domain based image analysis, we want to perform basic image transforms and general statistical analysis on the dataset. In the entire dataset we have collected images from different sources and they all come in different sizes. In that we decided to choose the size of the images and labels to be $256 * 256 * 3$, this is to keep the training performance items of computation and memory. As shown in Figure 16, the image resizing process take place by converting $1000 * 1000$ image size to $256 * 256$ standard image size. Also the images from different dataset sources had different pixel data types like eight bit unsigned integer, 32 bit float, etc. Since the training involves floating point operations, we would like to convert all the image pixel data types to 32 bit float.

Figure 16

*Image resizing from $1000 * 1000$ to $256 * 256$*



Note. Figure shows image resizing technique from $1000 * 1000$ image size to $256 * 256$.

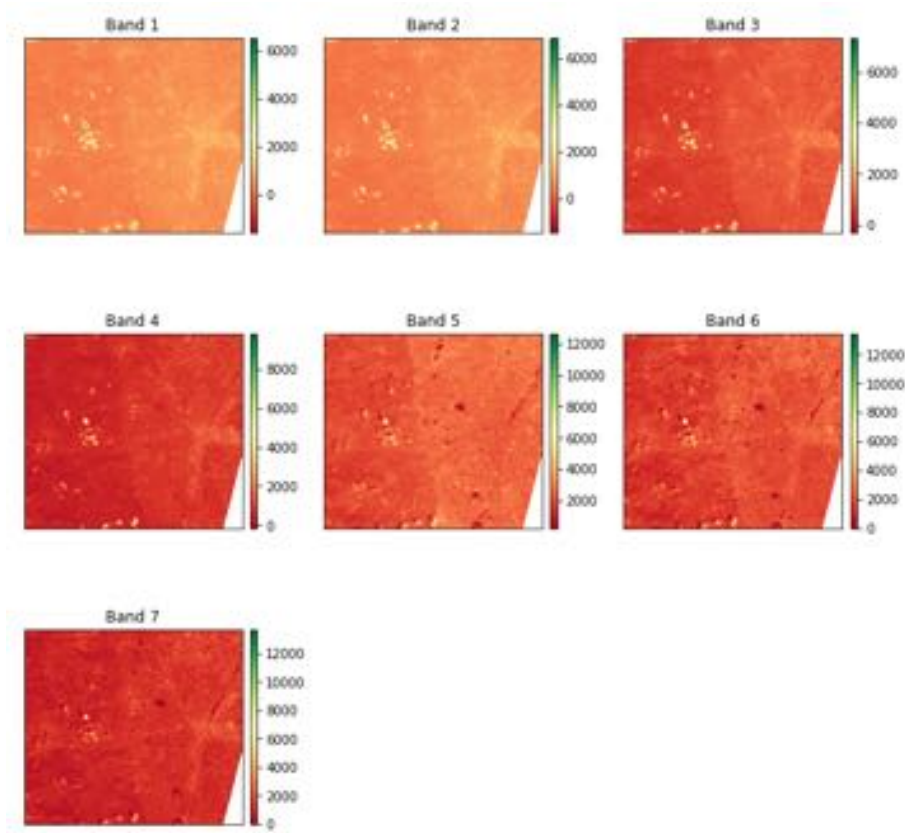
For statistical analysis we are computing the mean, median, and standard deviation of pixel values of each image in the constructed dictionary where we store the image_id and its statistics as key value pairs. We plot the histogram for each of these general statistics, and after visualizing them, we observe that the majority of the images are clustered at the center with very

few images of either side resembling a normal distribution or a bell curve. From these plots we can set lower and higher thresholds and select all the images within the threshold while discarding the others. Before discarding, we also perform a manual inspection to see if the images are really valid. This provides a first level of filtering or pruning technique to cleanup the dataset.

3.3.1.2 Domain EDA and Cleaning. Images from Mapbox and NASA earth data are raw satellite images. Which provides us a lot of opportunities for exploration. The metadata of this dataset shows information like the shape of the dataset, number of bands , and pixel information. Raw images will not have three bands so we need to explore those image bands. In research from Kakarla (2020), for instance we want to visualize a Landsat8 dataset which has seven bands. Plotting those seven bands by using plot bands. As shown in Figure 17, All seven bands are uniquely identified.

Figure 17

The plot of all seven bands in Landsat8 data set



Note. Plots of all seven bands in the dataset. Adapted from *Exploratory Data Analysis (EDA) on Satellite Imagery Using EarthPy*, by S. Kakarla, 2020

(<https://towardsdatascience.com/exploratory-data-analysis-eda-on-satellite-imagery-using-earthpy-c0e186fe4293>). Copyright 2020 by Towards Data Science.

As shown in Figure 17, the multiband images are hard to visualize and understand by humans. So we need to convert that into composite RGB three band images. To do that we need to plot images for red, green, and blue bands with index values three, two, and one. These index values are extracted by subtracting one from each index by using a zero based index system

(Kakarla, 2020). As shown in Figure 18, the images are converted from seven bands to three bands so that we can understand it better.

Figure 18

Converted RGB Composite Image



Note. RGB composite image after converting it from seven bands. Adapted from *Exploratory Data Analysis (EDA) on Satellite Imagery Using EarthPy*, by S. Kakarla, 2020

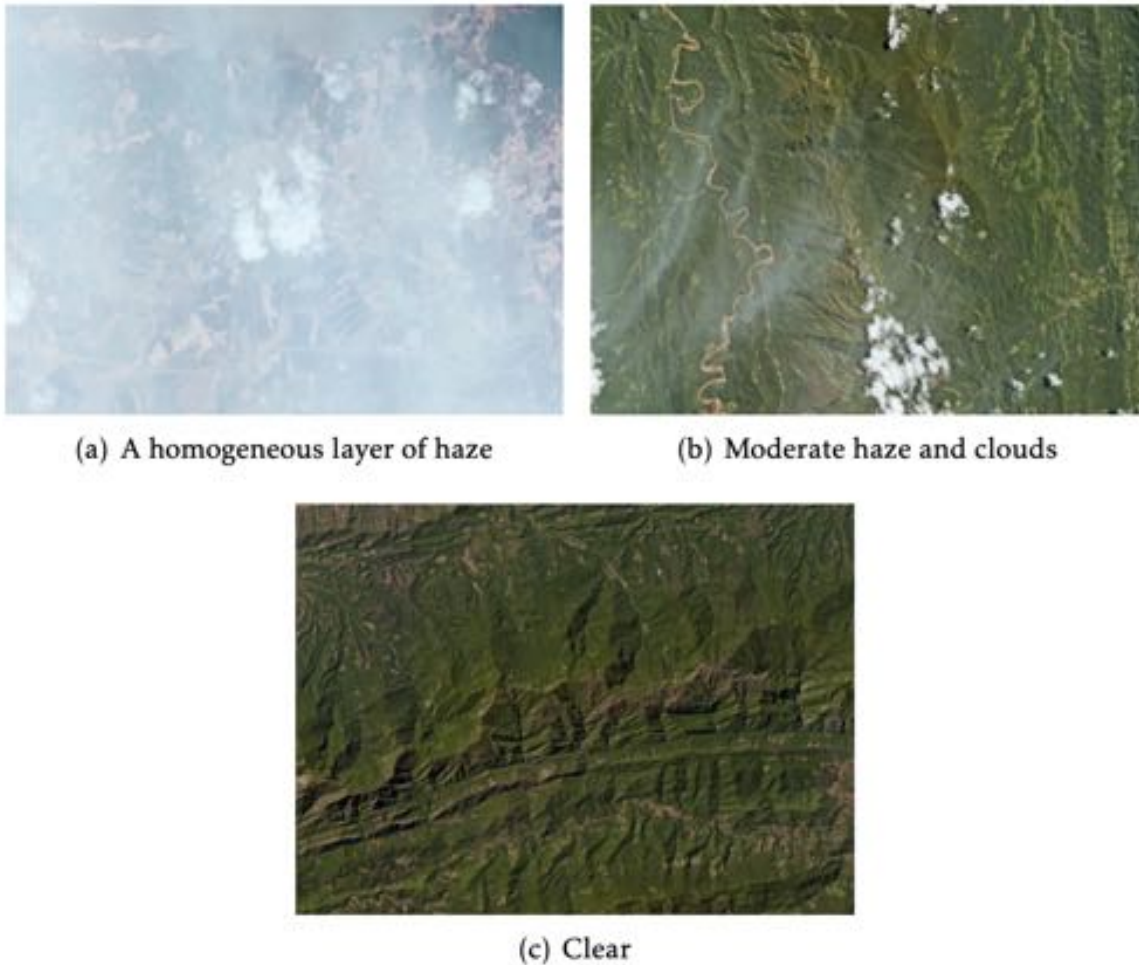
(<https://towardsdatascience.com/exploratory-data-analysis-eda-on-satellite-imagery-using-earthpy-c0e186fe4293>). Copyright 2020 by Towards Data Science.

Depending on the region some satellite images contain large amounts of haze and clouds for example satellite images of top of the hill surface in the colder regions. For removal of clouds and haze from satellite images we used two techniques, the first technique uses dark channel prior. And the second technique involves the process of haze removal using near infrared. These techniques are used in general images but also provide considerably good results on satellite images. Between the two techniques the dark channel prior provides best results by removing the most amount of cloud or haze and makes the images look very realistic. As shown in Figure 19,

the above method was able to remove clouds successfully from the satellite images to provide clear images (Hultberg, 2018).

Figure 19

Cloud Cleaning from Satellite Images



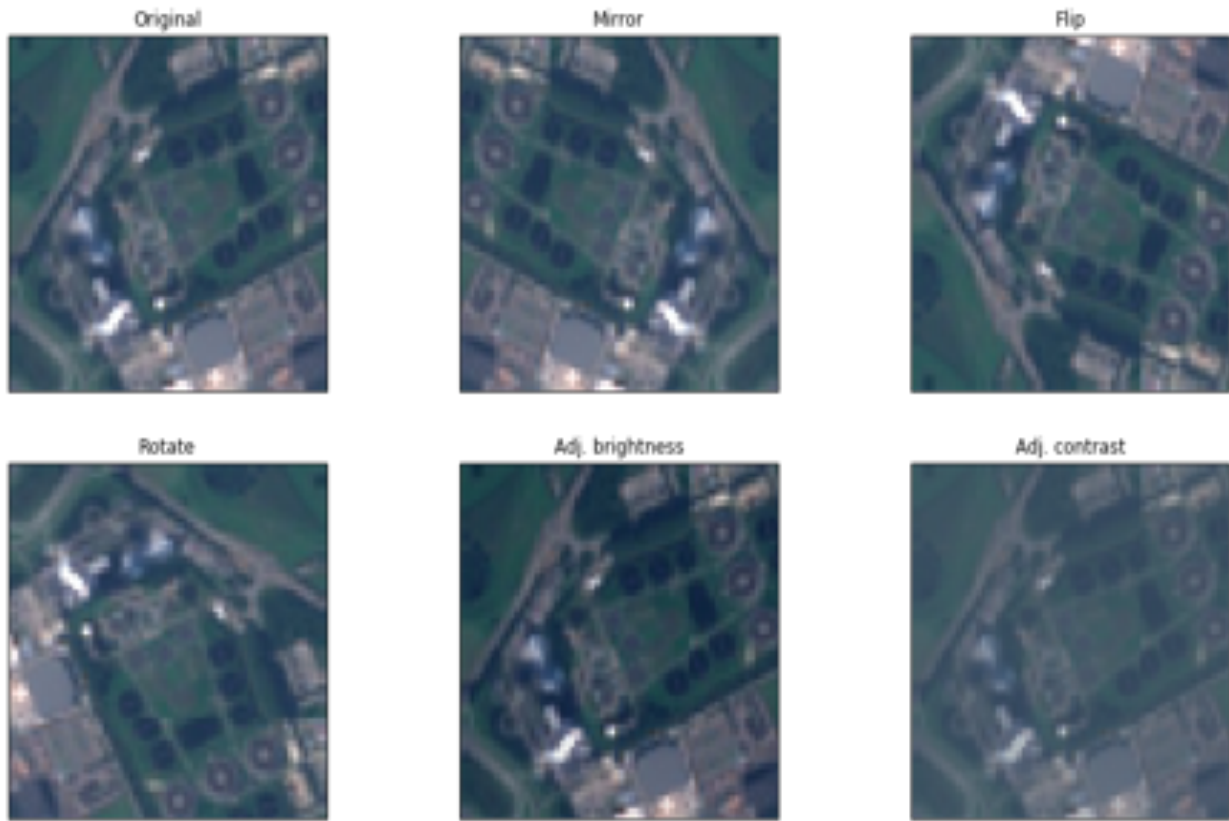
Note. Images show a homogeneous layer of haze, Moderate haze and clouds, and a clear satellite image after removing the clouds. Adapted from *Dehazing of satellite Images* by Hultberg, J. (2018) (<https://liu.diva-portal.org/smash/get/diva2:1215181/FULLTEXT01.pdf>). Copyright 2018 by Diva-Portal

3.4 Data Transformation

3.4.1 Augmentation

To train the model to handle all kinds of variances on the real world data and to better generalize the learning process we need to upsample the image dataset by using image augmentation techniques like shifting, flipping, adjust brightness, adjust contrast, rotation and transformation. The shifting is performed both vertically and horizontally where we set the vertical and horizontal offsets for the pixels keeping the images size constant. For rotation we use image processing filters that rotate the images in both clockwise and anticlockwise directions keeping the image size constant. The rotation angles are selected randomly. Flipping is similar to rotation but along the axis on the image plane whereas rotation is along the axis that is perpendicular to the image plane. As shown in Figure 20, the transformation involves filtering the images to either blur or sharpen or adjust brightness or also to zoom in or zoom out keeping the image size constant. The augmentation techniques are performed on random images based on the factor we use for upsampling (Puc, 2020).

While reading the data, we learned that, satellites images are randomly stored in our data repository, we are having satellite images of different classes like Ocean images, Landscape images (urban and rural) in one repository. To train the model on each class we need to segregate those images by creating a directory for each class based on geography. This also helps us to access images easily for future use.

Figure 20*Satellite image augmentation visualizations*

Note. Example of data augmentation. Adapted from *Semi supervised learning in satellite image classification*, by J. Puc, 2020

(<https://medium.com/sentinel-hub/semi-supervised-learning-in-satellite-image-classification-e0874a76fc61>). Copyright 2020 by Medium.

3.4.2 Data Normalization

Convolutional neural network models work well if the input image is normalized. By definition, normalization means the value of each pixel should be within the range 0.0 to 1.0. There are many techniques that are available for normalization of input images namely Instance normalization, Layer normalization, and batch normalization. In instance normalization the mean

and standard deviation of the pixel values are calculated per channel and the pixels of that channel are subtracted by the mean value and then scaled with the inverse standard deviation to perform normalization. Layer normalization is about calculating mean and standard deviation for the entire image whereas batch normalization is about calculating mean and standard deviation over a fixed batch size of images. Once the mean and standard deviation are computed the pixels are normalized in the same way as it is described in instance normalization by subtracting mean and scaling by inverse normalization.

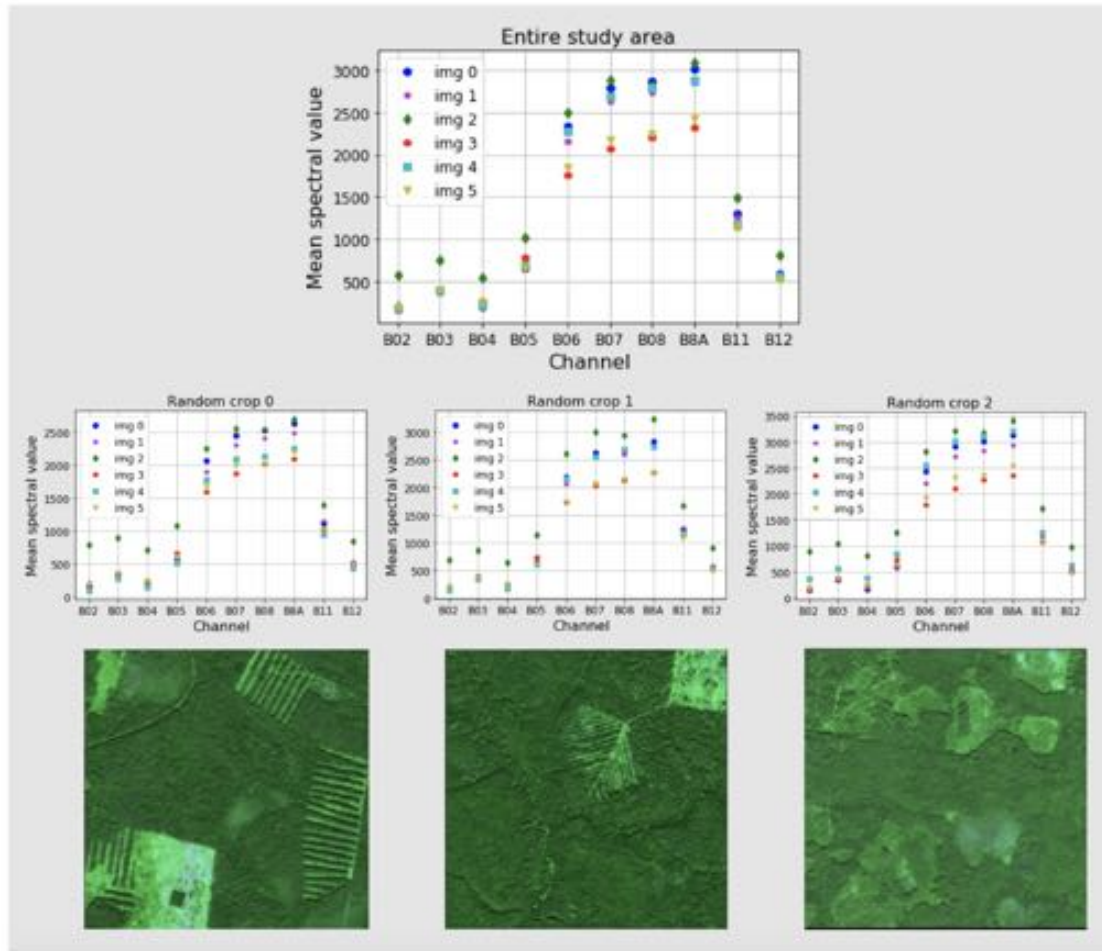
As described by the author Illarionova et al. (2021), apart from these popular normalization techniques we can also use Min Max normalization because this technique is better suited for satellite images and works best when performed per channel and also Min Max normalization technique preserves the same scale as original feature distribution. Min Max normalization is quite a known technique to normalize data. The minimum value of every feature in the data gets converted into zero and for the maximum value of that feature will be one, the other values which are in between zero and one are transformed into decimals. For example the minimum value of an array of values is 20 and the maximum value is 40. After applying Min Max normalization 30 one of the middle values of that array is transformed into 0.5. The formula for Min Max normalization with having range zero to one is $(\text{Value} - \text{Min}) / (\text{Max} - \text{Min})$ (“Normalization”, n.d.).

As shown in Figure 21, a more generalized application of Min Max normalization of images where there are eight bands in an image. The same technique can be applied for any number of bands in the images. The author Illarionova et al. (2021), explained in his paper that Landsat8 dataset's pixel values are in the range [0, 1000]. They have used B02, B03, B04, B05, B06, B07, B08, B011, B012, and B8A bands in their research. And they have resampled all the

bands to 10m. As shown in Figure 21, the average value of each image with respect to each channel (band) is presented. The average values of the entire study area distribution changes drastically for images of the same day and one year apart. If we consider the mean values of the graph which is cropped randomly as $200 * 200$ from the entire study area is also having drastic change in the distribution of band values. According to the author Illarionova et al. (2021), it is impossible to train the model with having these kinds of distribution so normalization of the data should take place to reduce the noise in the satellite images. Here we are considering the mean and standard deviation of the images having the range zero to one to calculate normalized data.

Figure 21

Mean values distribution of the images vs each bands of the entire study area and its random crops



Note. Figure shows Mean values distribution of the images vs each channel for the entire study area and random crops. Adapted from “MixChannel: Advanced Augmentation for Multispectral Satellite Images,” by Illarionova et al., 2021, *Remote Sensing*, 13(11), 2181, (<https://doi.org/10.3390/rs13112181>). Copyright 2021 by ResearchGate.

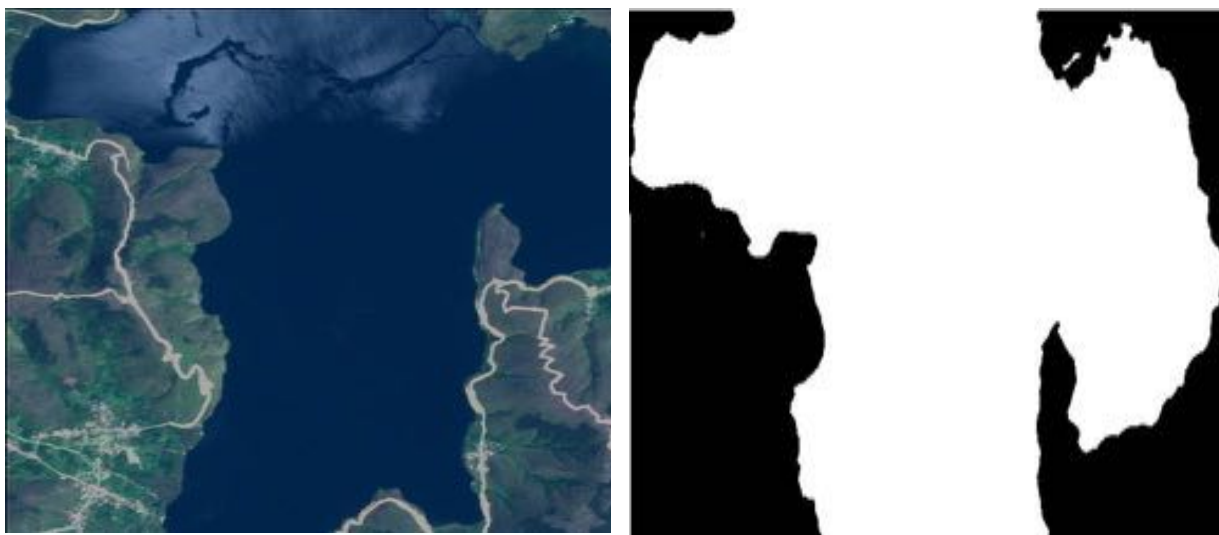
3.5 Data Preparation

The project would be designed to handle both binary classes and multiple classes classification of the image pixels in the generated masks outputs. We use binary classes for those

problems where we want to detect a specific region of interest ignoring rest of the things in the images for example building roof area detection and water body detection. As shown in Figure 22, the image shows the water body and the corresponding label shows the mask generated for that water body. All other objects other than the water body are represented by the same class.

Figure 22

Figure shows original image vs labelled binary masked image



Note. Sample satellite image of water bodies and vegetation and its labelled binary mask image. Left image shows the original one and the right one is the label image. Adapted from *Satellite images of water bodies*, by F. Escobar, 2020

(<https://www.kaggle.com/franciscoescobar/satellite-images-of-water-bodies>). CC By-NC-SA 4.0

For multiclass classification models can detect multiple objects in the image and generate masks for them which will be of different colors, which in term means the model tries to classify each pixel into one of those classes. As shown in Figure 23, The image and the corresponding masks depict the multiclass classification.

We have pregenerated images and their masks for Kaggle datasets and Inria data. For the dataset using Mapbox API we use the same API to generate the masks along with the virtually generated satellite images. Some data in our resources do not have masks for them, we are manually generating the image masks by using a tool called GroundWork. Hand labelling for thousands of images is a lot of work, instead of that we can use the GroundWork tool which makes it easy to label Geo-spatial images. The way it works is very simple. We are just dropping images into the tool and specifying the different colors for the labels like water, buildings, trees, etc. Once we draw the outlines for those segments, we can download those images for training the data (“Data labeling, n.d.). As shown in Figure 24, We used different colors for different classes, for instance blue for Arable land, green for Pastures, violet for Heterogeneous agricultural areas, brown color for Forests, and so on (Ulmas & Liiv, 2020). As shown in Figure 25, the resulting multiclass segmentation assigns different colors to each class.

Figure 23

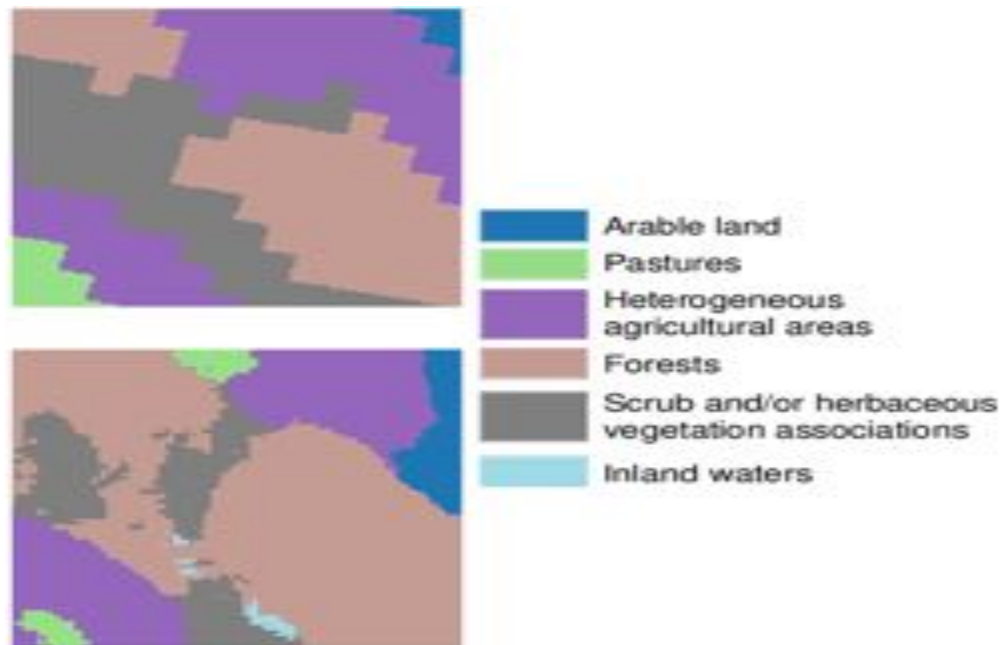
Figure shows original images vs their multiclass classification labels



Note. Original images vs their multiclass classification segmented labels. Adopted from “Segmentation of Satellite Imagery using U-Net Models for Land Cover Classification,” by Ulmas, P. and Liiv, I, 2020, *ArXiv*, *abs/2003.02899*. Copyright 2020 by ResearchGate.

Figure 24

Figure shows classification of labels with respect to colors



Note. Classification of labels with respect to the colors which we have manually assigned.

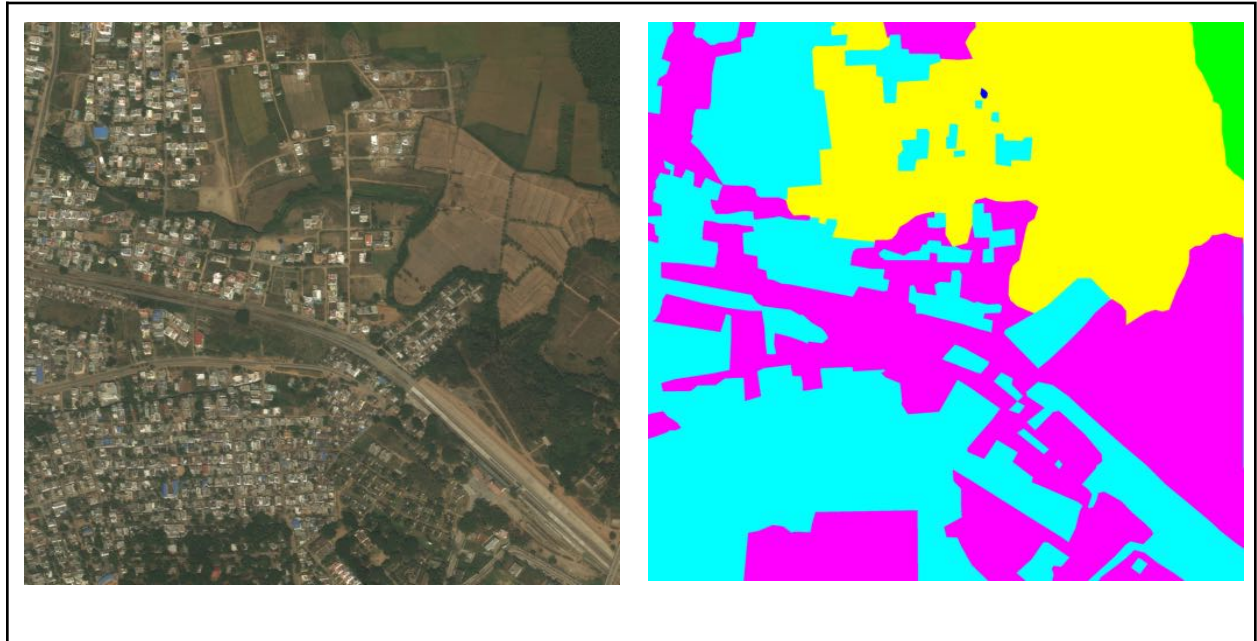
Adopted from “Segmentation of Satellite Imagery using U-Net Models for Land Cover

Classification,” by Ulmas, P. and Liiv, I., 2020, *ArXiv*, *abs/2003.02899*. Copyright 2020 by

ResearchGate.

Figure 25

Land Cover Kaggle Multiclass Segmentation samples



Note. Figure shows multiclass segmentation result. Adapted from *Deep Globe Land Cover Classification Dataset*, by B. Ashwath, 2020

(<https://www.kaggle.com/balraj98/deepglobe-land-cover-classification-dataset/metadata>)

3.5.1 Split testing, training, and validation sets

We would like to split the data into train, validation, and test sets in the ratio 75:15:10. This ratio applies to the Random Splitting and K-Fold Cross Validation Splitting but for Region Based Splitting we don't follow the fixed ratio because splitting is based mainly on regions of the satellite images.

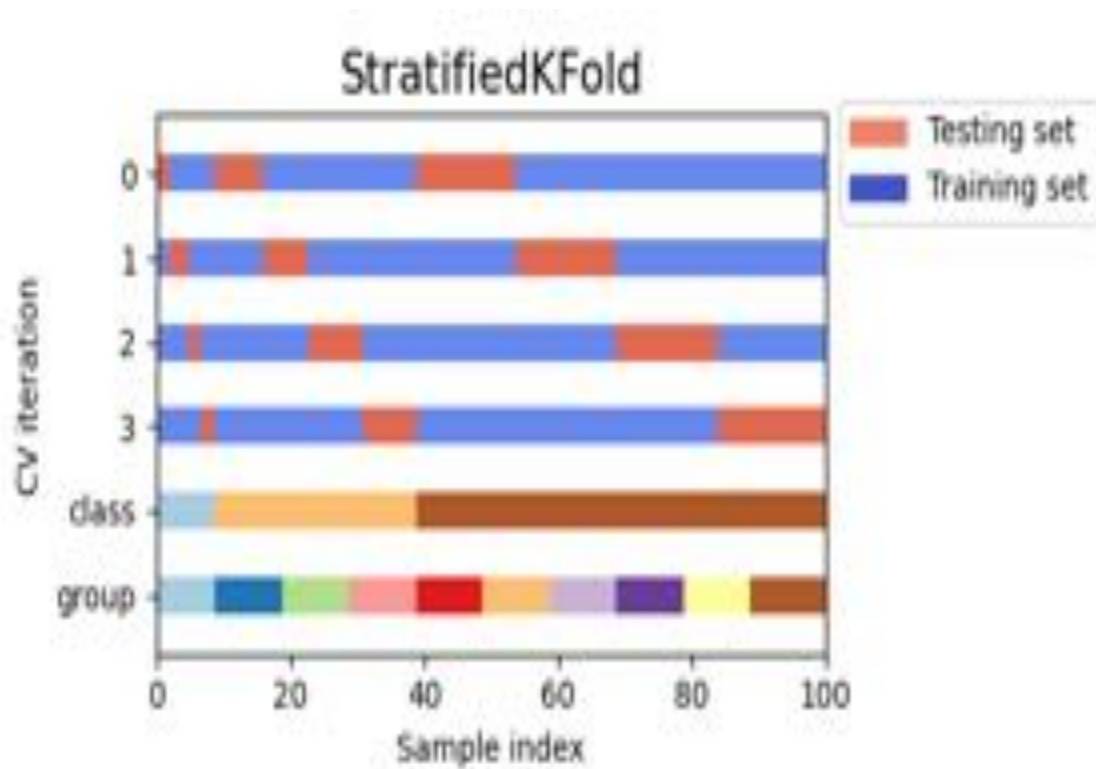
To prepare data for training, test and validation, first of all we are assigning image ID to all satellite images. Using those image IDs, we are creating a dictionary of key value pairs. For instance the dictionary will look like {key: (image ID, label)}, where key will be the randomly assigned unique number and value is the tuple of corresponding jpeg image ID and the label of

that image. Some of our datasets have labels of the image and some are not. We are manually creating those labels by masking the images and storing them. From the constructed dictionary, we are extracting keys and creating an array.

3.5.1.1 KFold Cross Validation. We would like to use the KFold Cross Validation method to split the training, test and validation datasets by using an array which we have created from a dictionary. Obviously while feeding the test data to the model, we are not giving the labels. The KFold Cross Validation method reduces the bias in the distribution of the data so that it will solve the model overfitting problem. In this project we are using Stratified KFold CV, Importing KFold, StratifiedKFold by using the ScikitLearn library. According to the author Manna (2020), stratified KFold will shuffle the data before splitting into KFold, where K is the number of splits for sets in our project. As shown in Figure 26, the behaviour of the Stratified KFold works in a way that folds are created such that the ratios of each class remain the same. Shuffling leads to maintaining the ratio of data. Also test set overlapping will not happen between each iteration (Pedregosa et al., 2011). After splitting the datasets by using Cross Validation, we will extract images and their labels by using keys to get the train, testing and validation sets of data to feed and test the model.

Figure 26

Visualization of Cross Validation behaviour



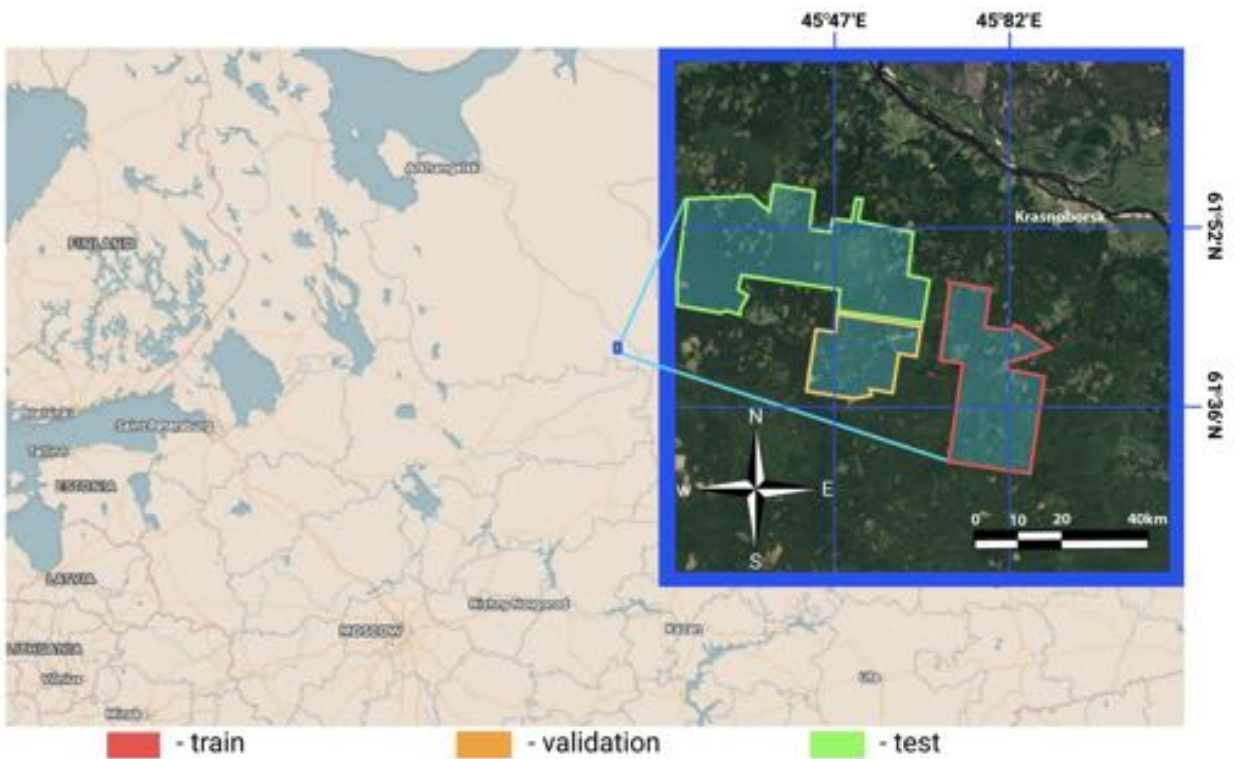
Note. Image shows the visualization of Cross Validation behaviour and how StratifiedKFold works. Adopted from “Scikit-Learn Machine Learning in Python,” by Pedregosa et al., 2011, *Journal of Machine Learning Research*, (12), pp. 2825-2830. Copyright 2011 by Scikit-Learn.

3.5.1.2 Region Based Splitting. This technique is unique to satellite image segmentation dataset splitting. In many papers the authors claim that to obtain more generalization in the model it is preferred to split the dataset based on the sub areas of the satellite images. For example, to train the model to detect the roof area of the building we should use different cities in the train, test, and validation images. The model could be trained on the images from the city of Chicago whereas the validation and testing could be performed on the images from New York and San Francisco. As shown in Figure 27, one such Region Based Splitting is performed on the

forest satellite images where the training, validation, and testing images cover completely different non overlapping regions of the forest.

Figure 27

Region based training, test, and validation splitting



Note. Figure shows how Region based splitting takes place for training, testing and validation data. Adapted from “MixChannel: Advanced Augmentation for Multispectral Satellite Images,” by Illarionova, S., Nesteruk, S., Shadrin, D., Ignatiev, V., Pukalchik, M., & Oseledets, I., 2021, *Remote Sensing*, 13(11), 2181, (<https://doi.org/10.3390/rs13112181>). Copyright 2021 by ResearchGate.

3.6 Data Statistics

3.6.1 Class Statistics

As shown in Figure 28, a three channel image shows multiclass labeled objects overlapped on the image. So in one of the dataset that has multiclass labels we would like to extract the statistics of all the classes. The colors of the classes are defined while building label data to train the model by using GroundWork tool.

Figure 28

Three band satellite images with multiple class labels



Note. Figure shows three band satellite images with multiple class labels. Adapted from *Neural network for satellite image segmentation*, by R. X.Jiang, 2018

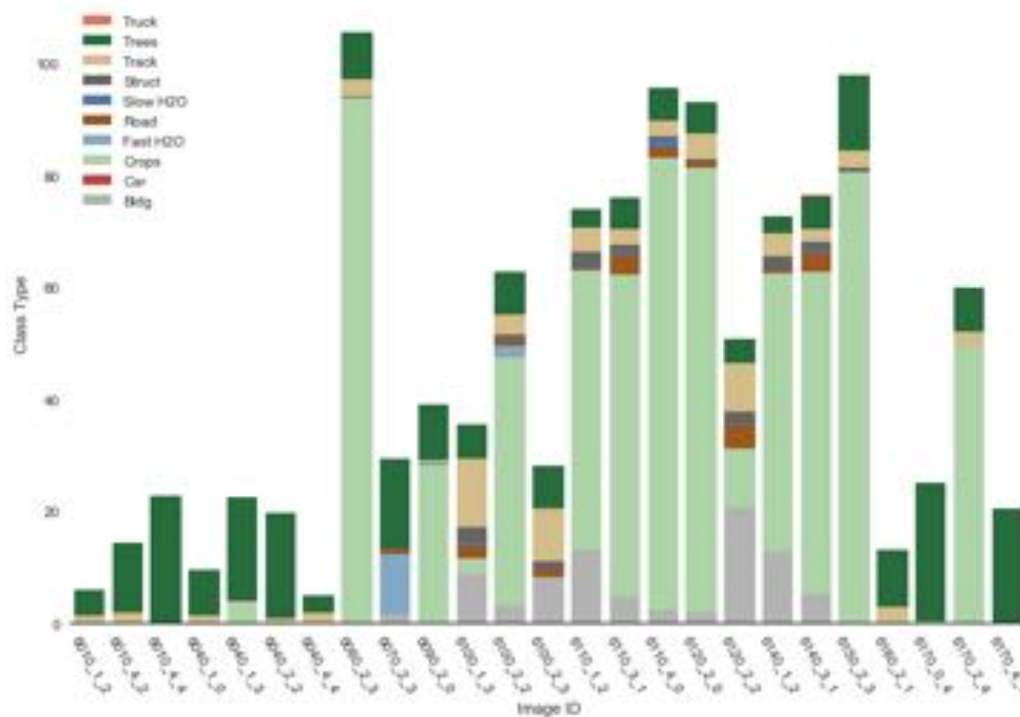
(<https://towardsdatascience.com/dstl-satellite-imagery-contest-on-kaggle-2f3ef7b8ac40>).

Copyright 2018 by Towards Data Science.

To analyze the Class Statistics of multiclass image masks, we select 25 image masks used in the training datasets and plot the contribution of each class in that mask. As shown in Figure 29, one such class statistics is shown where the dataset contains image masks with 10 classes. Based on the visual inspection it is clear that these images are mostly dominated by the trees class shown in dark green color class labels. So this kind of visualization is very useful to understand the dominance distribution of classes (Jiang, 2018).

Figure 29

Training data statistics of all classes



Note. Figure shows the training data statistics of all classes presented in the percentages .

Adapted from *Neural network for satellite image segmentation*, by Jiang, R. X. 2018

(<https://towardsdatascience.com/dstl-satellite-imagery-contest-on-kaggle-2f3ef7b8ac40>).

Copyright 2018 by Towards Data Science.

3.6.2 Summary

Until now we have collected enough satellite data from various sources and also generated our own data. By Visualizing those images we got to know that our data needs some preprocessing steps. Our raw satellite images were of various resolutions and datatypes which were converted into a fixed resolution and data type. We then performed data exploration cleaning and validation steps which involved studying the different bands of the satellite images and also possible reduction of bands which are understandable by humans, the cleaning and validation steps involved computing mean, meadia, standard deviation for all the images and getting rid of the outlier images that have very skewed statistics. The data transformation steps are necessary to train the model to adapt for various real world scenarios such as blurring, rotational and positional invariances. In the data preparation phase we split the dataset into training test and validation subsets. Splitting techniques are based on Random, KFold and Regional splitting techniques. Once the dataset is ready and prepared we would like to normalize the images that are very necessary for training Convolutional Networks efficiently.

4. Model Development

4.1 Model Proposals

The satellite image segmentation problem using the Deep Learning technique is new to the recent research industry. Before Dynamic Neural Network (DNN) came into picture, There were many segmentation or classification methods like Random Forest (RF) and Conditional Random Fields (CRF), which are based on feature extraction from images (Xiaolong Liu¹ et al., 2018). Some other methods are based on unsupervised clustering and supervised SVM classifiers. The Convolutional Neural Network is more popular in recent years for image recognition and speech recognition. Some papers also solved the classification problem by using CNN technology. There are some limitations of these existing methods for image segmentation, they are:

- Extraction of image features manually is a kind of tricky and tough job, while solving the segmentation problem.
- Though Convolution Neural Network (CNN) is popular in image segmentation it has some limitations like there is a chance of losing features while passing through the pooling and also input size is restricted because of the fully connected layer (Yoshihara et al., 2018).

To extract features manually we should have the domain knowledge of the images. It's a difficult task because images have different variations. The deep learning models on the other hand do a much better job in automatically extracting the abstract features from the images and inherently handle most of the computer vision tasks (Yusuf Artan, 2011). As a result, deep learning models have much higher performance and are easy to implement compared to the older

solutions. As mentioned by the author Long et al. (2015) in their paper, the limitations of a Convolutional Neural Network are solved by developing an architecture called Fully Convolutional Networks. During the time in 2014 where we have VGG, the RESNET Convolutional Network was only able to take a fixed size of input. But in our project we are working on the arbitrary size as input. Where, instead of predicting single numbers whether it is a tree or not. For every pixel in the image FCN allows us to predict the what is the class of those pixels. Fully Convolutional Network designed with downsampling and upsampling operations. U-Net is also a FCN with skip connections which helps to extract feature information from an encoder network (Ivanovsky, 2019).

4.1.1 Proposed Models for Satellite Image Segmentation

Before explaining the Fully Convolutional Network Architecture, first we are exploring the traditional Convolutional Neural Network to compare the difference between these two architectures since FCN is also a CNN. As shown in the Figure 31, CNN built for image classification there are two main fundamental parts in the CNN architecture they are:

- Feature Learning
 - Convolution, learn features from images.
 - apply ReLu non-linearity
 - Max pooling reduces dimensionality of the image
- Class Probabilities
 - Flatten, Conv and Pooling layers output to extract high-level features
 - Fully Connected layer uses high level features to classify image by using projection
 - By using Softmax Architecture will display output probability of a image

Dense layers (Fully Connected) project the input vector to output vector space thereby changing the size of the output vector. Series of dense layers can be used to gradually reduce or increase the vector dimensions to match the classification vector size. Also the dense layers have a matrix of learnable parameters that transform the features from one cartesian space to another. CNN uses these dense layers as the final layers of the network used in image classification. Just before the dense layers the image output of the last convolution layer is flattened to convert the image array to a one dimension vector and sent to the series of dense layers. Since we process the vectors of features it is hard to reconstruct the image by just using these dense layers due to a loss of information (Yoshihara et al., 2018).

In general, in the images every pixel is spatially related to its neighboring pixels. This spatial relationship is exploited in the Convolutional Neural Network. The filters used in Convolutional Neural Network have a receptive field which is much smaller than the image size for example, a 3×3 filter processes only 3×3 pixels in the given image thereby exploiting the spatial dependence. The subsequent layers will have a receptive field to process the output of the previous image. The combined receptive field of this layer will be much higher than the previous layer (Raghav, 2020).

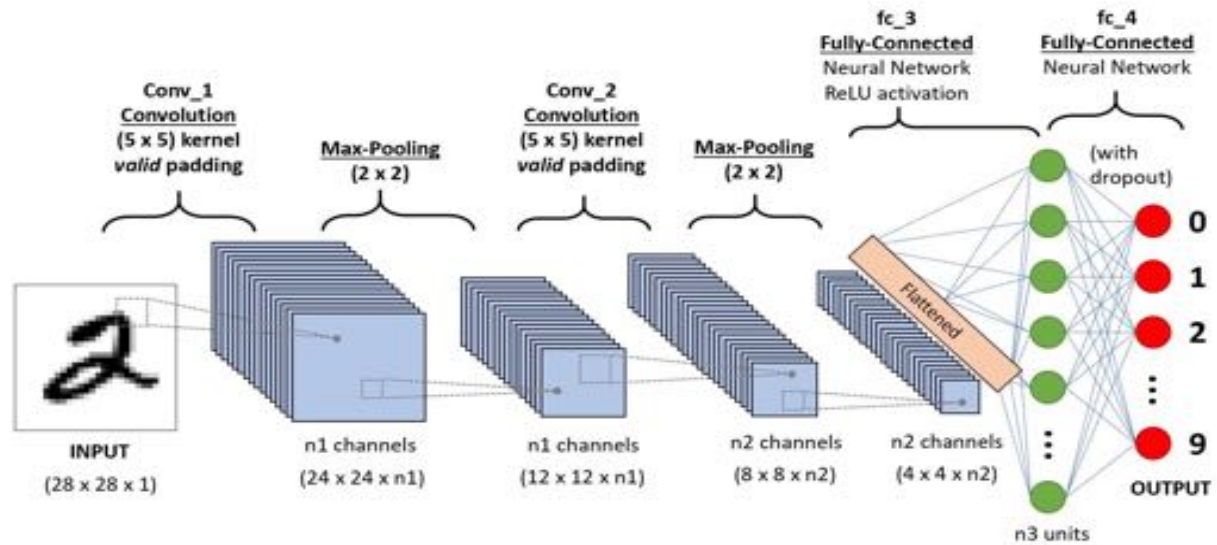
Each convolution operation per pixel consists of a dot product of the filter pixels and the image pixels that are in the receptive field of the filter. Entire image convolution consists of shifting the filter at a constant stride across the rows and then through the columns of the image thereby resulting in a convolved output image. In the beginning of the training, the filter pixels are initialized using random distribution and then adjusted based on the gradients derived in the back propagation. This process of adjusting the filter pixels or weights is called learning the

features of the image. More filters we use will allow us to learn more features present in the given image (Raghav, 2020).

As shown in Figure 30, after each Convolution operation we are applying or activating Non-linearity on the volume which is nothing but an output of a convolutional layer having height(h), width (w), and the depth which is nothing but the number of filters (d). On this volume we are applying element-wise non-linearity. To do that architecture will use the Rectified Linear Unit (ReLU) which takes any real numbers as input and which shifts numbers which are less than zero means negative numbers to zero and numbers greater than zero it sweeps as it is. Non-linear operation replaces all negative values by zero. To solve more complex problems, this architecture Introduces nonlinearities (Raghav, 2020).

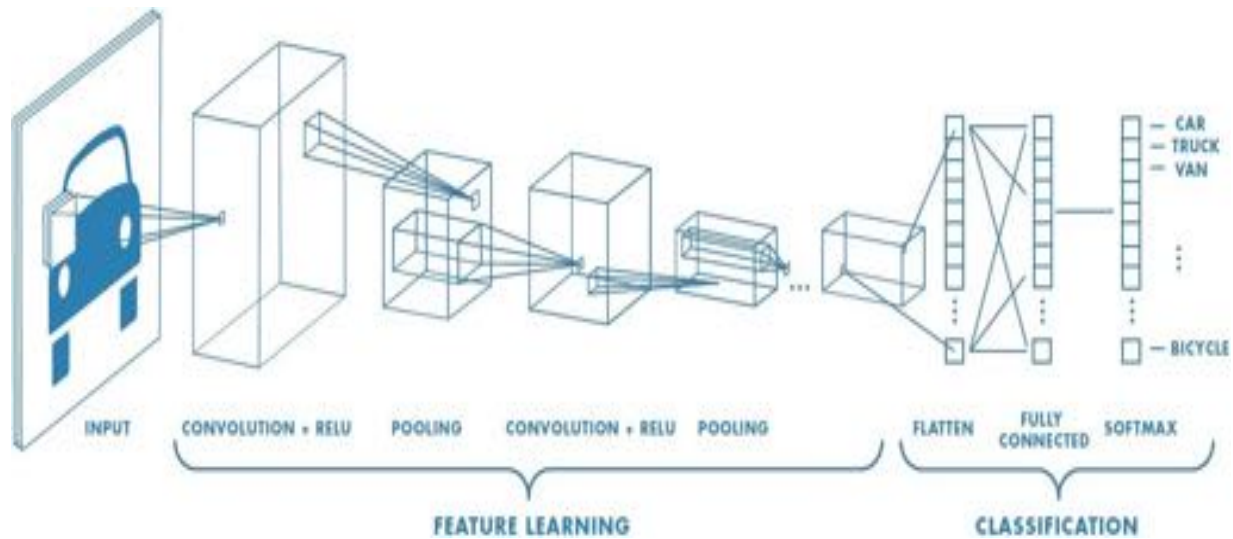
Pooling operation is done before applying the next convolutional layer and repeating the same step over the whole architecture. Pooling allows us to reduce image size which is called down sampling, conceptually pooling in a given local receptive field either selects the output of the neurons that is maximum which is called max pooling or the average of all the pixels in the receptive field called average pooling (Raghav, 2020). We are learning the hierarchy of features by layering Convolution, Non-linearity, and the Pooling operations.

Finally, take those extracted features and classify the image. The fully connected dense layer receives the final set of features and projects them on to output vector space; multiple of these projections could be performed to get to a size of output space which is the same as the classification vector. And finally this vector is processed by the Softmax layer to convert the float values to probability values thereby giving the probability of the classes.

Figure 30*Traditional CNN Architecture*

Note. Traditional Convolutional Neural Network Architecture. Adapted from *What is Convolutional Neural Network Architecture*, by P. Ratan, 2020

(<https://www.analyticsvidhya.com/blog/2020/10/what-is-the-convolutional-neural-network-architecture/>). Copyright 2020 by Analytics Vidhya.

Figure 31*Steps in CNN architecture*

Note. Figure shows the main two steps in CNN architecture. Adapted from *Understanding of Convolutional Neural Network (CNN) - Deep Learning*, by P. Raghav, 2018

(<https://medium.com/@RaghavPrabhu/understanding-of-convolutional-neural-network-cnn-deep-learning-99760835f148>). Copyright 2018 by Medium.

4.1.1.1 Fully Convolutional Network (FCN). FCN is an extension of CNN which only has the Convolutions, Non-linear, and Pooling steps. To overcome the limitations of the Convolutional Neural Networks like RESNET and VGGNET, The author Long et al. (2015), proposed an architecture called Fully Convolutional Network (FCN).

The important thing in this network is that it can handle the arbitrary size input instead of fixed size. The dense network (Fully Connected Network) in the CNN is removed in the FCN to make predictions on arbitrary size input. FCN uses the technique called Up sampling - Transposed Convolution/Deconvolution. The Upsampling approach enables this architecture to

decode the original resolution. And the skip connection is for branching the input, sending it to the next stage without processing, which basically skips the processing. This Fully Convolutional Network is trying to do pixel-wise semantic means that each pixel which identifies the class it belongs to is different from previous networks like RESNET and VGGNET which do classification of the image to classify the image dog or cat. In Semantic Segmentation each pixel needs to have a class that's a huge difference (Long et al., 2015). Main parts of the Fully Convolutional Networks (FCN) are listed below:

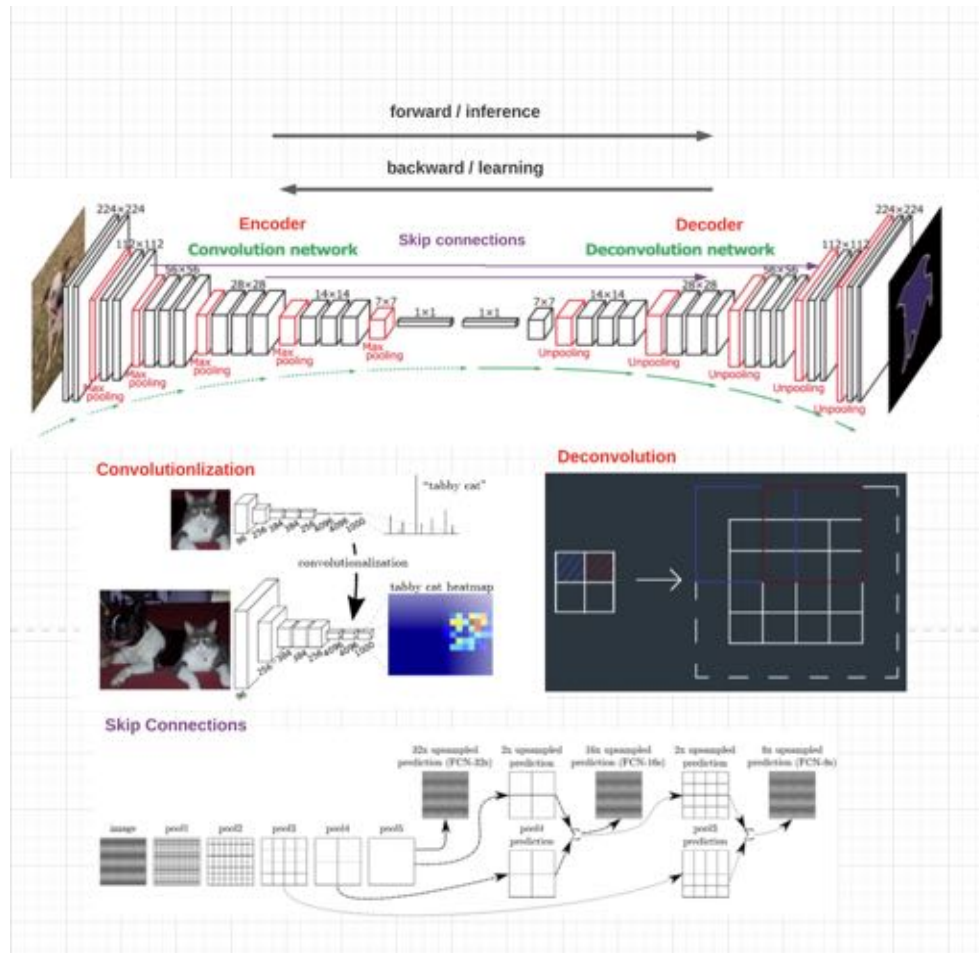
- Architecture
 - Can take arbitrary size as input instead of fixed size
 - Up sampling-Transposed Convolution/ Deconvolution/Fractionally-strided convolution
 - Skip connection, branching the input and sending it to the next stage without processing.
- Pixel-wise semantic segmentation

As shown in Figure 32, a Fully Convolutional Network consists of an Encoder and corresponding Decoder layer (Yoshihara et al., 2018). Encoder will project a lower dimension. Decoder will reconstruct the original image. Specific thing to note here for FCN model is that it consists of many convolution blocks in the encoder part of the model but only one upsampling decoder convolution block. As shown in Figure 32, instead of flatten it, we are using $1 \times 1 \times d$ dimensions in the spatial dimension. Our main goal is to do Semantic Segmentation on the satellite image, for this we need to scale it back to the original image dimension since each pixel has a class. We are rescaling it to the original image dimension by using the Decoder part which

is called upsampling in the Deconvolution network. As we are using a 3D vector without flattening it, there is a chance of losing information for this reason some FCN variations have two skip connections to take the previous information and connect to the deconvolution network.

Figure 32

Fully Convolutional Network Architecture



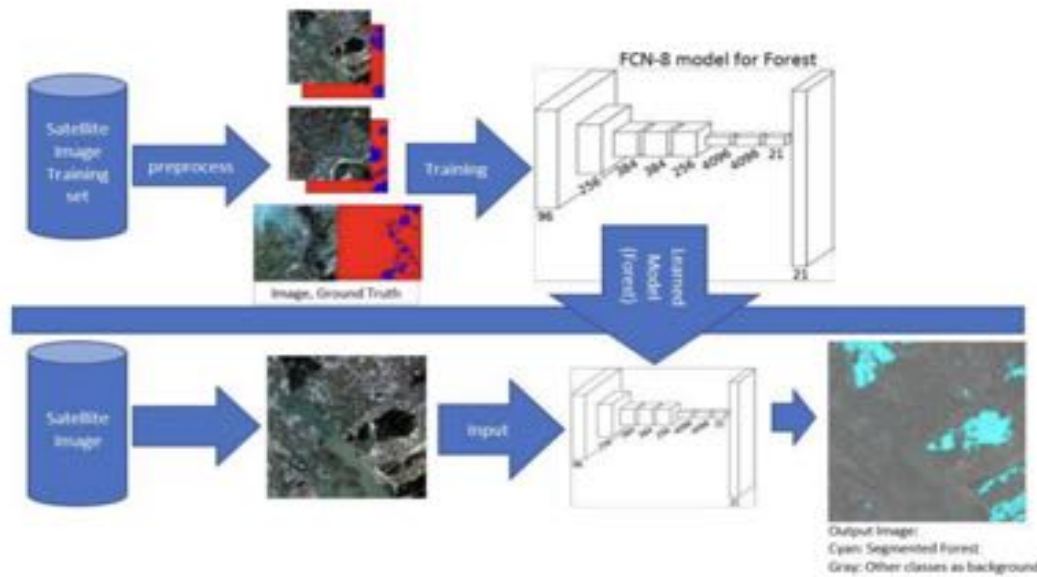
Note. Figures show Fully Convolutional Neural Network Architecture and steps of Convolution and Deconvolutional processes. Adapted from "Learning Deconvolution Network for Semantic Segmentation," by Noh, H., Hong, and Han, B., 2015, *IEEE International Conference on Computer Vision (ICCV)*, 2015, pp. 1520-1528, (doi: 10.1109/ICCV.2015.178).

As shown in Figure 32, network structure which shows multi-class segmentation by each pixel class in the image with FCN architecture. As said earlier, the FCN consists of an Encoder and Decoder network. In the architecture, we are considering 256 x 256 satellite image size as input.

The Encoder (Convolution) part is the same as the typical CNN architecture. Which has the repeated batch normalizations and convolutions followed by the nonlinearity by using a rectified linear unit (ReLU) before entering into the next convolution layer, we are applying a max pooling operation to reduce the dimensionality (down sampling). For concat processing simplification, in the Convolutions the padding value is always set to one so that height x weight of the feature map should be constant. As shown in Figure 33, the Training part of the FCN contains the input image and the mask image. In this model, we are trying to segment the Forest satellite image data. In the prediction part, we are giving an input image of size 256 x 256 and segmenting it by using the FCN8 model.

Figure 33

Training and Prediction stage of the FCN model



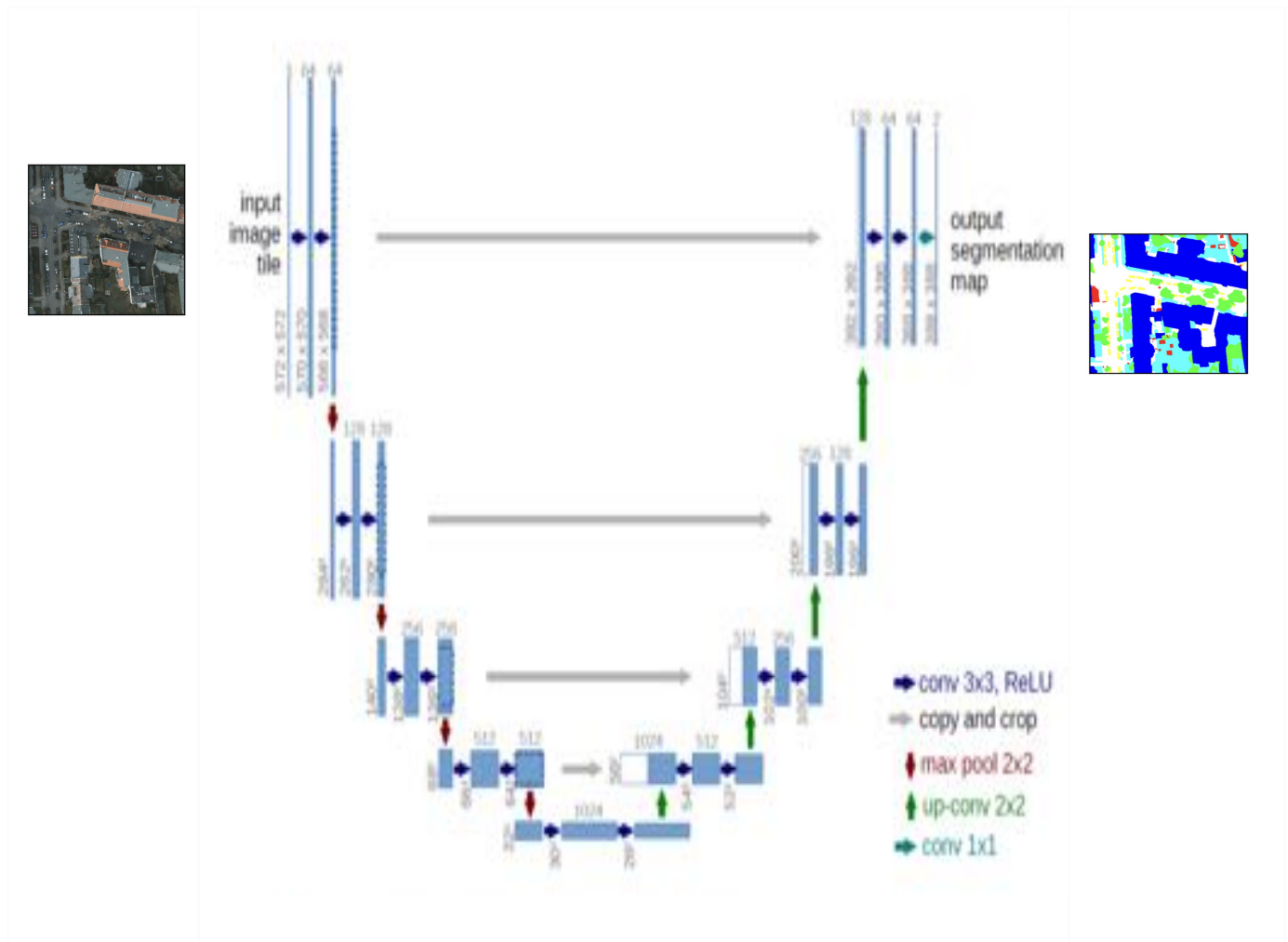
Note. A figure shows the proposed approach of a Fully Convolutional Layer for Satellite Image Segmentation. The upper part is the training part and the lower part is the prediction part.

Adopted from “*LULC Segmentation of RGB Satellite Image Using FCN-8*,” by Nayem et al., 2020, *CoRR*, *abs/2008.10736*. (<https://arxiv.org/abs/2008.10736>). Copyright 2020 by CORR.

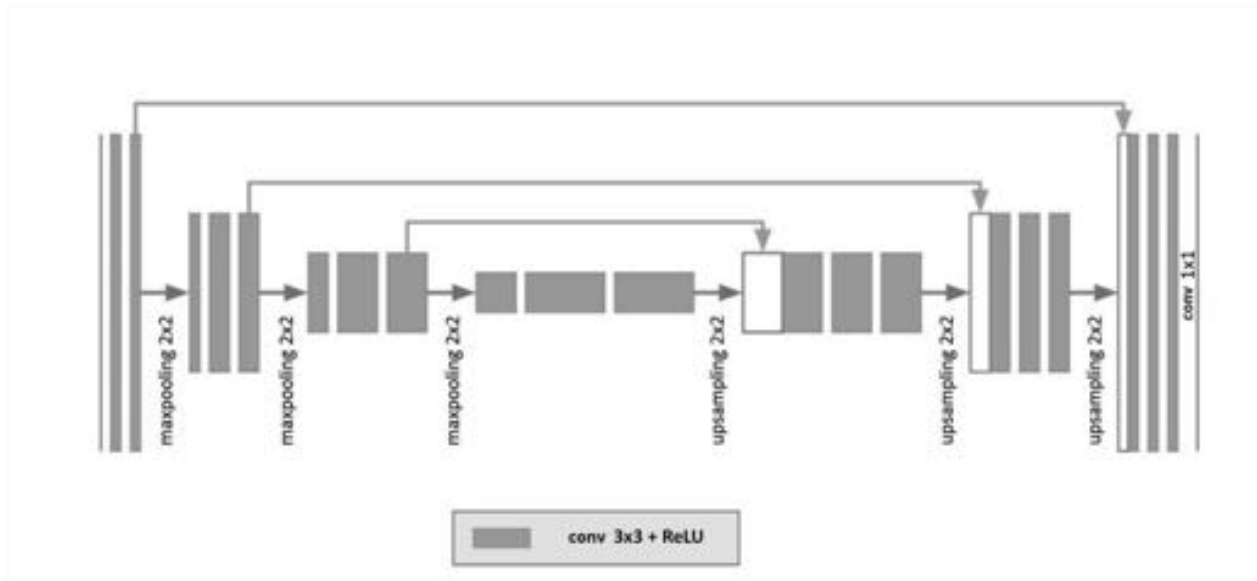
4.1.1.2 U-NET. U-NET is also an FCN that was proposed and effectively used for image segmentation in medical applications (Ronneberger et al., 2015). It’s called U-NET because it is in U shape. For Satellite Image Segmentation, we are using the same method. U-NET is constructed on the Fully Convolutional Network. As shown in Figure 35, we have Convolutions, ReLU activation, and max Pooling process with these in the extra added methods in U-NET architecture includes concatenation or skip connections from contracting path to the expanding path and multiple upsampling layers (Ronneberger et al., 2015).

As shown in Figure 34, the first half of the architecture is called the encoder part where the image is downsampled to smaller dimensions resulting in arrays that capture only the features of the image. The other half of the architecture is called the decoder part that uses the smaller dimension array of features to reconstruct the output masks in multiple stages. In between, there is some concatenation between the encoder and decoder part which is called skip connection. From these we will get localization information that makes semantic segmentation in U-NET (Chhor et al., 2017).

In the architecture they have used the image size of 572×572 and they have set the filtered value to 64 which means 64 features. But we can modify it based on our requirements. So we are extracting features from the multiple convolutions by using the encoder part. To capture the characteristics of each pixel in the image ReLU and Pooling are used on each Convolution. On the right side a symmetric expanding or decoder part contains a sequence of upconvolutions or up samples and concatenations that are necessary for reconstruction of the mask which will be of the same size as the input image (Ronneberger et al., 2015).

Figure 34*U-NET Architecture*

Note. U-NET architecture. Adopted from “U-NET: Convolutional networks for biomedical image segmentation. Medical Image Computing and Computer-Assisted Intervention,” by Olaf Ronneberger et al., 2015, *Lecture Notes in Computer Science*, 234–241. (https://doi.org/10.1007/978-3-319-24574-4_28). Copyright by Springer 2015

Figure 35*U-NET Encoder and Decoder Model*

Note. Figure shows the U-NET encoder and Decoder. Adopted from “Building Detection on Aerial Images Using U-NET Neural Networks,” by Ivanovsky et al., *2019 24th Conference of Open Innovations Association (FRUCT)*. (<https://doi.org/10.23919/fruct.2019.8711930>).

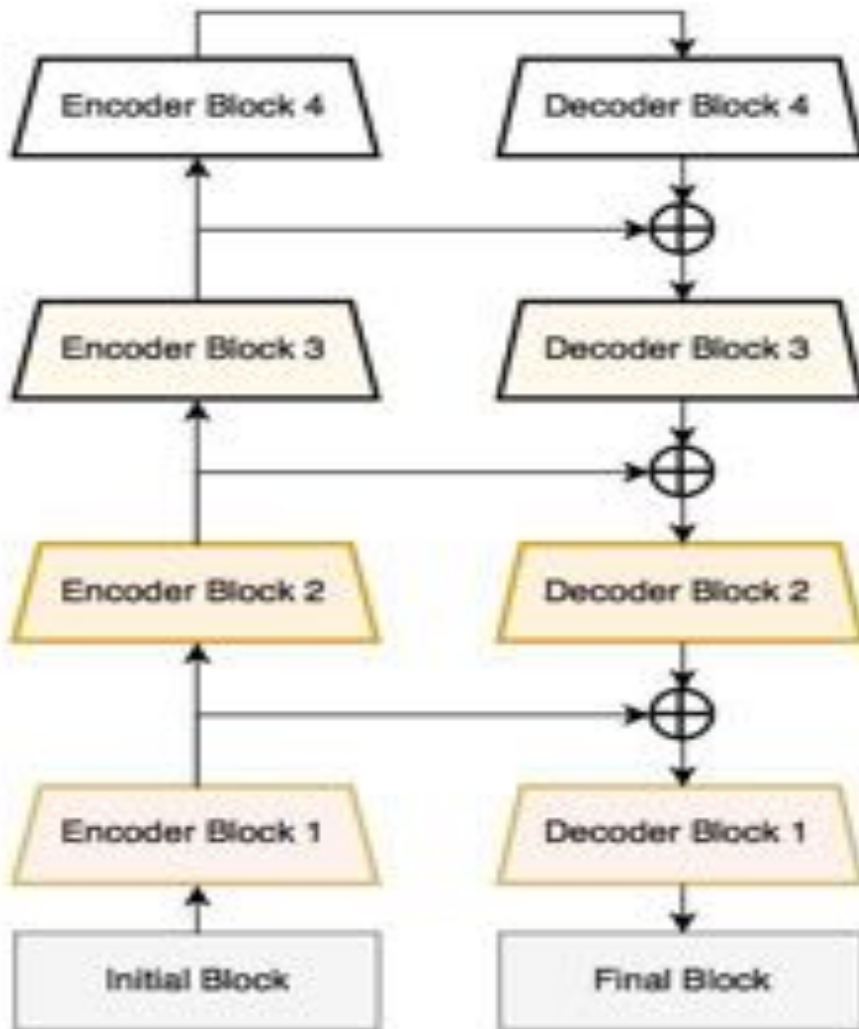
Copyright by FRUCT 2019.

4.1.1.3 LinkNet. LinkNet is a residual model which converts U-NETs normal stacking method to adding for feature extraction. Author Chaurasia and Culurciello (2017) have proposed the main difference in the LinkNet architecture which is that we can train the model without increasing the significant number of parameters. So they are using only 11.5 million parameters to train the model and 21.2 GigaFLOPs (GFLOPs) are used in this model to process an image of size 640 x 360 x 3. This architecture is introduced to minimize the number of parameters so that the segmentation process should work in real time.

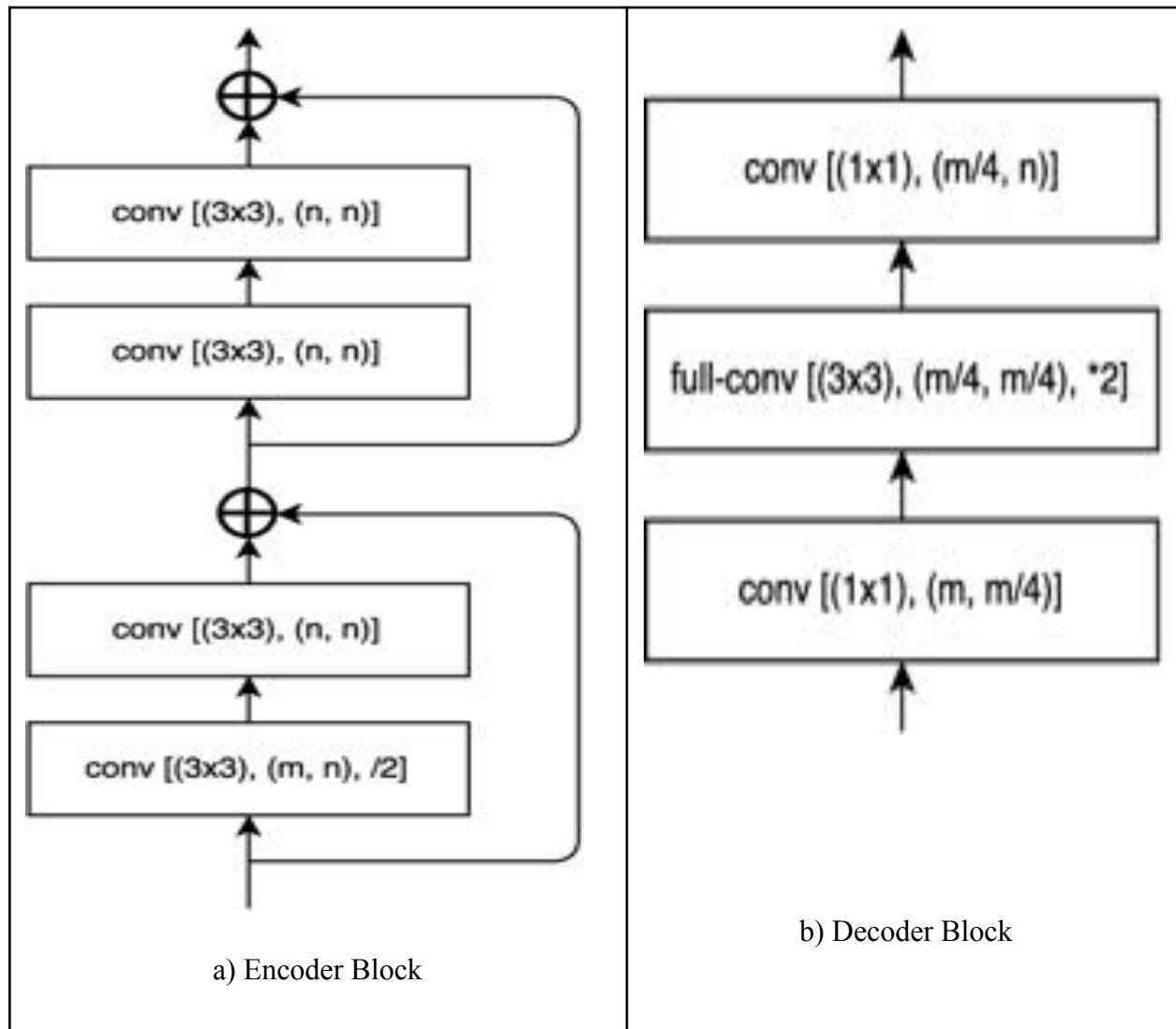
In Satellite Image Segmentation we are also exploring the LinkNet method for semantic segmentation purposes. As we saw in the U-NET which has the Encoder and Decoder

architecture. We can see the same networks (Encoder and Decoder) in LinkNet also. As shown in Figure 36, there are four blocks in the subnets, which means Encoder and Decoder. Left side of the architecture shows the Encoder part and the right side of the architecture determines the Decoder part. As shown in Figure 37, each encoder block of the architecture contains a 3×3 filter of four convolutions since it has the four blocks, two merging operations, and for pooling it has the 2×2 max pooling filter. As shown in Figure 37, each decoder block of the LinkNet architecture has the same structure as the Encoder part but instead of merging and max pooling operations it has 2×2 filter upsampling operation (Ivanovsky et al., 2019).

As shown in Figure 36, Before moving to the first encoder part, the architecture initializes the input block which performs batch normalization and the nonlinearity (ReLU) activation functions of two operations with 2×2 filters of convolutions layer and max pooling. After the last Decoder part, the architecture is assigned a final block which contains a sequence of operations like the upsampling process, two blocks of batch normalization, Relu nonlinearity, and a convolution layer. Assigned 2×2 filters for all these final block operations (Ivanovsky et al., 2019).

Figure 36*LinkNet Architecture*

Note. Figure shows the Architecture of the LinkNet architecture. Adopted from “LinkNet: Exploiting encoder representations for efficient semantic segmentation,” by Chaurasia, A., & Culurciello, E. (2017), *2017 IEEE Visual Communications and Image Processing (VCIP)*. Published. (<https://doi.org/10.1109/vcip.2017.8305148>)

Figure 37*Convolutional modules in encoder-block*

Note. Figure shows the Architecture of the LinkNet architecture. Figure a) Encoder Block and b) Decoder Block. Adopted from “LinkNet: Exploiting encoder representations for efficient semantic segmentation,” by Chaurasia, A., & Culurciello, E. (2017), *2017 IEEE Visual Communications and Image Processing (VCIP)*. Published. (<https://doi.org/10.1109/vcip.2017.8305148>)

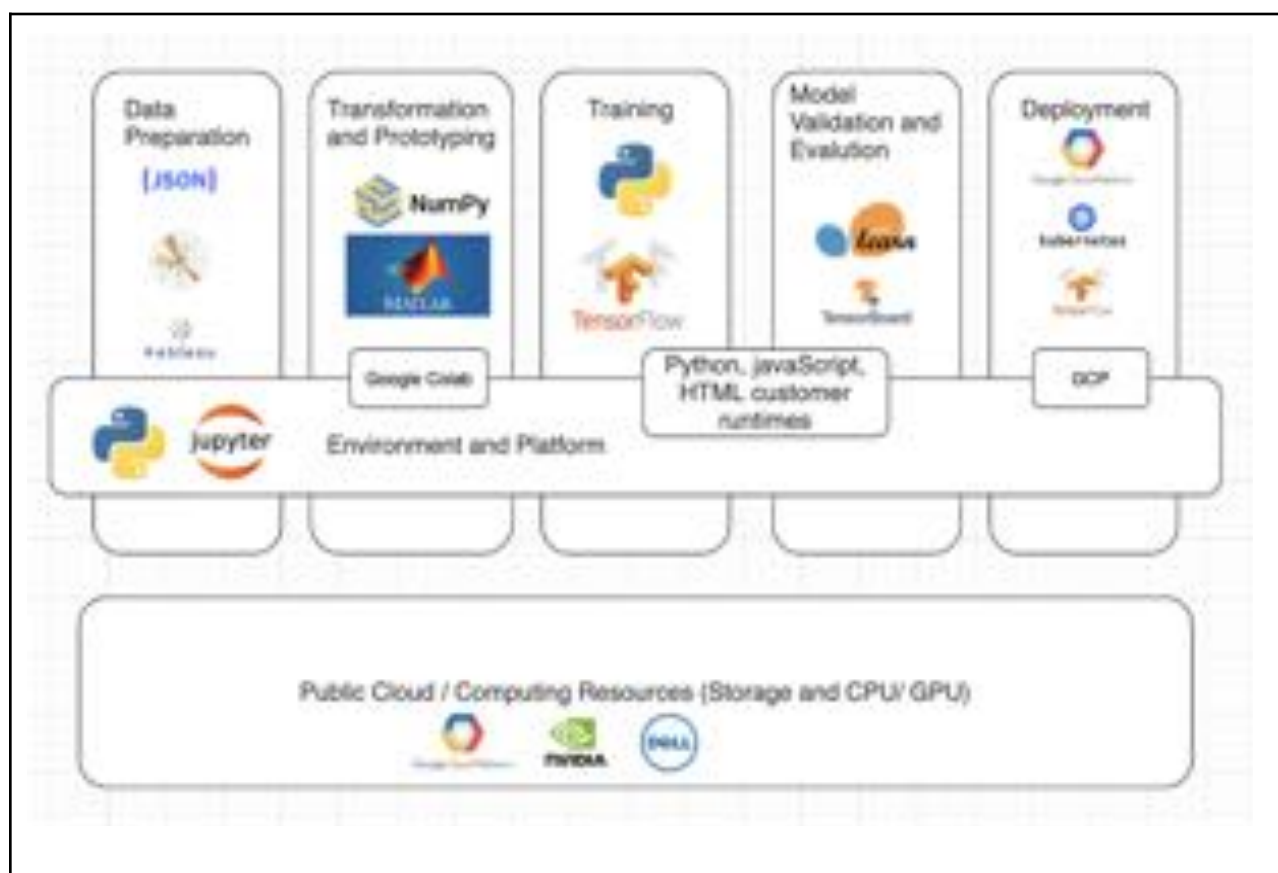
4.2 Model Supports

In this section we give an overview of all the tools and platforms used in the satellite image segmentation project to implement and deploy the deep learning models. As shown in Figure 38, the base platform we use will be Google cloud platform that provides the compute resources and storage hardware required for the project along with the basic operating system. We also use the Google cloud services and applications to deploy the final inference model packaged in a kubernetes container. As shown in Figure 39, the flow diagram of the project shows more detailed functionalities we implement at each stage.

There are several tools that are used at every stage of the project. The data preparation mainly deals with cleaning up and pre-processing large datasets. We use serialization tools like JSON and few visualization tools like Matplotlib and Tableau in this stage. The transformation and prototyping involves some amount of image processing that requires Numpy and Matlab tools. The model is created and trained using TensorFlow. We use Scikit learn and TensorBoard tools to validate the loss curves and other training performance metrics. The final deployment requires APIs and services in GCP aligned with Kubernetes.

Figure 38

Deep Learning models tools and implementation schema



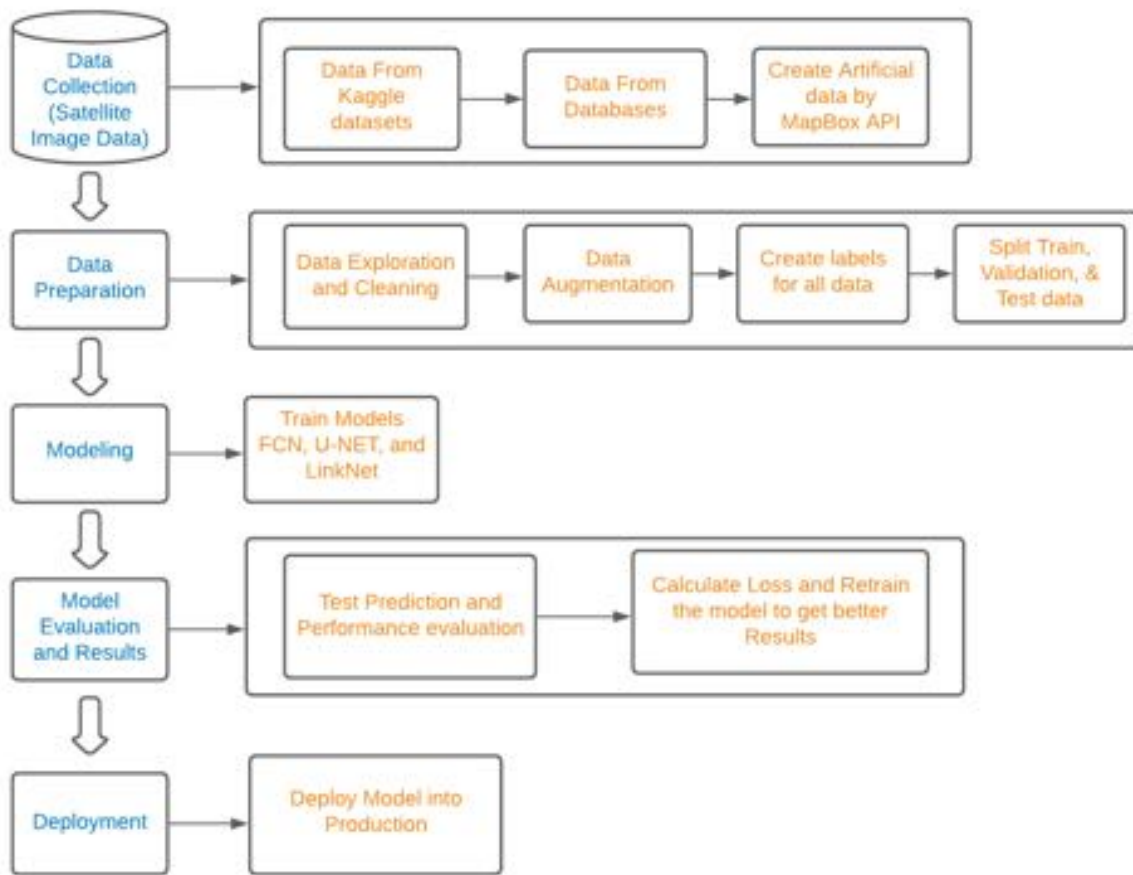
Note. The Figure shows Detailed information of Deep Learning models, tools and implementation schema.

As shown in Figure 39, to achieve the objective of this project, we need to first collect the datasets of satellite images. The datasets were collected from multiple sources like Kaggle and public databases like Inria and Spacenet. For more generalization and to introduce more variations, we also created virtual satellite images using MapBox API. Once the dataset was collected, we stored the data in the Google Cloud Storage and preprocessed it. The preprocessing stage involved data cleanup and augmentation processes. We decided to build multiple deep

learning models to train on datasets like FCN, UNET and LinkNet. In the evaluation phase we compared the performance of each model under various evaluation metrics. Finally the best performing model was selected and used for deployment purposes.

Figure 39

Data Flow Diagram of the Project



Note. Figure shows the Overview of the project Data Flow diagram with multiple stages.

4.3 Model Comparison and Justification

In this section we are comparing models based on their strength and limitations. In this project we are using deep learning models which are Fully Convolutional Network (FCN), U-NET, and LinkNet. The targeted problem for each model would be Semantic Segmentation of Multi Class Satellite images. Based on the strengths and limitations we can conclude that the skip connections present in the U-NET and LinkNet models play a crucial role in image mask reconstruction. Having multiple decoder blocks compared to a single block in FCN provides a more staged approach in reconstruction. Also the model performs well when the decoder blocks have learnable parameters instead of static parameters like FCN.

In most cases U-NET and LinkNet perform much better than the FCN because of the reasons mentioned above but between them, LinkNet tends to be more superior since it has lesser learning parameters that will significantly decrease the training time of the model. As shown in Table 4, which gives the in-depth information about the targeted problem, Approach, Strengths, and limitations of the FCN model, U-NET, and LinkNet models.

Table 4*Models Strength and Limitations*

Model	Targeted Problem	Approach	Strength	Limitations
FCN	Semantic Segmentation of Multi Class Satellite images	Deep Learning	1) Input Image size 2) Spatial Information 3) Computational cost	1) Up Samples only once 2) Not all FCN have skip connections 3) Absent uses bilinear interpolation (static)
U-NET	Semantic Segmentation of Multi Class Satellite images	Deep Learning	1) Supports all the advantages of FCN 2) Multiple upsampling layers 3) Uses learnable weight Filters 4) Multiple skip connections & concatenates	1) Optimal depth is apriori unknown 2) Unnecessarily restrictive fusion scheme by skip connections (Zhou et al., 2020)
LinkNet	Semantic Segmentation of Multi Class Satellite images	Deep Learning	1) Uses Less parameters 2) It has Residual model (res-block) 3) Uses skip connections within the bocks 4) Use in real time applications	1) Unnecessarily restrictive fusion scheme by skip connections (Zhou et al., 2020)

Note. Table shows target problem, approach, strength, and limitations of all models

As shown in Table 5, here we describe the basic structural differences between the models. The main structural components are the decoder blocks, learnable filter configuration and the concatenation connections between the corresponding upsampling and downsampling

blocks.

Table 5

Comparison of all models based on parameters

Parameters	FCN	U-NET	LinkNet
Decoder / upsampling	Only one or fewer than encoders	Same number as encoders	The same number of encoders
Skip Connections	Not present, some variants have two	Every decoder connected to the corresponding encoder	Every decoder connected to the corresponding encoder
Learnable filters in upsampling	Absent uses bilinear interpolation (static)	Present and dynamic	Present and dynamic

Note. Comparison of all three models based on the three parameters

4.3.1 Justification to use the Models

In the first level of justification, we would like to provide reasons for choosing deep learning models over traditional machine learning modes. The traditional machine learning models require a lot of hand tuning and manual feature extraction which is not generic and cannot be scaled across multiple datasets of the same class. For example, the hand tuned model trained on city and building satellite images may not work with forest satellite images. Unlike the traditional ML models, the deep learning models learn and generalize extremely well if they are trained on multiple datasets.

In the next level of justification, we would like to provide reasons for choosing FCN, UNET and LinkNet as our models instead of various other deep learning models and the reasons are described as follows:

- Arbitrary image size
 - The models are very flexible in terms of image sizes. The input or training image sizes can be of arbitrary resolution
- Number of training images
 - The Unet model requires significantly less training samples to train for required accuracy.
 - The variants of FCN with skip connections and LinkNet also use a lesser number of training samples but slightly more than Unet.
- These models are basically encoder-decoder style networks dealing with image IO

4.4 Model Evaluation Methods

After training a model with satellite images, our next step is to evaluate those models based on the test satellite images. We are using some evaluation metrics to analyze how accurately our model is segmenting the classes.


4.4.1 Accuracy (A)

Accuracy is used to check the percentage of accurate classifications done on the test data. Accuracy metric is the ratio of the number of right objects classified to the total number of objects (Ivanovsky et al., 2019). As shown in Figure 40, the yellow color is for True Negative (TN), green is for True Positive (TP), light green is for False Negative (FN), and red is for False Positive results (FP) (Long, 2021).

Figure 40*Accuracy metric*

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$$

Fraction predicted correctly



Note. Figure shows the formula of Accuracy score metric. Adopted from *Understanding Data Science Classification Metrics in Scikit-Learn in Python*, by Long, A., 2021.

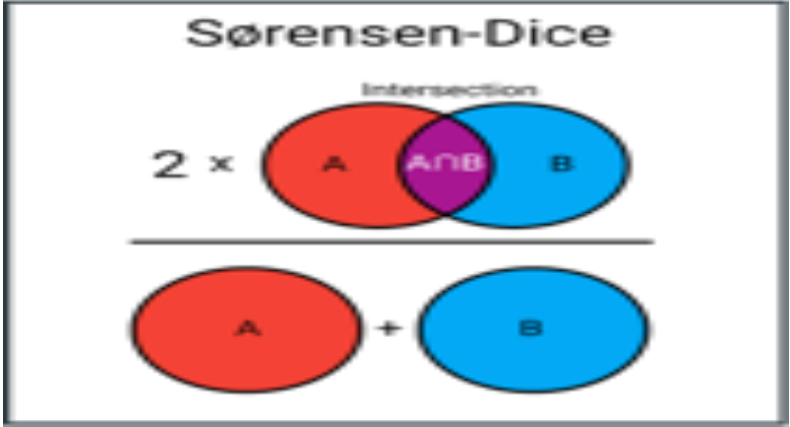
(<https://towardsdatascience.com/understanding-data-science-classification-metrics-in-scikit-learn-in-python-3bc336865019>). Copyright by Medium 2021.

4.4.2 Sorensen-Dice coefficient (DSC)

The Sorensen-Dice coefficient (DSC) metric tries to quantify the segmentation results. Which is defined as comparison between expert markups and predicted masks. To show the similarity degree between two sets (expert markups and predicted masks), the Sorensen-Dice coefficient (DSC) indicator considers the interval values of $[0, 1]$. As shown in Figure 41, the numerator represents multiplication of intersection $|A \cap B|$. Denominator represents the sum of expert markup A and predicted masks B (Ivanovsky et al., 2019). DSC is also like the Jaccard Index which is used to measure the similarity and diversity of the data (Grootendorst, 2021).

Figure 41

Sorensen-Dice Coefficient formula



$$D(x, y) = \frac{2 |x \cap y|}{|x| + |y|}$$

Note. Figure shows the Sorensen-Dice Coefficient formula. Adopted from 9 Distance Measures in Data Science | Towards Data Science, by Grootendorst, M., 2021.

(<https://towardsdatascience.com/9-distance-measures-in-data-science-918109d069fa>). Copyright by Medium 2021.

4.4.3 Confusion Matrix

Confusion matrix is an ordered structure of classification metrics, where the rows depict the predicted values and the columns depict the reference values or the ground truth. In a binary classifier, the true class is classified or labelled as one and the false class is classified or labelled as zero so that the target variable has two values: true and false (Bhandari, 2021). For instance

let's take an example: we want to segment the binary class of Road as positive and not road (background) as negative. The Confusion matrix rows and column values listed as:

- True Positives (TP) says the model predicted positive values correctly.
- True Negatives (TN) says the model predicted negative values correctly.
- False Negatives (FN) says the model incorrectly predicted negative for positive values.
- False Positive (FP) says the model incorrectly predicted positive for the negative values.

As shown in Figure 42, The top left corner contains True Positives (TP) which means the model identifies labels correctly which means road pixels are identified correctly. The True Negative (TN) in the bottom corner of the figure represents a model classified and predicted correctly for the background and labelled as zero. The bottom left hand corner contains False Negatives (FN) represents, when the positive result is there but the algorithm says negative result that means the image has the road (positive) but the algorithm is labelled as the background (negative). Lastly, the top right hand corner contains False Positives (FP) which represent the algorithm mistakenly classified negative results as positive. That means the image has the background (negative) but the algorithm is labelled as the road (positive). The model is evaluated based on these metrics and generalizes how well the model is performing on the test images (Muruganandham, 2016).

Figure 42

Confusion Matrix parameters

		ACTUAL VALUES	
		POSITIVE	NEGATIVE
PREDICTED VALUES	POSITIVE	TP	FP
	NEGATIVE	FN	TN

Note. Figure shows the Confusion Matrix. Adapted from *Confusion Matrix for Machine Learning*, by Bhandari, A., 2021.

(<https://www.analyticsvidhya.com/blog/2020/04/confusion-matrix-machine-learning/>). Copyright by Analyticsvidhya 2021.

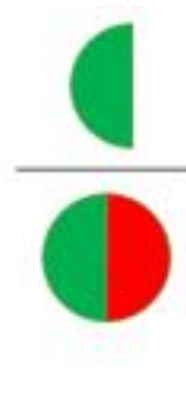
4.4.4 Precision

Precision metric calculates the purity of positive prediction labels compared to the actual ground truth labels. Using Precision to see how many of the segmented objects in the image are the same as the actual labeled objects (Long, 2021). As shown in Figure 43, the Precision metric formula is defined by True Positive and False positive values. Precision is also called the true predicted value.

Figure 43*Precision metric*

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} = \frac{\text{Green Semi-Circle}}{\text{Green and Red Circle}}$$

Fraction of predicted positives that are actually positive




Note. Figure shows the formula of Accuracy score metric. Adopted from *Understanding Data Science Classification Metrics in Scikit-Learn in Python*, by Long, A., 2021.

(<https://towardsdatascience.com/understanding-data-science-classification-metrics-in-scikit-learn-in-python-3bc336865019>). Copyright by Medium 2021.

4.4.5 Recall

As shown in Figure 44, the Recall metric calculates the ratio of accurate segmented labels to the total number of predicted labels. Which also defines the completeness of positive predicted labels to the ground truth labels (Long, 2021). We can also say, how many objects we captured out of all the multiclass labelled objects in the ground truth image. As shown in Figure 44, True Positive and False Negative relationships. Recall is also called as True positive rate.

Figure 44*Recall metric*

$$\text{Recall (Sensitivity)} = \frac{TP}{TP + FN} = \frac{\text{Fraction of positives predicted correctly}}{\text{Total positives (TP + FN)}}$$


Note. Figure shows the formula of Accuracy score metric. Adopted from *Understanding Data Science Classification Metrics in Scikit-Learn in Python*, by Long, A., 2021.

(<https://towardsdatascience.com/understanding-data-science-classification-metrics-in-scikit-learn-in-python-3bc336865019>). Copyright by Medium 2021.

4.4.6 Intersection over Union or Jaccard Index (IOU)

As shown in Figure 45, the Jaccard Index or Intersection Over Union calculates the area that is common between the predicted pixel classes and the reference pixel classes to the total area of the envelope that consists of both predicted and reference pixel classes (Tiu, 2021). The Jaccard Index is also used as the loss function in segmentation. As shown in Figure 45, the intersection and union part of the both labels are defined by rectangles (Tiu, 2021).

Figure 45*Jaccard Index Formula*

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

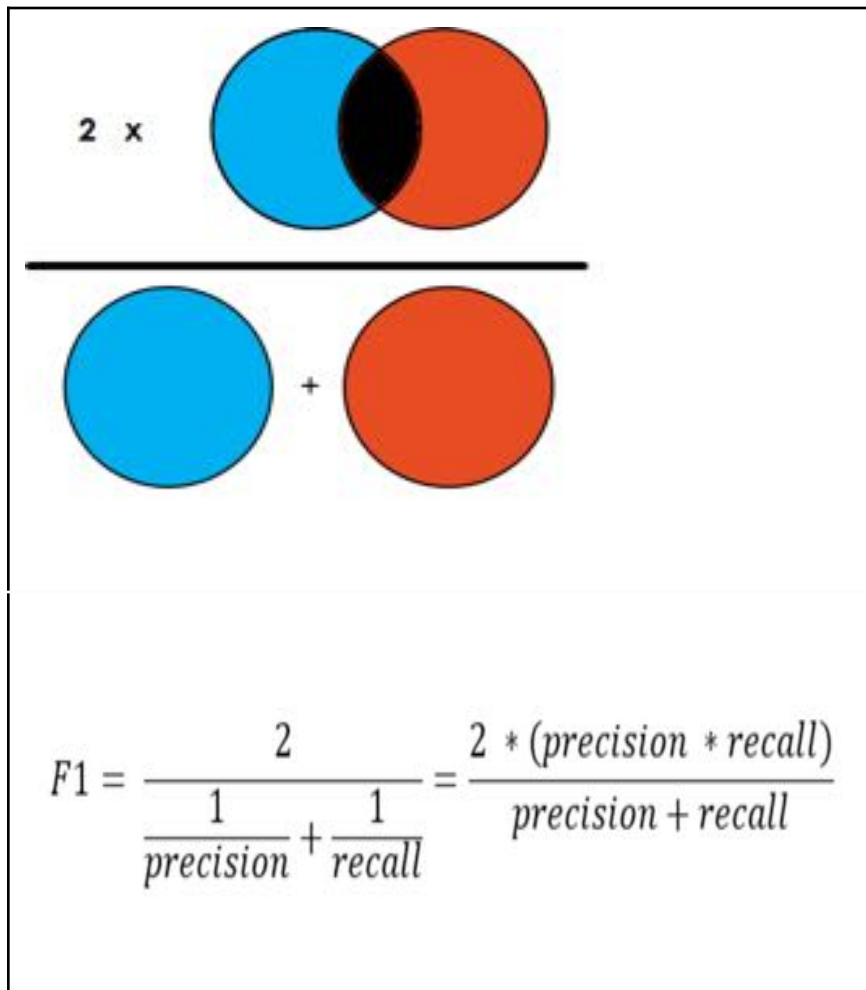
Note. Figures show the Jaccard Index Formula. Adopted from *Metrics to Evaluate your Semantic Segmentation Model*, by Tiu, E., 2021.

(<https://towardsdatascience.com/metrics-to-evaluate-your-semantic-segmentation-model-6bcb99639aa2>). Copyright by Towards Datascience 2021.

4.4.7 F1 Score

F1 score combines both the Precision metric which is the true predicted value and the Recall which is the true positive rate. The precision and recall are inversely proportional to each other, where the increase in precision of the model requires the recall to be decreased. This problem is solved by using the F1 score metric. As shown in Figure 46, F1 score captures both values in one (Bhandari, 2021).

Figure 46*F1 score or Dice Coefficient formula*



$$F1 = \frac{2}{\frac{1}{precision} + \frac{1}{recall}} = \frac{2 * (precision * recall)}{precision + recall}$$

Note. Figures show the F1 score Formula. Adopted from *Metrics to Evaluate your Semantic Segmentation Model*, by Tiu, E., 2021.

(<https://towardsdatascience.com/metrics-to-evaluate-your-semantic-segmentation-model-6bcb99639aa2>). Copyright by Towards Datascience 2021.

4.4.8 Receiver Operating Characteristic (ROC) curve

This metric is used for binary classification problems, since we are attempting both binary and multiclass segmentation problems, we are trying to see the ROC curve result for

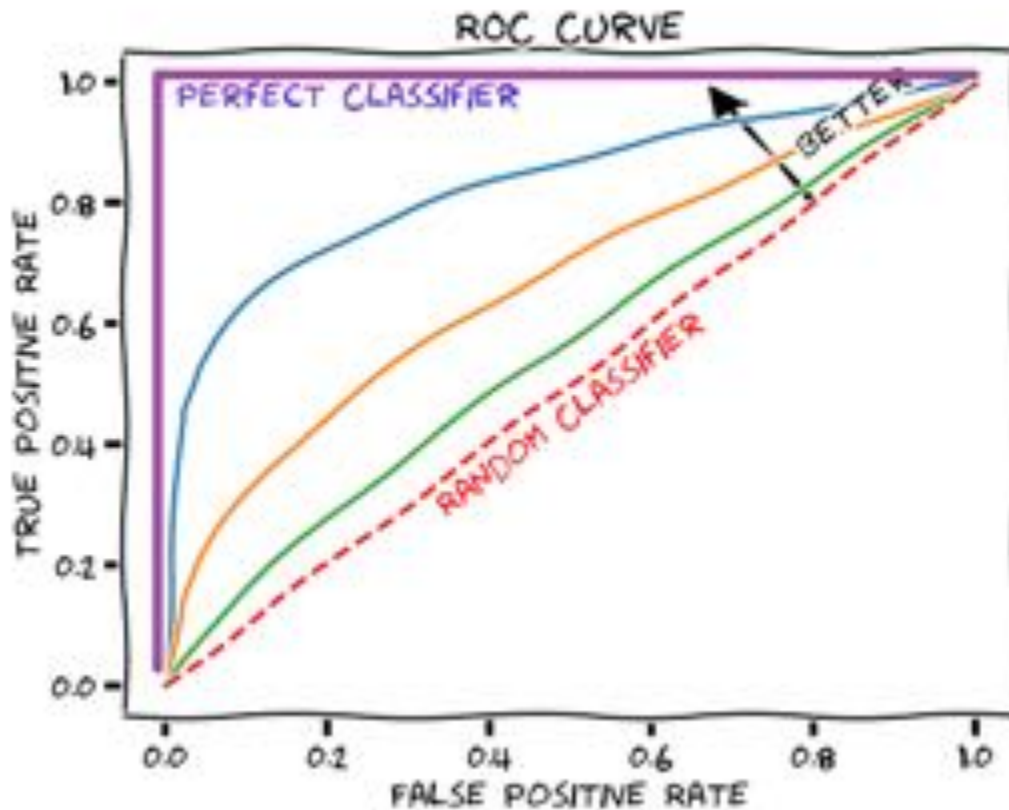
Binary classifications. The ROC curve is used to plot the rate of true positive occurrences with respect to the rate of false positive occurrences at various threshold values. The rate of the true positive occurrence is defined as the ratio of correct positive occurrences to the total number of occurrences. The rate of false positive occurrences is defined as the ratio of negative occurrences that are incorrectly predicted or labelled as positive occurrences to the total number of occurrences. The area below the ROC curve tells how much the model is able to distinguish between classes. The ROC curve represents probabilities and AUC represents the degree of separability (Narkhede, 2021). For better accuracy results, we need to maximize the area so that the model is able to distinguish classes accurately.

According to Draelos (2020b), The best model will have the AUROC value 1.0 and the worst model will have the value 0.5. As shown in Figure 47, the AUROC values and their meanings are listed below:

- The area under red dashed lines has AUROC value 0.5 which represents poor performance.
- The value < 0.7 is considered as average.
- $0.7 < \text{value} < 0.80$ is considered good.
- The value > 0.80 is considered excellent.
- The area of the envelope outlined by the purple line represents the AUROC value of 1.0 which is considered the best perfect model (Draelos, 2020b).

Figure 47

AUROC curve graph with respect to FPR and TPR



Note. Figure shows the AUCROC curve with respect to FPR and TPR. Adopted from *Measuring Performance: AUC (AUROC)*, by Draelos, V. A. P. B. R., 2020. (<https://glassboxmedicine.com/2019/02/23/measuring-performance-auc-auroc/>). Copyright by Glass Box 2020.

As shown in Table 6, for FCN models we are using metrics like Accuracy, Precision, F1 score, Intersection Over Union (IOU), Recall, and ROC curves to compare the training performance of the models. For the U-Net model we use metrics like Accuracy, DSC, and IntersectionOver Union (IOU) to evaluate the model performance results. For the LinkNet model we are using metrics like Accuracy, IntersectionOver Union (IOU), and DSC to compare

and analyze the model performance results. The detailed explanation of these results are explained in model validation and evaluation results.

Table 6

Evaluation metrics used on the models

Model	Evaluation Metrics used
FCN	1) Accuracy score 2) Intersection Over Union (IOU) 3) Precision score 4) F1 Score score 5) Recall score 6) ROC curve
U-NET	1) Accuracy score 2) DSC score 3) Intersection Over Union (IOU)
LinkNet	1) Accuracy score 2) DSC score 3) Intersection Over Union (IOU)

Note. Table shows information about evaluation metrics used for each model in this project

4.5 Model Validation and Evaluation Results

In this section we are validating and evaluating model results based on the evaluation metrics discussed in the above section. The models Fully Convolutional Neural Network (FCN), U-NET, and LinkNet are trained on the training satellite image datasets. After training the model our next step is to evaluate the model performance by using test satellite images.

4.5.1 Fully Convolutional Network (FCN) Results

These model results are evaluated by using the Accuracy, Recall, Intersection Over Union (IOU), Precision, F1 score metric, and ROC curve.

4.5.1.1 FCN evaluation results for all satellite images. As shown in Table 7, we are evaluating the Classes like Forest, Built Up, Farmland, and water based on the metrics. Here we are comparing metric results of the FCN-8 model with the eCog matric results. eCog results are from the software which segments the satellite images. We can see that the Average Accuracy, IOU, and Precision of all the classes for FCN-8 is better than the eCog results (Nayem et al., 2020).

Table 7

Evaluation metrics for multiple classes by using FCN-8 model

Class	Accuracy		IoU		Recall		Precision	
	FCN-8	eCog	FCN-8	eCog	FCN-8	eCog	FCN-8	eCog
Forest	0.82	0.80	0.73	0.77	0.30	0.63	0.81	0.72
Builtup	0.83	0.73	0.71	0.58	0.52	0.30	0.51	0.19
Farmland	0.73	0.63	0.60	0.47	0.33	0.23	0.59	0.32
Water	0.93	0.73	0.86	0.59	0.76	0.69	0.78	0.40
Average	0.85	0.74	0.76	0.60	0.43	0.46	0.75	0.41

Note. Table shows the metrics evaluation results of different classes in the satellite image.

Adopted from “*LULC Segmentation of RGB Satellite Image Using FCN-8,*” by Nayem et al., 2020, *CoRR*, *abs/2008.10736*. (<https://arxiv.org/abs/2008.10736>). Copyright 2020 by CORR.

4.5.1.2 FCN Evaluation Results for Region Based Satellite Images. We are evaluating the FCN models based on the skip connections to see the accuracy results. First of all we will test our satellite images with the basic FCN model. Secondly we will test on the FCN model without skip connection. Finally we are testing satellite images by using the FCN 8 which has skip connections. As we said earlier, while splitting the dataset for train, validate, and test datasets we

are considering region-based splitting. Here we are testing and validating our model on the satellite images of cities Prague and Massachusett.

4.5.1.2.1 Evaluation Results for Prague Test Dataset. We are evaluating the Prague test dataset based on the Precision, F1-Score, Recall, and ROC-AUC curves. For instance we are considering binary segmentation to detect road and background in the satellite image. We are applying a confusion matrix to the Prague dataset to get information on the number of pixel values we are dealing with. As shown in Table 8, the left and right side of the table represents the number of pixel values and the normalized values. The Confusion matrix is labelled as 2.3% of the Correct road class and 98% of the Correct background class in the Prague city test data. This Confusion matrix result shows a high imbalance dataset for Prague city (Muruganandham, 2016).

Table 8

Confusion Matrix for Prague dataset

		prediction outcome		prediction outcome	
		+	-	+	-
actual value	+	TP 32263	FN 105731	TP 0.234	FN 0.766
	-	FP 105773	TN 6006233	FP 0.017	TN 0.983

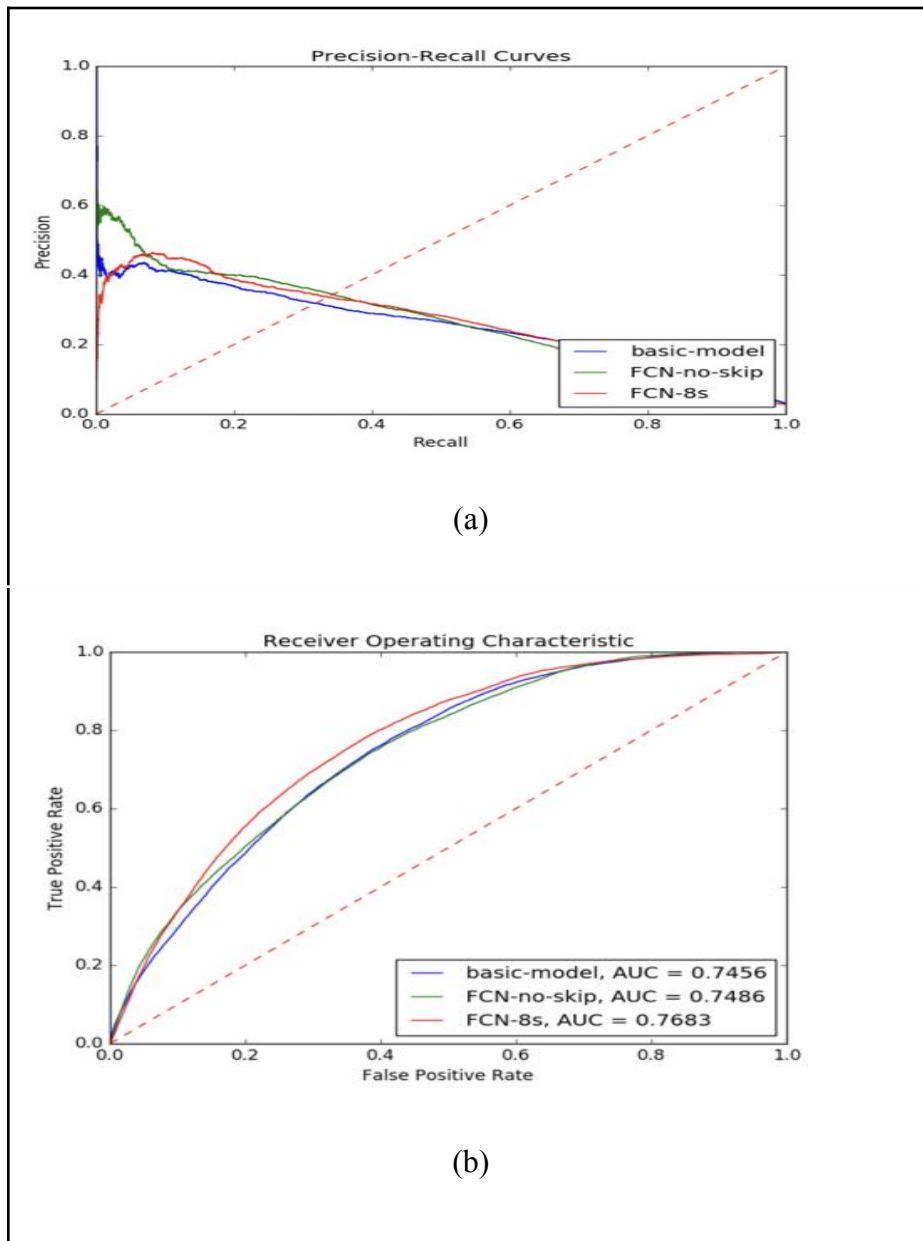
Note. Figure shows Confusion Matrix results for the Prague dataset. Adopted from “Semantic Segmentation of Satellite Images using Deep Learning,” by Muruganandham, S., 2016.

(<https://www.diva-portal.org/smash/get/diva2:1013270/FULLTEXT01.pdf>), Copyright by Diva Portal 2016.

As shown in Figure 48, The imbalance satellite image data of the Prague city's evaluation is done on the Precision-Recall curves and ROC curves. In these results we can see how the model evaluates the results of unknown datasets. As shown in Table 9, For this Prague dataset, the FCN basic model is working better than the FCN-no-skip model. FCN basic model metric F1 score 0.322 is higher than the FCN-no-skip model. The IOU of this Prague dataset is shown as 20% which means these low values of all the metrics indicate the model needs some more training so we need to improve our training pipeline (Muruganandham, 2016).

Figure 48

Prec-Rec and ROC curves for Prague test set



Note. Figure shows Prec-Rec and ROC curves for the Prague test set. *Adopted from* “Semantic Segmentation of Satellite Images using Deep Learning,” by Muruganandham, S., 2016.

(<https://www.diva-portal.org/smash/get/diva2:1013270/FULLTEXT01.pdf>), Copyright by Diva Portal 2016.

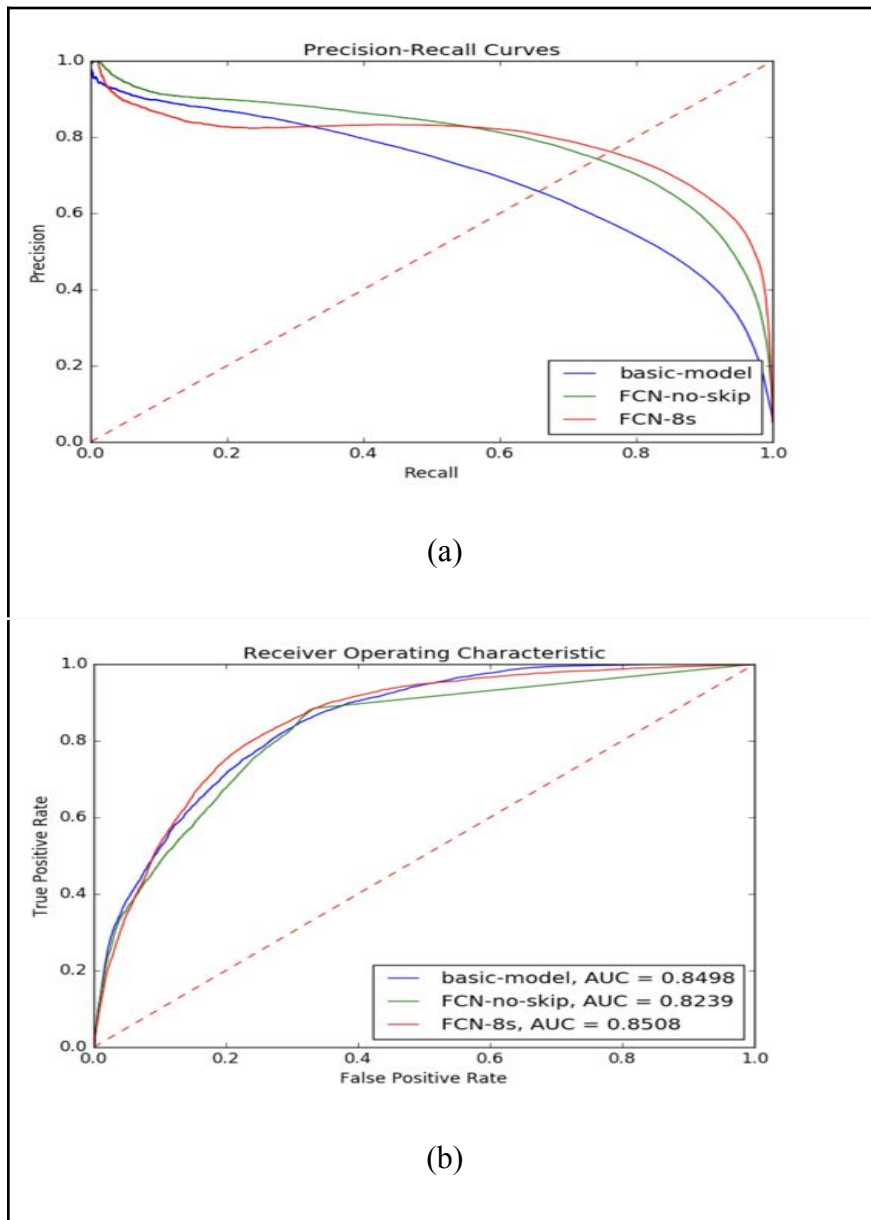
Table 9*Evolution metrics results for Prague test data*

<i>Model</i>	<i>ROC-AUC</i>	<i>BEP Threshold</i>	<i>Precision</i>	<i>Recall</i>	<i>F1-Score</i>
basic-model	0.7456	0.340	0.233	0.233	0.23
FCN-no-skip	0.7486	0.396	0.319	0.325	0.322
FCN-8s	0.7683	0.487	0.319	0.283	0.30

Note. Table shows Evolution metrics results for the Prague test data. *Adopted from* “Semantic Segmentation of Satellite Images using Deep Learning,” by Muruganandham, S., 2016.

(<https://www.diva-portal.org/smash/get/diva2:1013270/FULLTEXT01.pdf>), Copyright by Diva Portal 2016.

4.5.1.2.1 Evaluation Results for Massachusetts Test Dataset. We are evaluating the Massachusetts test dataset based on the Precision, F1-Score, Recall, and ROC-AUC curves. As shown in Figure 49, the three models which are basic model, FCN-no-skip, and FCN8 (with skip) are evaluated on the Prec-Rec curves and the ROC curves. We can see the Prec_Rec curves are showing high values which means the Massachusetts test data is closer to the Massachusetts train data so here we need to take into consideration that we need to evaluate or validate the model on completely different datasets. As shown in the table 10, the FCN 8 (with skip) model is showing the best performance results of F1-Score, Precision, and Recall 0.762 and the ROC- AUC curve result of 0.85 which is considered to be the best performance model in this case (Muruganandham, 2016).

Figure 49*Prec-Rec and ROC curves for Massachusetts test set*

Note. Figure shows Prec-Rec and ROC curves for the Massachusetts test set. Adopted from “Semantic Segmentation of Satellite Images using Deep Learning,” by Muruganandham, S., 2016. (<https://www.diva-portal.org/smash/get/diva2:1013270/FULLTEXT01.pdf>), Copyright by Diva Portal 2016.

Table 10*Evolution metrics results for Massachusetts test data*

<i>Model</i>	<i>ROC-AUC</i>	<i>BEP Threshold</i>	<i>Precision</i>	<i>Recall</i>	<i>F1-Score</i>
basic-model	0.849	0.274	0.657	0.657	0.657
FCN-no-skip	0.82	0.415	0.742	0.742	0.742
FCN-8s	0.85	0.523	0.762	0.762	0.762

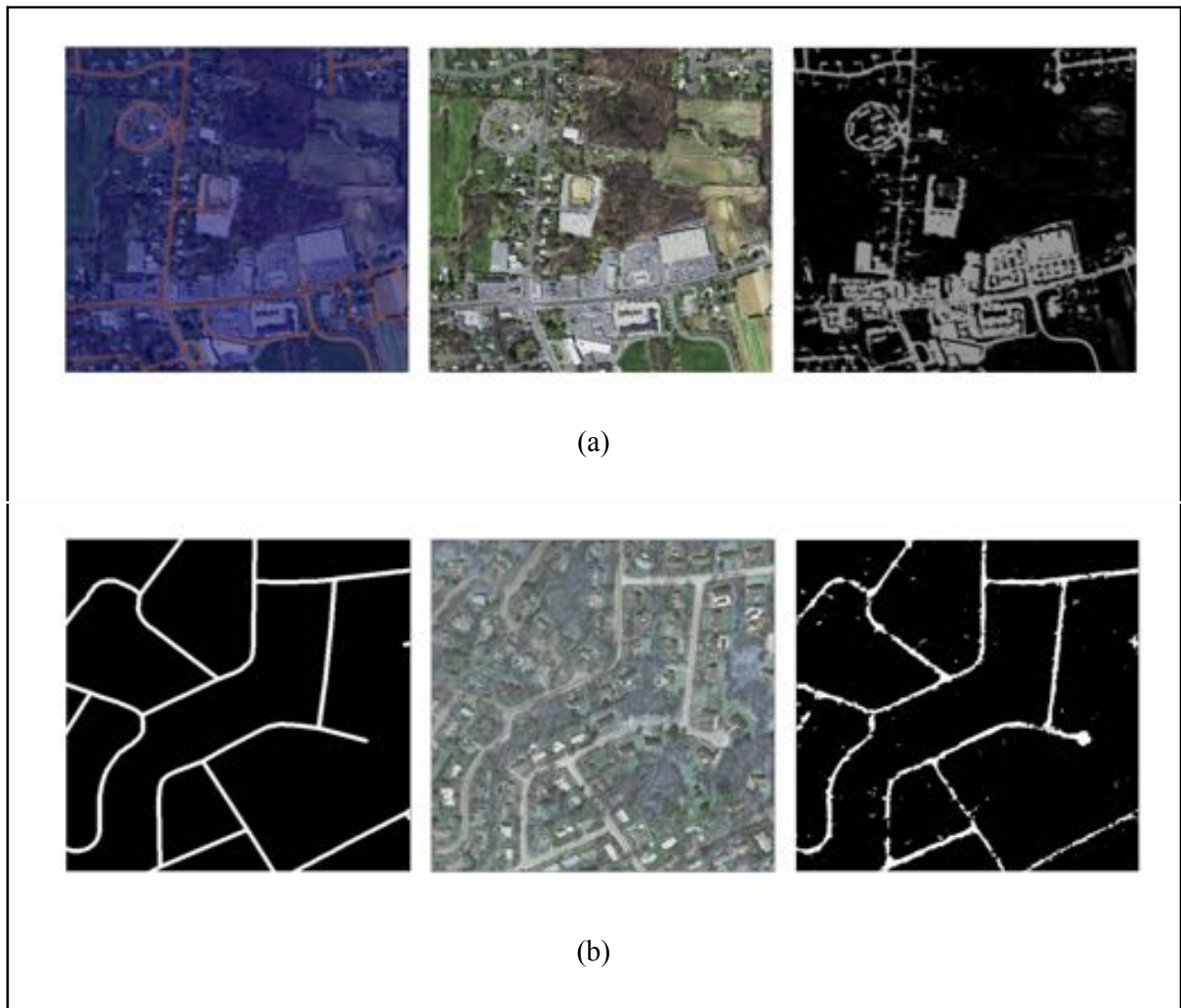
Note. Table shows Evolution metrics results for Massachusetts test data. *Adopted from* “Semantic Segmentation of Satellite Images using Deep Learning,” by Muruganandham, S., 2016.

(<https://www.diva-portal.org/smash/get/diva2:1013270/FULLTEXT01.pdf>), Copyright by Diva Portal 2016.

We are again training the models based on different datasets to increase the accuracy. As shown in Figure 50, the model learns the satellite images for the Massachusetts dataset. As shown in Figure 51, validation is performed on the learned model and presents the sample segmented images for the Massachusetts dataset. As shown in Figure 52, the validation is performed on the learned model and presents the sample segmented images for the Pardue dataset. Our models are now clearly able to segment the roads from the background (Muruganandham, 2016).

Figure 50

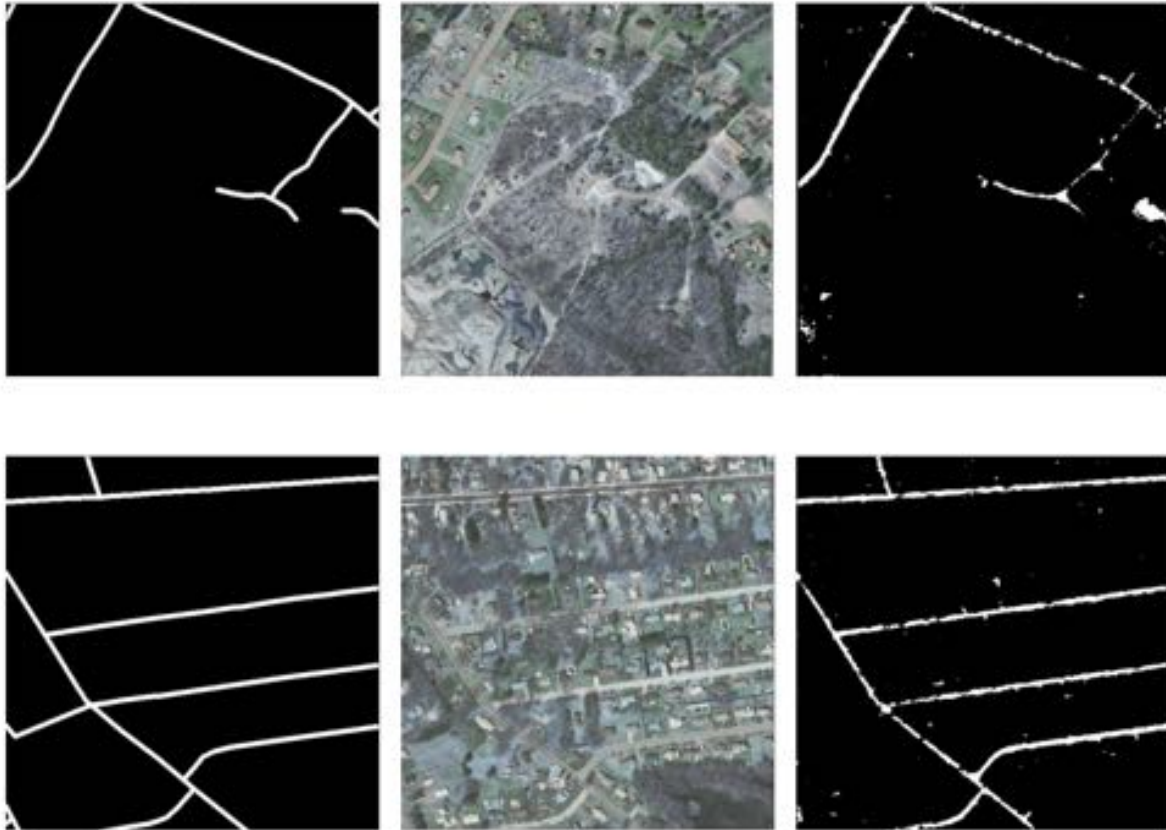
Model learning on the Massachusetts validation set



Note. Figure shows Model learning on the Massachusetts validation set. Adopted from “Semantic Segmentation of Satellite Images using Deep Learning,” by Muruganandham, S., 2016. (<https://www.diva-portal.org/smash/get/diva2:1013270/FULLTEXT01.pdf>), Copyright by Diva Portal 2016.

Figure 51

Sample results of Massachusetts validation sets after the model learning



Note. Figure shows Sample results of Massachusetts validation sets after the model learning.

Adopted from “Semantic Segmentation of Satellite Images using Deep Learning,” by

Muruganandham, S., 2016.

(<https://www.diva-portal.org/smash/get/diva2:1013270/FULLTEXT01.pdf>), Copyright by Diva

Portal 2016.

Figure 52

Sample results of Prague validation sets after the model learning



Note. Figure shows Sample results of Prague validation sets after the model learning. Adopted from “Semantic Segmentation of Satellite Images using Deep Learning,” by Muruganandham, S., 2016. (<https://www.diva-portal.org/smash/get/diva2:1013270/FULLTEXT01.pdf>), Copyright by Diva Portal 2016.

4.5.2 U-NET and LinkNet Model Results

Since U-NET and LinkNet models are almost the same both have multiple skip connections and multiple deconvolutions, we are evaluating these two models on the metrics like Accuracy and the Sorensen-Dice coefficient (DSC). The author Ivanovsky (2019) mentions in his paper, the accuracy metric in terms of image segmentation means the fraction of accurately predicted pixels over the total number of image pixels. The DSC is used to calculate the segmentation quality.

As shown in Table 11, the Accuracy score of the U-NET model is comparatively higher than the LinkNet model. As shown in Table 12, The quality of segmentation Sorensen-Dice coefficient (DSC) of U-NET is higher than the LinkNet model. As shown in Figure 53, the graph of segmentation quality of U-NET and LinkNet shows the U-NET model is segmenting the labels better than the LinkNet. From these results we can say that the U-NET model is performing better than the LinkNet model. However, the LinkNet training period is taking 1.5 hours which is less than the learning period of the U-NET which is one hour. Both models are encoder-decoder network based architecture with the decoder extracts features from the encoder which allow the model to learn more useful information from the input satellite image to give a well accurate output image (Ivanovsky, 2019).

Table 11

Accuracy result of U-NET and Link-Net models

Model	Accuracy (A)
U-Net	96,31%
LinkNet	95,85%

Note. Table shows the U-NET and LinkNet model Accuracy results. Adopted from “Building Detection on Aerial Images Using U-NET Neural Networks,” by Ivanovsky et al., 2019 24th Conference of Open Innovations Association (FRUCT). (

<https://doi.org/10.23919/fruct.2019.8711930>). Copyright by FRUCT 2019.

Table 12

Sorensen-Dice coefficient (DSC) result of U-NET and Link-Net models

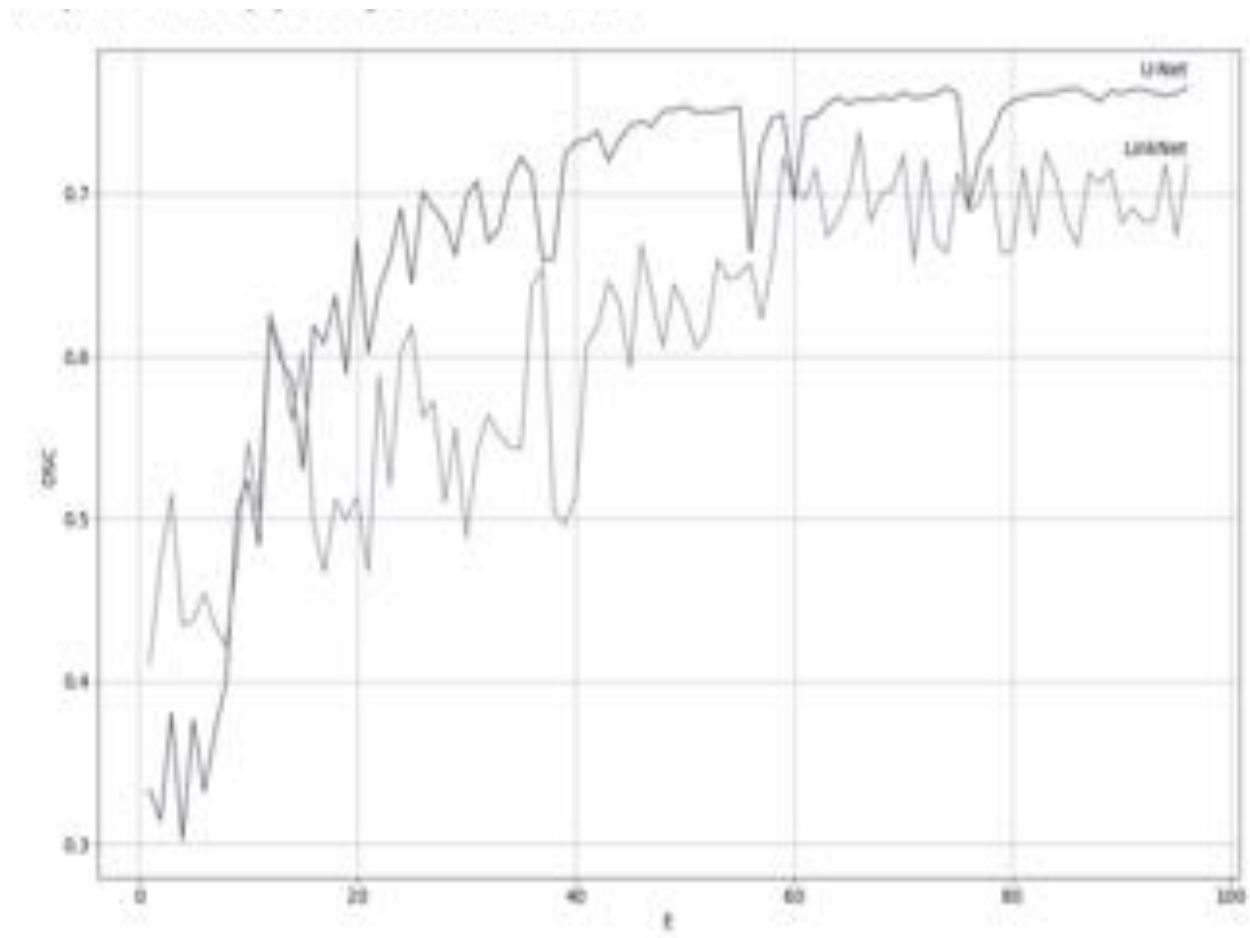
Model	Sorensen-Dice coefficient (DSC)
U-Net	0,77
LinkNet	0,72

Note. Table shows the U-NET and LinkNet model DSC results. Adopted from “Building Detection on Aerial Images Using U-NET Neural Networks,” by Ivanovsky et al., 2019 24th Conference of Open Innovations Association (FRUCT).

(<https://doi.org/10.23919/fruct.2019.8711930>). Copyright by FRUCT 2019.

Figure 53

Segmentation quality graph of U-NET and LinkNet



Note. Graph shows the U-NET and LinkNet model DSC results. Adopted from “Building Detection on Aerial Images Using U-NET Neural Networks,” by Ivanovsky et al., *2019 24th Conference of Open Innovations Association (FRUCT)*.

(<https://doi.org/10.23919/fruct.2019.8711930>). Copyright by FRUCT 2019.

4.5.2 U-NET and FCN Model Results

In this section, we are comparing results of FCN and U-NET models by using accuracy and validation curves. We are also validating the models based on different sized images since both models can handle arbitrary image size. As shown in Figure 54, the models are evaluated

on the size 256 x 256. As shown in Figure 55, the models are evaluated on the size 512 x 512. As shown in Table 8, the hyperparameters like learning rate, batch size and epochs for the models FCN and U-NET are shown to analyze different sized images. For training these both models we are using 1200 satellite image samples and for validating we are using 300 samples (Ozturk et al., 2020b).

As shown in Table 13, we are using hyperparameters like weight update coefficient as 0.0001, Batch size as 16, and epochs of size 100 for Adam optimizer. For SGD optimizer, we have a weight update coefficient of 0.01, batch size of four, and epochs of 80 (Ozturk et al., 2020b).

Table 13

Hyperparameters used for FCN and U-NET

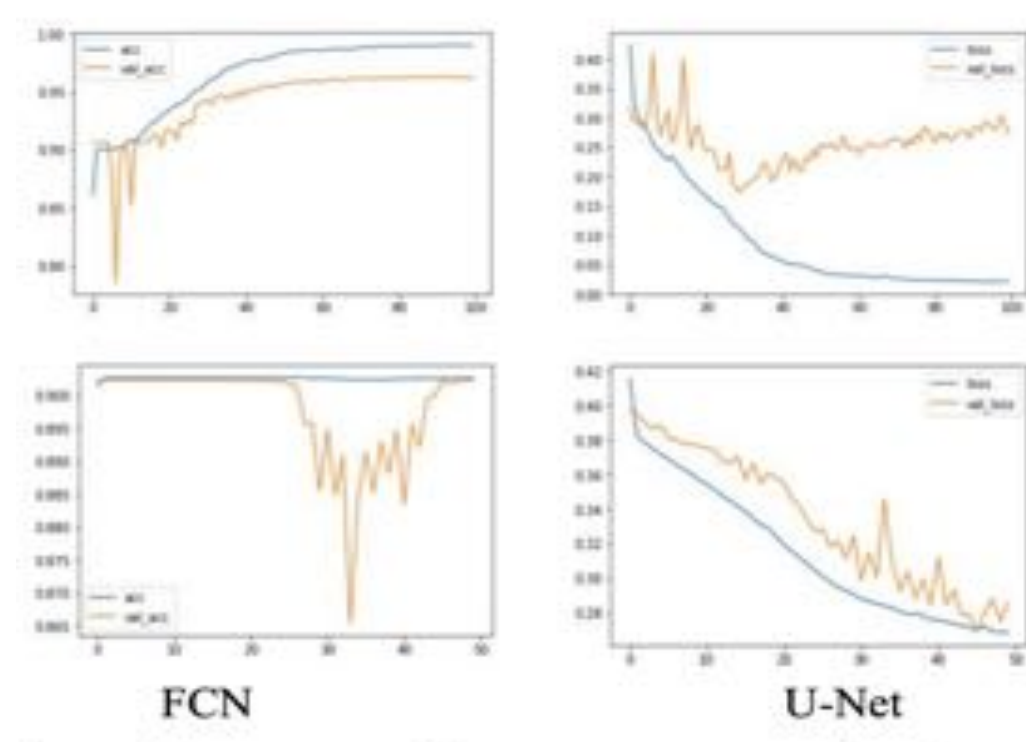
	U-Net (256)	FCN (256)	U-Net (512)	FCN (512)
Optimizer	Adam	Adam	Adam	SGD
Learning Rate	0.0001	0.0001	0.0001	0.01
Batch Size	16	16	16	4
Epochs	100	100	100	80

Note. Table shows Hyperparameters used for FCN and U-NET models. Adopted from “Comparison of Fully Convolutional Networks (FCN) and U-Net for Road Segmentation from High Resolution Imageries,” by Ozturk et al., 2020. *International Journal of Environment and Geoinformatics*, 7(3), 272–279. (<https://doi.org/10.30897/ijegeo.737993>)

As shown in Figure 56, the image size 256 x 256 for the model U-NET accuracy is 96.33% and the FCN accuracy score is 90.23 %. The U-NET model accuracy is 6.1 % higher than the FCN model with the image size 256 x 256. The U-NET accuracy score for image size 512 x 512 is calculated as 96.18% and for FCN it is 97.69%. For image size 512 x 512 FCN is performing better than the U-NET. According to loss results, U-NET is performing better than the FCN model for size 256 x 256 (Ozturk et al., 2020b).

Figure 54

Accuracy and loss curves of FCN and U-NET models for size 256 X 266

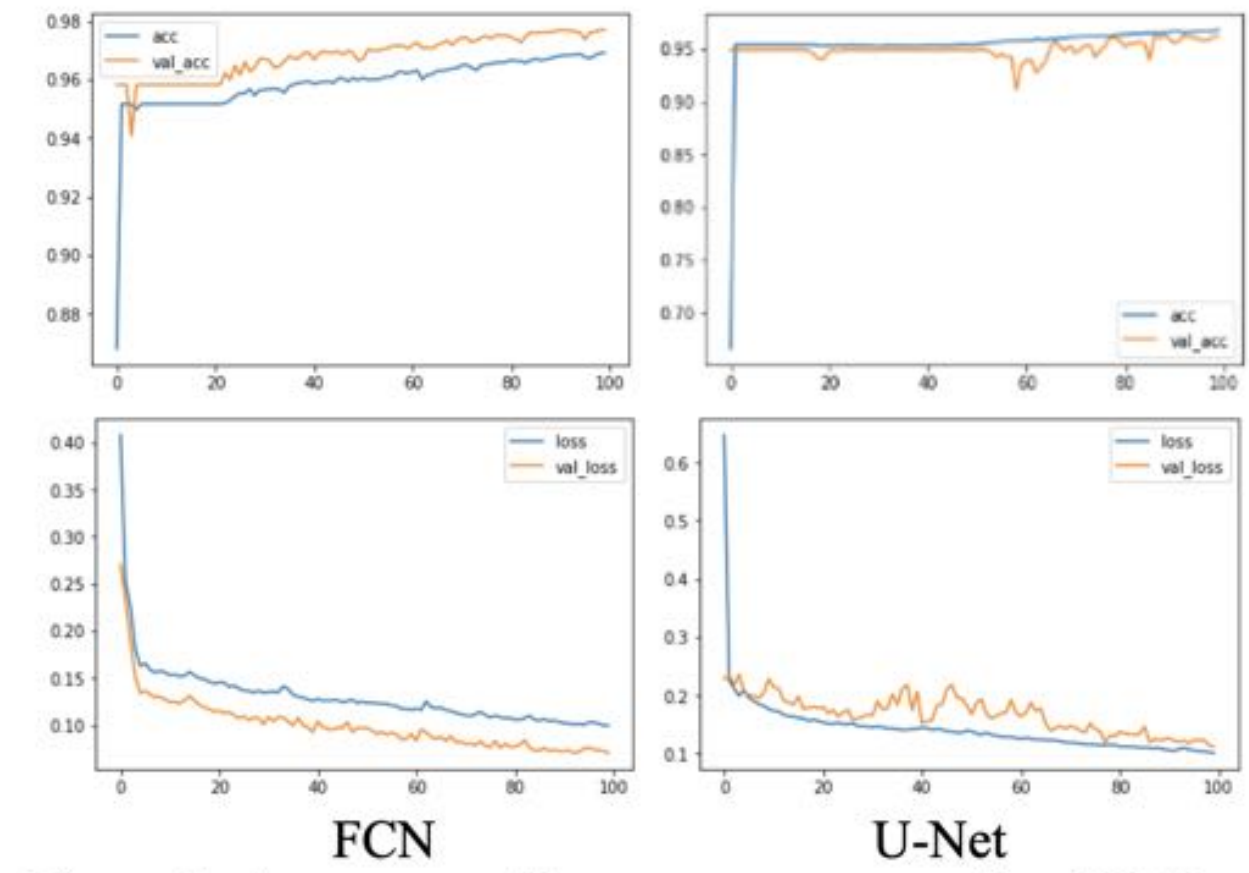


Note. Figure shows Accuracy and loss curves of FCN and U-NET models for size 256 X 266.

Adopted from “Comparison of Fully Convolutional Networks (FCN) and U-Net for Road Segmentation from High Resolution Imageries,” by Ozturk et al., 2020. *International Journal of Environment and Geoinformatics*, 7(3), 272–279. (<https://doi.org/10.30897/ijgeo.737993>)

Figure 55

Accuracy and loss curves of FCN and U-NET models for size 512 X 512

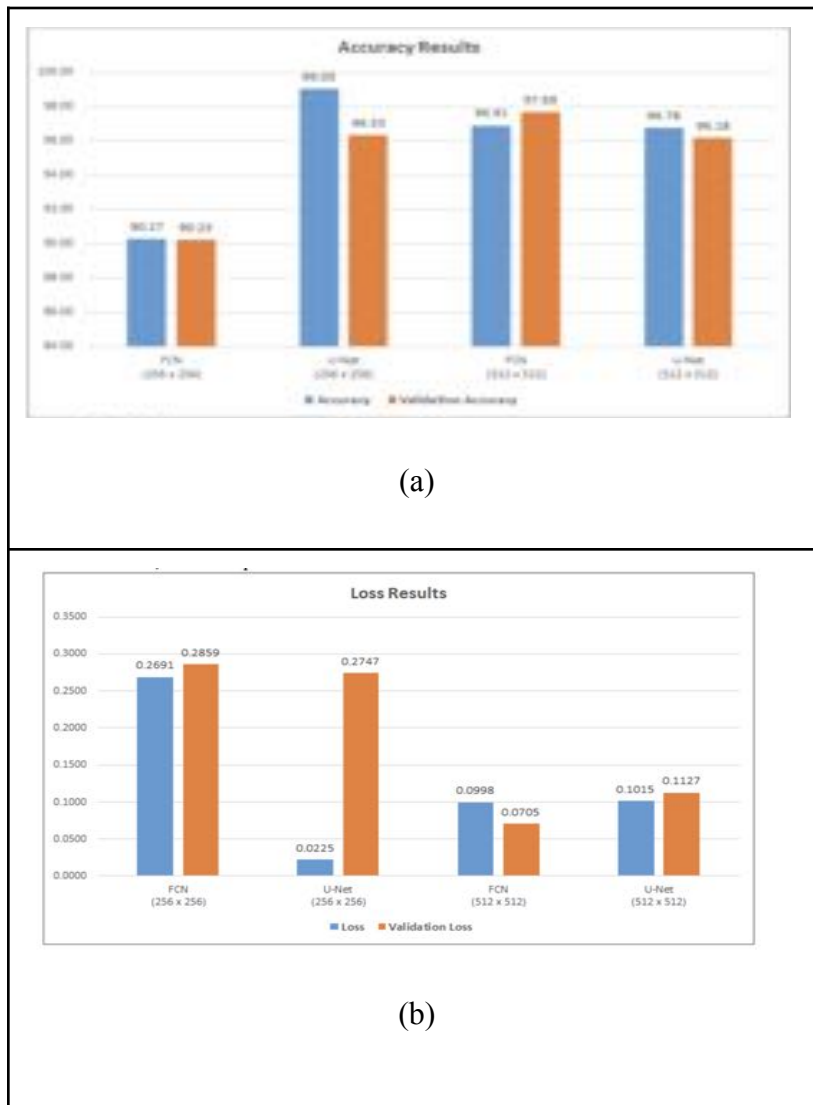


Note. Figure shows Accuracy and loss curves of FCN and U-NET models for size 512 x 512.

Adopted from “Comparison of Fully Convolutional Networks (FCN) and U-Net for Road Segmentation from High Resolution Imageries,” by Ozturk et al., 2020. *International Journal of Environment and Geoinformatics*, 7(3), 272–279. (<https://doi.org/10.30897/ijegeo.737993>)

Figure 56

Accuracy and loss plots of FCN / U-NET models on different size images

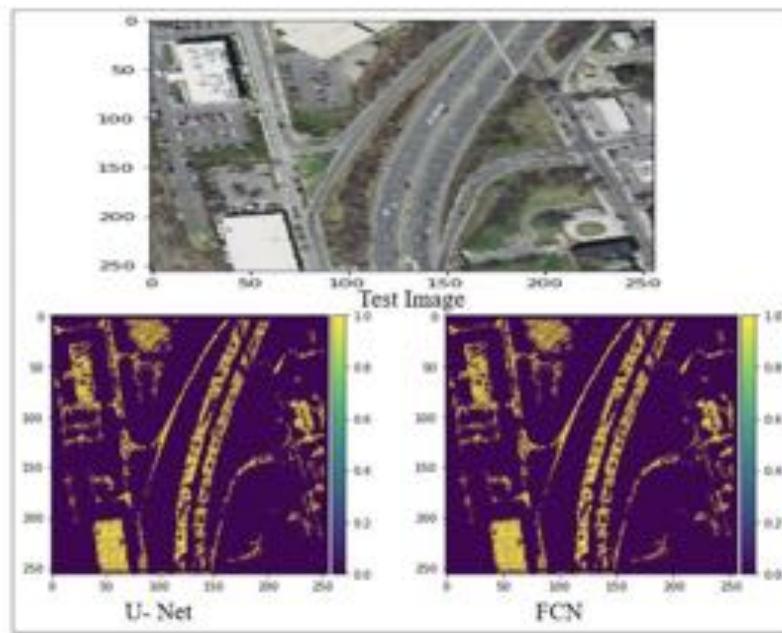


Note. Graph shows Accuracy and loss results of FCN and U-NET models on different size images. Adopted from “Comparison of Fully Convolutional Networks (FCN) and U-Net for Road Segmentation from High Resolution Imageries,” by Ozturk et al., 2020. *International Journal of Environment and Geoinformatics*, 7(3), 272–279. (<https://doi.org/10.30897/ijgeo.737993>)

As shown in Figure 57, evaluation performance of both FCN and U-NET model sample segmented image size 256 x 256 are represented, which are obtained by both models. In these segmented images, the yellow color represents the road class and the purple represents the background of the satellite image. As shown in Figure 58, the performance of both FCN and U-NET model sample segmented image size 512 x 512 are represented, which are obtained by both models. In these segmented images, the yellow color represents the road class and the purple represents the background of the satellite image (Ozturk et al., 2020b).

Figure 57

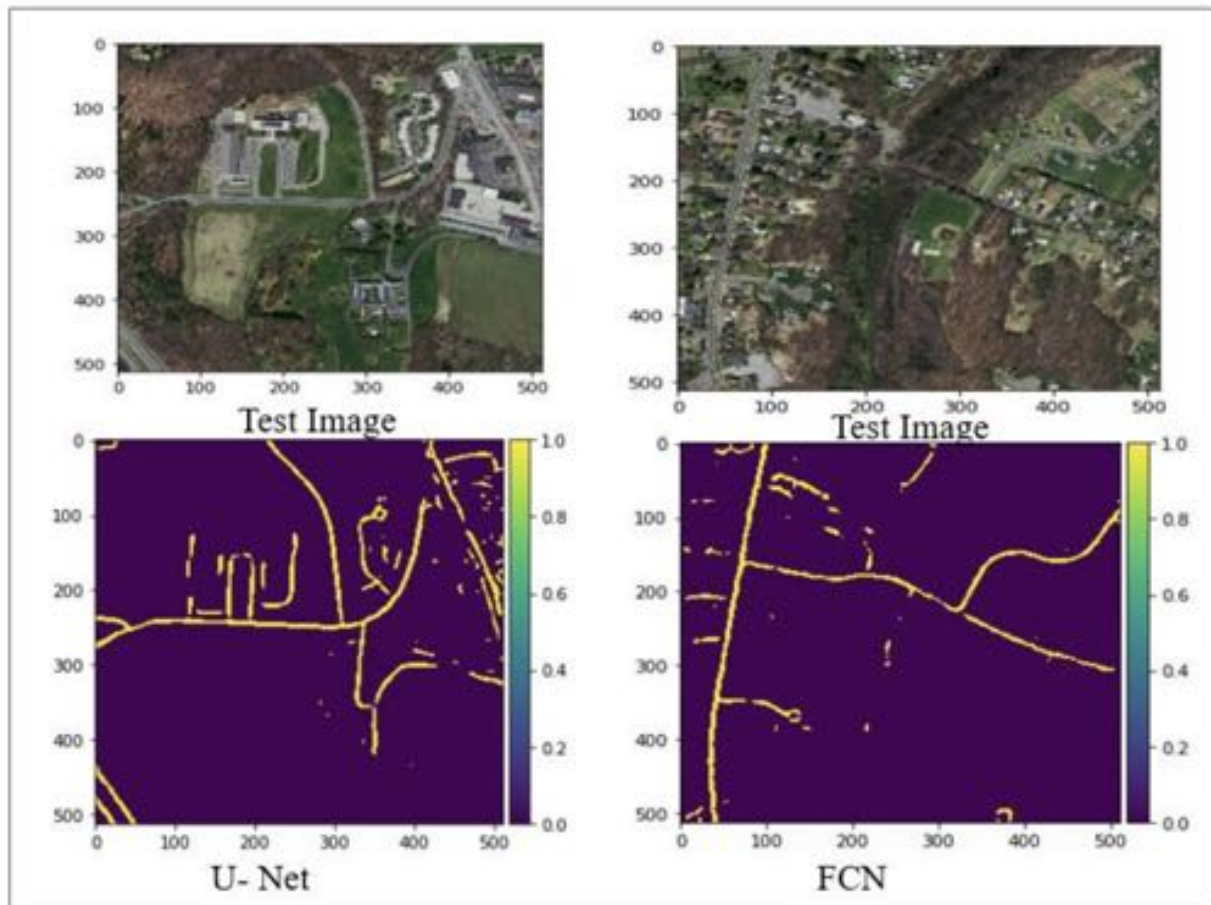
Segmented image results for size 256 x 256



Note. Figure shows Segmented image results for size 256 x 256. Adopted from “Comparison of Fully Convolutional Networks (FCN) and U-Net for Road Segmentation from High Resolution Imageries,” by Ozturk et al., 2020. *International Journal of Environment and Geoinformatics*, 7(3), 272–279. (<https://doi.org/10.30897/ijegeo.737993>)

Figure 58

Segmented image results for size 512 x 512



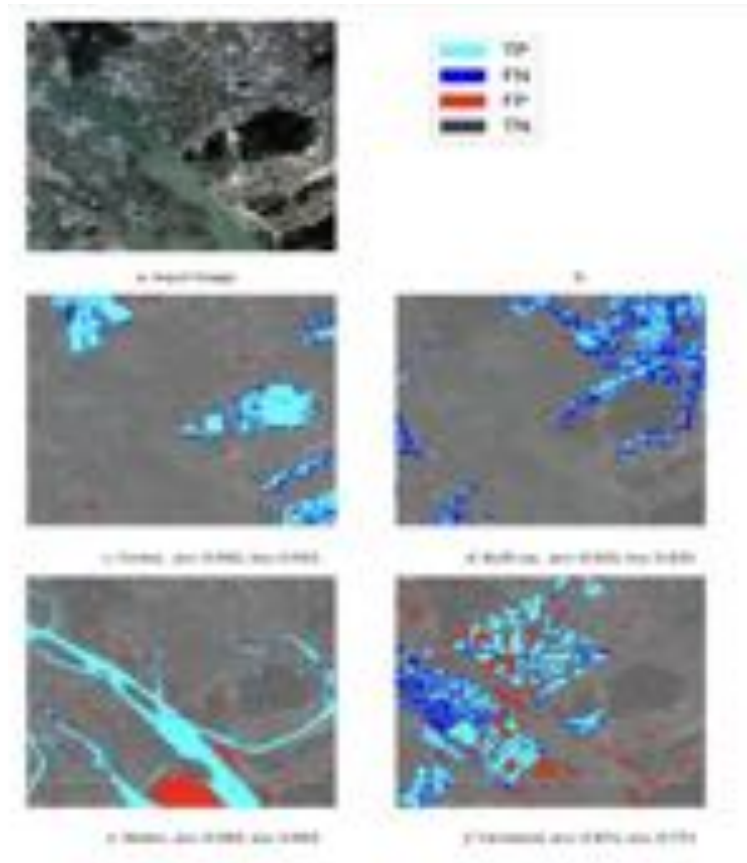
Note. Figure shows Segmented image results for size 512 x 512. Adopted from “Comparison of Fully Convolutional Networks (FCN) and U-Net for Road Segmentation from High Resolution Imageries,” by Ozturk et al., 2020. *International Journal of Environment and Geoinformatics*, 7(3), 272–279. (<https://doi.org/10.30897/ijegeo.737993>)

As shown in Figure 59, for the FCN 8 model the input satellite image is masked based on the evaluation metrics such as True Positive (TP) which is represented in Cyan, True Negative (TN) represented in black, False Positive (TN) is represented in red, and finally False Negative (FN) is represented in Blue. The model is evaluating correctly most of the times the True

Negative, which is black color segmentation, is higher in that satellite image. The model evaluated forest with an accuracy score of 96% and for water with the accuracy of 94% (Nayem et al., 2020).

Figure 59

Evaluation results showed on the input result



Note. Figure shows Evaluation results shown on the input result. Adopted from “*LULC Segmentation of RGB Satellite Image Using FCN-8*,” by Nayem et al., 2020, *CoRR*, *abs/2008.10736*. (<https://arxiv.org/abs/2008.10736>). Copyright 2020 by CORR.

As we said earlier in this project we are segmenting both binary and multiclass satellite images. Here we have presented some sample results from the models for multiclass segmentation. The sampled results are evaluated on the land cover dataset. As shown in table 14,

the classes of satellite images are masked with the multiple colors. As shown in Figure 60, the sampled results of the model used for semantic segmentation are labelled or segmented correctly with the different colors with respect to the above table for each class in the image (Baratam, 2021).

Table 14

Multiclass segmentation table

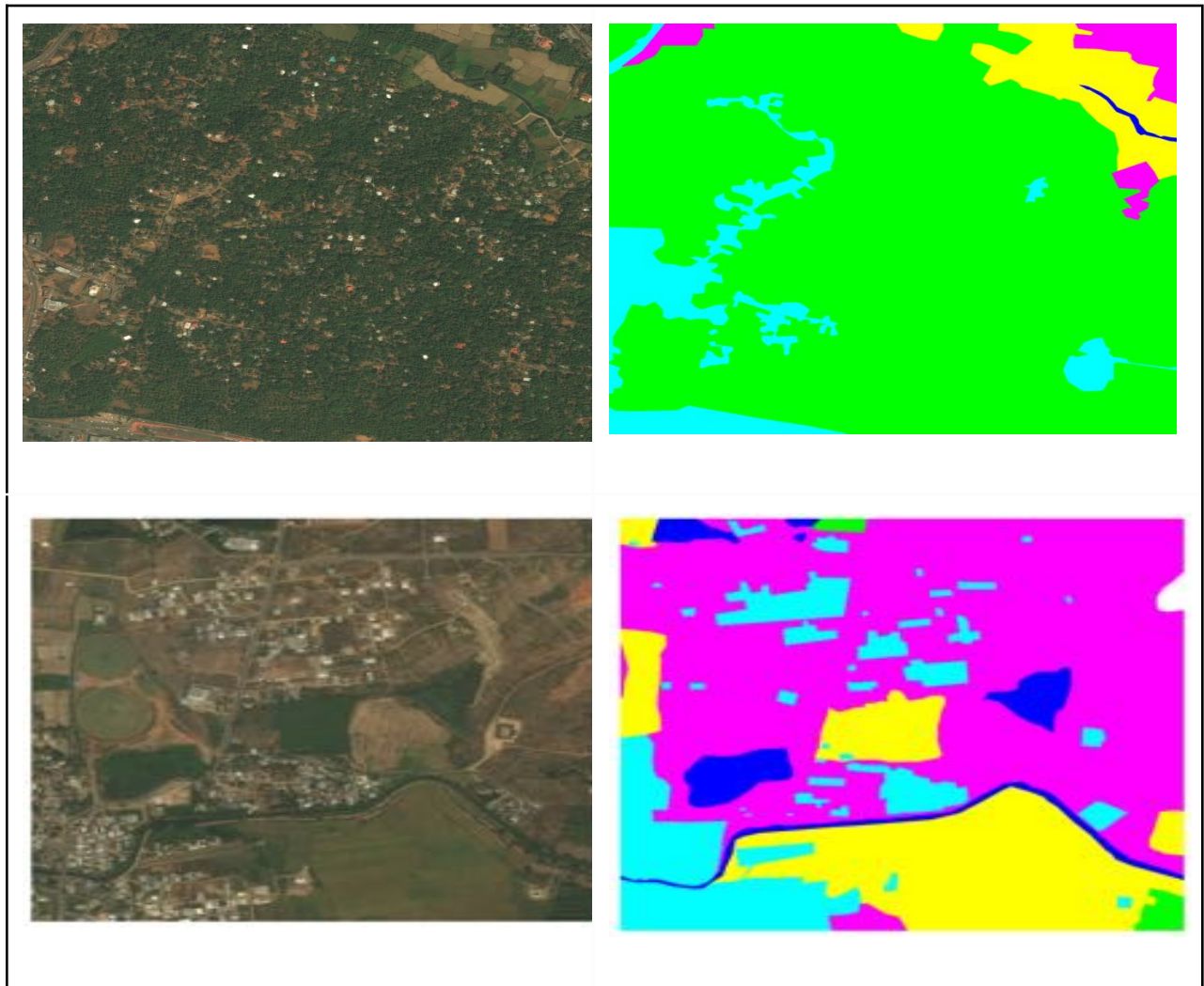
Class	Mask Colour	Pixel Count (in millions)	Proportion
Urban Land	Cyan	461.19	11.27%
Agricultural Land	Yellow	2379.65	58.14%
Range Land	Magenta	343.12	8.38%
Forest	Green	444.92	10.87%
Water	Blue	138.39	3.38%
Barren Land	White	323.39	7.90%
Unknown	Black	2.35	0.06%

Note. Table shows Multiclass segmentation with labelled colors. Adopted from *Land Cover*

Classification with U-Net - Srimannarayana Baratam, by Baratam, S., 2021.

(<https://baratam-tarunkumar.medium.com/land-cover-classification-with-u-net-aa618ea64a1b>).

Copyright by Medium 2021

Figure 60*Multiclass segmentation results*

Note. Figure shows Multiclass segmentation results of satellite images. Adopted from *Land Cover Classification with U-Net - Srimannarayana Baratam*, by Baratam, S., 2021.

(<https://baratam-tarunkumar.medium.com/land-cover-classification-with-u-net-aa618ea64a1b>).

Copyright by Medium 2021

References

- Akar, Z., & Güngör, O. (2012). Classification of multispectral images using Random Forest algorithm. *Journal of Geodesy and Geoinformation*, 1(2), 105–112.
<https://doi.org/10.9733/jgg.241212.1>
- Artan, Y. (2011). Interactive Image Segmentation Using Machine Learning Techniques. *2011 Canadian Conference on Computer and Robot Vision*. Published.
<https://doi.org/10.1109/crv.2011.42>
- Ashwath, B. (2020, November 11). *DeepGlobe Land Cover Classification Dataset*. Kaggle.
<https://www.kaggle.com/balraj98/deepglobe-land-cover-classification-dataset/metadata>
- Badrinarayanan, V., Kendall, A., & Cipolla, R. (2017). SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(12), 2481–2495.
<https://doi.org/10.1109/tpami.2016.2644615>
- Bagwell, R. (n.d.). *GIBS Available Imagery Products - Global Imagery Browse Services (GIBS) - Earthdata Wiki*. EarthData.
<https://wiki.earthdata.nasa.gov/display/GIBS/GIBS+Available+Imagery+Products>
- Baratam, S. (2021, July 17). *Land Cover Classification with U-Net - Srimannarayana Baratam*. Medium.
<https://baratam-tarunkumar.medium.com/land-cover-classification-with-u-net-aa618ea64a1b>
- Bernard, K., Tarabalka, Y., Angulo, J., Chanussot, J., & Benediktsson, J. A. (2012). Spectral–Spatial Classification of Hyperspectral Data Based on a Stochastic Minimum Spanning Forest Approach. *IEEE Transactions on Image Processing*, 21(4), 2008–2021.

<https://doi.org/10.1109/tip.2011.2175741>

Bhandari, A. (2021, July 23). *Confusion Matrix for Machine Learning*. Analytics Vidhya.

<https://www.analyticsvidhya.com/blog/2020/04/confusion-matrix-machine-learning/>

Boushaala, A.A. (2014). An Approach for Project Scheduling Using PERT/CPM and Petri Nets

(PNs) Tools. <https://doi.org/10.6084/M9.FIGSHARE.1093976.V1>

Chaurasia, A., & Culurciello, E. (2017). LinkNet: Exploiting encoder representations for

efficient semantic segmentation. *2017 IEEE Visual Communications and Image*

Processing (VCIP). Published. <https://doi.org/10.1109/vcip.2017.8305148>

Chhor, G., & Aramburu, C.B. (2017). Satellite Image Segmentation for Building Detection using U-net.

<https://www.semanticscholar.org/paper/Satellite-Image-Segmentation-for-Building-Detection-Chhor-Aramburu/abb13964a435e1ac0c77b7dd68095e9da81b90aa>

CRISP-DM. (2021, August 24). Data Science Process Alliance. Retrieved October 10, 2021,

from <https://www.datascience-pm.com/crisp-dm-2/>

Crisp-dm-2: *What is CRISP-DM?*. (n.d.). datascience-pm.

<https://www.datascience-pm.com/crisp-dm-2/>

Data labeling. (n.d.-b). GroundWork. <https://groundwork.azavea.com/data-labeling/>

Data management plans. (n.d.). Stanford Libraries. Retrieved October 10, 2021, from

<https://library.stanford.edu/research/data-management-services/data-management-plans>

Data management services: data management plans. (2021). Stanford Libraries.

<https://library.stanford.edu/research/data-management-services/data-management-plans>

Draelos, V. A. P. B. R. (2020, February 2). *Measuring Performance: AUC (AUROC)*. Glass Box.

<https://glassboxmedicine.com/2019/02/23/measuring-performance-auc-auroc/>

Demir, I., Koperski, K., Lindenbaum, D., Pang, G., Huang, J., Basu, S., Hughes, F., Tuia, D., &

Raskar, R. (2018). DeepGlobe 2018: A Challenge to Parse the Earth through Satellite

Images. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*

Workshops (CVPRW). Published. <https://doi.org/10.1109/cvprw.2018.00031>

Escobar, F. (2020). Satellite images of water bodies (version 2) [Data set].

<https://www.kaggle.com/franciscoescobar/satellite-images-of-water-bodies>

Gantt Charts: Definitions, Features, & Uses. (n.d.). TeamGantt. Retrieved October 11, 2021,

from <https://www.teamgantt.com/what-is-a-gantt-chart>

Grootendorst, M. (2021, March 30). *9 Distance Measures in Data Science | Towards Data*

Science. Medium.

<https://towardsdatascience.com/9-distance-measures-in-data-science-918109d069fa>

Hansen, B. (2018, September 27). *What is resource management: why is it important?*. Wrike:

<https://www.wrike.com/blog/what-is-resource-management/>

He, K., Gkioxari, G., Dollár, P., & Girshick, R. (2017). Mask R-CNN, *2017 IEEE International*

Conference on Computer Vision (ICCV), 2017, pp. 2980-2988, <https://doi.org/10.1109/ICCV.2017.322>.

Hultberg, J. (2018). *Dehazing of satellite Images* [Master's thesis in Electrical Engineering,

Linköping University].

<https://liu.diva-portal.org/smash/get/diva2:1215181/FULLTEXT01.pdf>

Humans In The Loop (2020). *Semantic segmentation of aerial imagery* (version 1) [Data set].

<https://www.kaggle.com/humansintheloop/semantic-segmentation-of-aerial-imagery>

Illarionova, S., Nesteruk, S., Shadrin, D., Ignatiev, V., Pukalchik, M., & Oseledets, I.

(2021). MixChannel: Advanced Augmentation for Multispectral Satellite Images.

Remote Sensing, 13(11), 2181. <https://doi.org/10.3390/rs13112181>

Ivanovsky, L., Khrushchev, V., Pavlov, V., & Ostrovskaya, A. (2019). Building detection on

aerial images using u-net neural networks, *2019 24th Conference of Open*

Innovations Association (FRUCT), 116-122.

<https://doi.org/10.23919/FRUCT.2019.8711930>.

Jiang, R. X. (2018, June 13). *Neural network for satellite image segmentation - Towards Data Science*. Medium.

<https://towardsdatascience.com/dstl-satellite-imagery-contest-on-kaggle-2f3ef7b8ac40>

Kakarla, S. (2020, September 23). *Exploratory Data Analysis (EDA) on Satellite Imagery Using*

EarthPy. Medium. <https://towardsdatascience.com/exploratory-data-analysis-eda>

[-on-satellite-imagery-using-earthpy-c0e186fe4293](https://towardsdatascience.com/exploratory-data-analysis-eda-on-satellite-imagery-using-earthpy-c0e186fe4293)

Landsat 8. (n.d.). USGS. <https://www.usgs.gov/core-science-systems/nli/landsat/>

[landsat-8?qt-science_support_page_related_con=0#qt-science_support_page_related_con](https://www.usgs.gov/core-science-systems/nli/landsat/landsat-8?qt-science_support_page_related_con=0#qt-science_support_page_related_con)

Liu, X., Deng, Z. & Yang, Y. (2019). Recent progress in semantic image segmentation.

2019 Artif Intell Rev 52, 1089–1106. <https://doi.org/10.1007/s10462-018-9641-3>

Long, A. (2021, October 11). *Understanding Data Science Classification Metrics in Scikit-Learn in Python*. Medium.

<https://towardsdatascience.com/understanding-data-science-classification-metrics-in-scikit-learn-in-python-3bc336865019>

- Long, J., Shelhamer, E., & Darrell, T. (2015). Fully convolutional networks for semantic segmentation, *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 3431-3440. <https://doi.org/10.1109/CVPR.2015.7298965>.
- Maggiori, E., Tarabalka, Y., Charpiat, G., Alliez, P. (2017). Can Semantic Labeling Methods Generalize to Any City? The Inria Aerial Image Labeling Benchmark. *2017 IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*. Web: <https://project.inria.fr/aerialimagelabeling/>
- Manna, S. (2020, June 26). *K-Fold Cross Validation for Deep Learning Models using Keras*. Medium. <https://medium.com/the-owl/k-fold-cross-validation-in-keras-3ec4a3a00538>
- Mapbox Satellite: global base map & satellite imagery. (n.d.). Mapbox. <https://www.mapbox.com/maps/satellite>
- Muruganandham, S. (2016, August). *Semantic Segmentation of Satellite Images using Deep Learning [Masters thesis, Luleå University of Technology]*. Diva-portal <https://www.diva-portal.org/smash/get/diva2:1013270/FULLTEXT01.pdf>
- Narkhede, S. (2021, June 15). *Understanding AUC - ROC Curve - Towards Data Science*. Medium. <https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5>
- Nayem, A. B. S., Sarker, A., Paul, O., Ali, A. A., Amin, M. A., & Rahman, A. K. M. M. (2020). LULC Segmentation of RGB Satellite Image Using FCN-8. *CoRR*, abs/2008.10736. Opgehaald van <https://arxiv.org/abs/2008.10736>
- Noh, H., Hong, S., and Han, B. "Learning Deconvolution Network for Semantic Segmentation," *2015 IEEE International Conference on Computer Vision (ICCV)*, 2015, pp. 1520-1528, doi: 10.1109/ICCV.2015.178.

Normalization. (n.d.). Codecademy. <https://www.codecademy.com/articles/normalization>

Nowaczynski, A. (2021, January 5). *Cookie and Privacy Settings*. Deepsense.Ai.

<https://deepsense.ai/deep-learning-for-satellite-imagery-via-image-segmentation/>

Ozturk, O., Sariturk, B., & Seker, D. Z. (2020b). Comparison of Fully Convolutional Networks (FCN) and U-Net for Road Segmentation from High Resolution Imageries. *International Journal of Environment and Geoinformatics*, 7(3), 272–279.

<https://doi.org/10.30897/ijegeo.737993>

Parate, Akshata. (2020). Integrating Crisp DM Methodology for a Business Using Tableau Visualization. [Unpublished manuscript].

<http://dx.doi.org/10.13140/RG.2.2.36619.31520>.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Duchesnay, E. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12, 2825–2830.

Puc, J. (2020, February 18). *Examples of data augmentation, typical in computer vision, on a satellite image*. <https://medium.com/sentinel-hub/semi-supervised-learning-in-satellite-image-classification-e0874a76fc61>

Raghav, P. (2020, February 5). *Understanding of Convolutional Neural Network (CNN) — Deep Learning*. Medium.
<https://medium.com/@RaghavPrabhu/understanding-of-convolutional-neural-network-cn-n-deep-learning>

Ratan, P. (2021, January 14). *Convolutional Neural Network Architecture | CNN Architecture*.

Analytics Vidhya.

<https://www.analyticsvidhya.com/blog/2020/10/what-is-the-convolutional-neural-network-architecture/>

Ronneberger, O., Fischer P., Brox T. (2015). U-Net: Convolutional networks for biomedical image segmentation. *Medical Image Computing and Computer-Assisted Intervention. 2015 Springer, Cham, 9351*. https://doi.org/10.1007/978-3-319-24574-4_28

Sathya, P., & Malathi, L. (2011). Classification and Segmentation in Satellite Imagery Using Back Propagation Algorithm of ANN and K-Means Algorithm. *International Journal of Machine Learning and Computing*, 422–426. <https://doi.org/10.7763/ijmlc.2011.v1.63>

Semantic segmentation of aerial imagery. (2020). <https://www.kaggle.com/humansintheloop/semantic-segmentation-of-aerial-imagery>

Sharma, P. (2021, August 27). *Image Segmentation | Types Of Image Segmentation*. Analytics Vidhya. <https://www.analyticsvidhya.com/blog/2019/04/introduction-image-segmentation-techniques-python/>

Tiu, E. (2021, January 8). *Metrics to Evaluate your Semantic Segmentation Model*. Medium. <https://towardsdatascience.com/metrics-to-evaluate-your-semantic-segmentation-model-6bcb99639aa2>

Ulmas, P., Liiv, I. (2020). Segmentation of satellite imagery using u-net models for land classification. *arXiv:2003.02899*

Van Etten, V., A., Lindenbaum, D., & Bacastow, T.M. (2018). SpaceNet: A Remote Sensing Dataset and Challenge Series. *ArXiv, abs/1807.01232*.

Wang, H. (2020). Satellite buildings semantic segmentation data (version 6) [Data set].

<https://www.kaggle.com/hyyyrwang/buildings-dataset>

Wang, S., Chen, W., Xie, S. M., Azzari, G., & Lobell, D. B. (2020). Weakly Supervised Deep Learning for Segmentation of Remote Sensing Imagery. *Remote Sensing*, 12(2).

doi:10.3390/rs12020207

What is Work Breakdown Structure? (n.d.). Visual-Paradigm. Retrieved October 11, 2021, from.

<https://www.visual-paradigm.com/guide/project-management/>

what-is-work-breakdown-structure

Wirth, R., & Hipp, J. (2000). Crisp-dm: towards a standard process model for data mining. 2000

Semantic scholar. <https://www.semanticscholar.org/paper/Crisp-dm%3A-towards-a>

-standard-process-modelfor-Wirth-Hipp/48b9293cfd4297f855867ca278f7069abc6a9c24

Yoshihara, A., Hascoet, T., Takiguchi, T., & Ariki, Y. (2018). Satellite Image Semantic

Segmentation Using Fully Convolutional Networks.

<https://www.semanticscholar.org/paper/Satellite-Image-Semantic-Segmentation-Using-Fu>

lly-Yoshihara-Hascoet/9e31c11b678e4ba0cbb8cb96094a68e4969fb8b8

Zhou Z, Siddiquee MMR, Tajbakhsh N, Liang J. UNet++: Redesigning Skip Connections to

Exploit Multiscale Features in Image Segmentation. *IEEE Trans Med Imaging*. 2020

Jun;39(6):1856-1867. doi: 10.1109/TMI.2019.2959609. Epub 2019 Dec 13. PMID:

31841402; PMCID: PMC7357299.