

A REPORT  
ON  
**AUTOMATIC LAND COVER CLASSIFICATION  
OF TEMPORAL SATELLITE IMAGES**

BY

Guntaas Singh 2018A7PS0269P

Nisarg Vora 2018A7PS0254P

AT



Indian Institute of Remote Sensing, Dehradun

A Practice School - I station of



Birla Institute of Technology and Science, Pilani

June, 2020

A REPORT  
ON  
**AUTOMATIC LAND COVER CLASSIFICATION  
OF TEMPORAL SATELLITE IMAGES**

BY

ID Number	Name	Branch
2018A7PS0269P	Guntaas Singh	B.E. (Hons.) Computer Science
2018A7PS0254P	Nisarg Vora	B.E. (Hons.) Computer Science

**Prepared in the partial fulfillment of the**

Practice School - I course

AT



Indian Institute of Remote Sensing, Dehradun

A Practice School - I station of



Birla Institute of Technology and Science, Pilani

June, 2020

# Acknowledgments

We would sincerely like to thank the Director of Indian Institute of Remote Sensing, Dr. Prakash Chauhan, for giving us an opportunity to work in this organization and gain exposure to corporate work culture, ethics, and etiquette to be followed while working in a professional environment.

We would like to extend our most sincere gratitude to our project in-charge, Dr. Hari Shanker Srivastava, Head of the Programme Planning and Evaluation Group (PPEG) at IIRS, for providing us with the opportunity to work with him on this project, and his guidance and mentorship during the same.

We wish to extend our gratitude to the faculty in charge of the PS-I program at IIRS, Rekha A., Assistant Professor at BITS Pilani - Bangalore Center, for her guidance and advice during the PS-I program, and her helpfulness and responsiveness while addressing all the concerns we raised during the same.

In addition, we would also like to thank the members of the Practice School Division, for their hard work and dedication in operating the PS-I programme remotely to ensure that we have a seamless learning experience.

Guntaas Singh  
Nisarg Vora

**BIRLA INSTITUTE OF SCIENCE AND TECHNOLOGY  
PILANI (RAJASTHAN)  
Practice School Division**

**Station:** Indian Institute of Remote Sensing

**Centre:** Dehradun

**Duration:** 3 weeks

**Date of start:** 18th May, 2020

**Date of submission:** 6th June, 2020

**Title of project:** Automatic Land Cover Classification of Temporal Satellite Images

ID Number	Name	Branch
2018A7PS0269P	Guntaas Singh	B.E. (Hons.) Computer Science
2018A7PS0254P	Nisarg Vora	B.E. (Hons.) Computer Science

**Name of guide:** Dr. Hari Shanker Srivastava

**Designation:** Scientist/Engineer - SG. Group Head, Programme Planning and Evaluation Group (PPEG).

**Name of PS faculty:** Rekha A.

**Key Words:** Semantic Segmentation, Support Vector Machines, Convolutional Neural Networks, U-Net Architecture

**Project Areas:** Remote Sensing, Machine Learning, Deep Learning

## Abstract

This project attempts automatic land cover classification of different parts of India into forest, urban, agricultural land and water bodies which can be used for various applications like natural resource management, urban expansion etc. Data from Agra district, Uttar Pradesh has been used to train Support Vector Machines and Convolutional Neural Networks which are then tested on Gandhinagar district, Gujarat. Google Earth Engine has been used to obtain data from Landsat 8 satellite images. For the purpose of classification Normalized Difference Vegetation Index (NDVI) values are calculated by masking all other light bands except near-infrared and red light bands. Temporal images with NDVI labels are fed as input to the models and subsequently, the accuracy of these models is reported.

# Contents

Acknowledgments . . . . .	iii
Abstract . . . . .	iv
<b>1 Introduction</b>	<b>1</b>
1.1 About IIRS . . . . .	1
1.2 Remote Sensing . . . . .	1
1.3 Objective: Land Cover Classification . . . . .	1
1.3.1 Techniques . . . . .	2
1.3.2 Significance . . . . .	2
<b>2 Background</b>	<b>4</b>
2.1 Normalized Difference Vegetation Index(NDVI) . . . . .	4
2.1.1 What is NDVI? . . . . .	4
2.1.2 Formula . . . . .	4
2.1.3 Correlation with plant health . . . . .	5
2.2 Image Classification . . . . .	6
2.2.1 Pixel Based Approach . . . . .	6
2.2.2 Object Based Approach . . . . .	6
2.2.3 Semantic Approach . . . . .	6
2.3 Supervised Machine Learning . . . . .	7
2.4 Deep Learning and Neural Networks . . . . .	8
2.5 Convolutional Neural Networks . . . . .	9
2.5.1 Convolutional Layer . . . . .	9
2.5.2 Pooling Layer . . . . .	11
2.5.3 Fully Connected Layer . . . . .	11
2.6 Semantic Segmentation . . . . .	12
2.7 FCN Architecture . . . . .	12
2.7.1 Transpose Convolution . . . . .	13
2.7.2 Skip Connections . . . . .	13
<b>3 Methodology and Implementation</b>	<b>15</b>
3.1 Software Overview . . . . .	15
3.1.1 Google Earth Engine . . . . .	15
3.1.2 Google Colab . . . . .	15
3.1.3 Python . . . . .	15
3.1.4 Scikit-learn . . . . .	16

3.1.5	TensorFlow . . . . .	16
3.2	Data Collection . . . . .	16
3.2.1	Datasets . . . . .	16
3.2.2	Study Areas . . . . .	16
3.2.3	Data Preparation . . . . .	17
3.3	Data Preprocessing . . . . .	17
3.4	K Nearest Neighbors Classifier . . . . .	17
3.5	Decision Tree Classifier . . . . .	17
3.6	Support Vector Machines . . . . .	17
3.6.1	Types of SVMs . . . . .	18
3.6.2	Advantages . . . . .	18
3.6.3	Disadvantages . . . . .	18
3.7	U-Net Architecture . . . . .	19
3.7.1	Components . . . . .	19
3.7.2	Implementation . . . . .	21
3.7.3	Training . . . . .	22
3.7.4	Advantages . . . . .	22
3.7.5	Disadvantages . . . . .	23
<b>4</b>	<b>Results</b>	<b>24</b>
<b>5</b>	<b>Conclusion</b>	<b>25</b>
5.1	Future Scope . . . . .	25
	<b>Appendix</b>	<b>26</b>
	List of dependencies . . . . .	26
	Code snippets . . . . .	26
	<b>References</b>	<b>27</b>
	<b>List of Abbreviations</b>	<b>29</b>
	<b>Glossary</b>	<b>29</b>

# List of Figures

1.1	NDVI distribution in 2000 and 2015 in Kathmandu valley, Nepal, derived from MODIS data. Depicts changes in land cover/land use over time. . .	2
2.1	(1) Without NDVI. (2) With NDVI. . . . .	5
2.2	Plant health directly affects reflectance in NIR and RED bands. . . . .	5
2.3	A single neuron. . . . .	8
2.4	A two layer neural network with fully connected layers. . . . .	9
2.5	The convolution operation performed using a 3x3 filter on a 5x5x1 image. . . . .	10
2.6	(1) A typical max pooling layer. (2) The max pooling operation. . . . .	11
2.7	AlexNet - an influential NN architecture that popularized the use of CNNs and GPUs to accelerate deep learning. . . . .	12
2.8	A $3 \times 3$ transpose convolution with stride 2 and padding 1. The output gets up-sampled by a factor of 2. . . . .	13
2.9	The different variants of the FCN architecture defined in [13], with different skip connections added to the base network. . . . .	14
2.10	Different networks based on the FCN architecture. . . . .	14
3.1	Different views of the training location - Agra. (1) True colour satellite image (2) False colour infrared image (3) Map of NDVI values (4) Ground truth land cover. [Google Earth Engine] . . . . .	17
3.2	Support Vector Machine . . . . .	18
3.3	The U-Net architecture. . . . .	19

# **1. Introduction**

## **1.1 About IIRS**

Formerly known as Indian Photo-interpretation Institute (IPI), the institute was founded on 21st April, 1966 under the aegis of Survey of India (SOI). It was established in collaboration with the Government of The Netherlands on the pattern of Faculty of Geo-Information Science and Earth Observation (ITC) of the University of Twente, The Netherlands. The original idea of setting up the institute came from India's first Prime Minister - Pandit Jawahar Lal Nehru during his visit to The Netherlands in 1957.

Since its establishment in 1966, IIRS is a key player for training and capacity building in geo-spatial technology and its applications through training, education and research in Southeast Asia. The training, education and capacity building programmes of the institute are designed to meet the requirements of professionals at working levels, fresh graduates, researchers, academia, and decision makers. [8, 9]

## **1.2 Remote Sensing**

Remote sensing is the method of getting information about Earth and its surface without making any physical contact, primarily using satellites. Remote sensing is used in numerous fields, including geography, land surveying and most Earth science disciplines (for example, hydrology, ecology, meteorology, oceanography, glaciology, geology); it also has military, intelligence, commercial, economic, planning, and humanitarian applications.[22] It may be split into "active" remote sensing and "passive" remote sensing. In active remote sensing method, sensors are used to capture the reflection of the signals emitted by the satellite on different objects. In passive remote sensing, the reflection captured by the sensors is mainly due to sunlight. In this project, we are using passive remote sensing data collected by Landsat 8 satellite.

## **1.3 Objective: Land Cover Classification**

Land Cover classification generally refers to the categorization or classification of human activities and natural elements on the landscape within a specific time frame based on established scientific and statistical methods of analysis of appropriate source materials. Land cover refers to the surface cover on the ground like vegetation, urban infrastructure,



water, bare soil etc. Identification of land cover establishes the baseline information for activities like thematic mapping and change detection analysis.[23]

### 1.3.1 Techniques

Mainly two techniques are used for land cover classification: Field survey and Remotely sensed imagery analysis. Field survey was the primary method used for land cover classification until a few decades back. However, it was very inaccurate and time-consuming. With developments in technology, the conventional method of field survey which can be inaccurate and time consuming has been replaced by remotely sensed imagery analysis which is highly accurate and covers wider range.

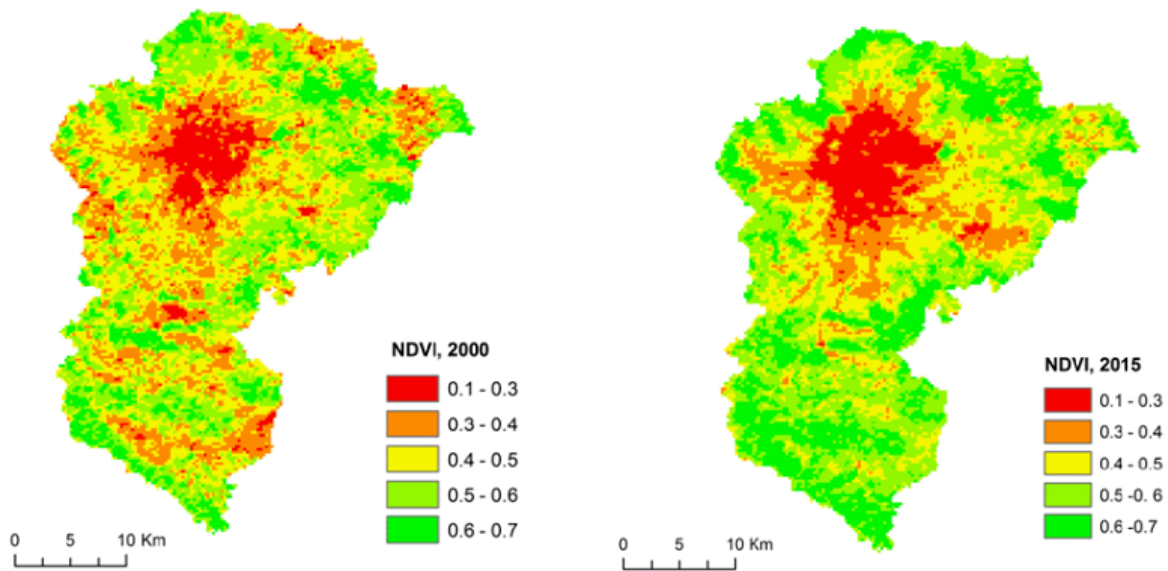


Figure 1.1: NDVI distribution in 2000 and 2015 in Kathmandu valley, Nepal, derived from MODIS data. Depicts changes in land cover/land use over time.

[1]

### 1.3.2 Significance

Land cover classification data is of great importance in the following applications:

- **Natural Resource Management:** Land cover classification can help policy makers decide how to best manage all the available resources by looking at different terrains. For example, the government can decide which area requires irrigation facilities the most using information about the land cover class distribution.

- **Wildlife habitat protection:** With shrinking habitats of wild animals, land cover classification can provide important data for helping in wildlife preservation and protection.
- **Urban Expansion/encroachment:** Proper planning is required for expansion of urban area. Different land cover features need to be studied to minimize the time and economic cost required for tackling natural obstacles to urban expansion.
- **Damage delineation (Tornadoes, flooding, fire, volcanic):** Live land cover data can provide us with information about where a certain natural disaster (like flood) has occurred/can occur. This can also help us to identify the areas at maximum risk of damage.
- **Target Detection - identification of land strips for use:** Using land cover data, identification of ideal locations for different urban construction becomes easy. For example, one can decide where exactly it is possible to build an airport, bridge etc.

## 2. Background

### 2.1 Normalized Difference Vegetation Index(NDVI)

Satellite data has become a very important parameter for judging crop progress, land cover classification etc. However, satellite data can not directly report crop health or classify land cover into different classes. NDVI which is calculated by measuring the reflectance of near-infrared and red light is probably the most important and accurate parameter for farmers to know crop health. NDVI values are so accurate that they can in fact, inform farmers about an upcoming drought upto two weeks in advance. Farmers can then make corresponding decisions to increase irrigation and other crop supplements.

#### 2.1.1 What is NDVI?

Light from the sun falls on Earth in three major light bands: ultraviolet, infrared and red. Most of the ultraviolet light is reflected back by the atmosphere. Healthy green plants absorb most of the visible light for photosynthesis. However, to prevent over-heating and dehydration the light in red band and near-infrared band is reflected back by the plants. The red light is visible to humans, while infrared is not. The NDVI values are calculated using the reflectance in red and near-infrared regions.

#### 2.1.2 Formula

NDVI is calculated in accordance with the formula:

$$NDVI = \frac{NIR - RED}{NIR + RED}$$

where,

$NIR$  = reflection in the near-infrared spectrum

$RED$  = reflection in the red range of the spectrum.

As per the formula above, the density of vegetation (NDVI) at a certain point of the image is equal to the difference in the intensities of reflected light in the red and infrared range divided by the sum of these intensities.

NDVI values range from -1.0 to 1.0, the positive values are usually for plants and vegetation where as the negative values usually indicate presence of snow, water or cloud cover. Values very close to 0 and less than 0.1 are due to presence of urban built ups

rocks etc. Moderate values between 0.2 and 0.3 are due to shrubs and crops whereas large values close to 1.0 represent temperate and tropical forests. These rough trends indicate the relevance of NDVI to the problem of land cover classification.



Figure 2.1: (1) Without NDVI. (2) With NDVI.

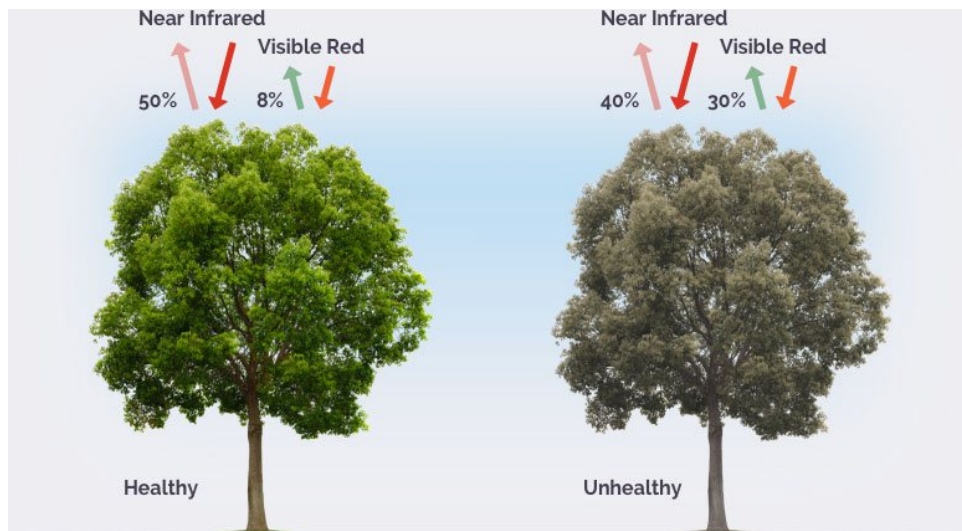


Figure 2.2: Plant health directly affects reflectance in NIR and RED bands.

### 2.1.3 Correlation with plant health

The amount of chlorophyll acts as a health indicator for the plants. Chlorophyll strongly absorbs visible light and the cell structure reflects near-infrared light. When chlorophyll levels are low, the cell structure gets destroyed and starts absorbing near-infrared light instead of reflecting it. Thus, value of NIR without vegetation decreases and hence the NDVI value becomes smaller.

## **2.2 Image Classification**

Image classification is a standard task in computer vision. In general, the image classification problem involves assigning one label out of a given fixed set of discrete labels to the input image on the basis of its visual content. While this is a trivial task for humans, robust image classification is a big challenge for a machine. To the computer, the image is just a grid of numbers which entirely change in unreliable ways with variations in viewpoint, illumination, occlusion, etc. As a result, there is no obvious algorithm which solves this problem. However, a data driven approach of providing the machine with many examples of each class and use of machine learning techniques has shown to be useful.[12]

There are different ways in which these techniques can be applied for classification of satellite imagery.

### **2.2.1 Pixel Based Approach**

In typical satellite images, pixel sizes are generally similar in size to the objects of interest. Most of the methods for image analysis using remote sensing data work on a per-pixel basis. However, with advances in remote sensing technology, the spatial resolution has become finer than the typical objects of interest, leading to an increase in within-class variability.[10]

### **2.2.2 Object Based Approach**

The term "objects" represents meaningful semantic entities or scene components that are distinguishable in an image.[10] This approach involves the partition of the image into meaningful geographical objects that share relatively homogeneous color, texture, etc.

### **2.2.3 Semantic Approach**

This aims to label each scene image with a specific semantic class. Here, a scene image refers to a local image patch manually extracted from large scale remote sensing images that contain explicit semantic classes.[10]

## 2.3 Supervised Machine Learning

Supervised learning is the machine learning task of learning a function that maps an input to an output on the basis of training data.[20, 14] The training data is the set of input-output example pairs used to train the classifier. If enough data is available then the trained model usually identifies the underlying data patterns and is then able to make predictions on previously unknown input. Supervised learning, however, faces the problem of overfitting when model is made very flexible in order to fit the underlying data and underfitting when the model is too rigid to fit the underlying data.

## 2.4 Deep Learning and Neural Networks

Application of traditional machine learning techniques requires handcrafted features, developing which demands a considerable amount of engineering skill and domain expertise. This, however, is not true for neural networks, which automatically learn these features from data using a general-purpose learning procedure.[10, 12]

A standard neural network consists of many simple, connected processors called neurons, each producing a sequence of real-valued activations. Input neurons get activated through data applied at the input of the network. Other neurons get activated through weighted connections from previously active neurons. [21] Each neuron can be seen as a single unit applying a non-linear activation function (such as sigmoid, tanh, ReLU) to a linear combination of the input activations to the neuron.[16]

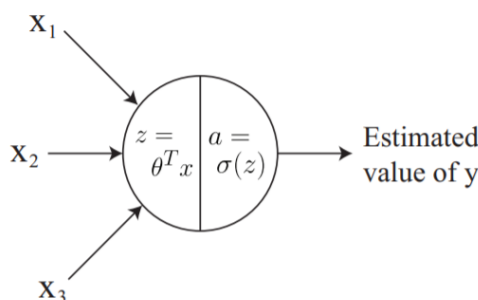


Figure 2.3: A single neuron.  
[16]

These single neurons can be stacked so that one neuron passes its output as input into the next neuron. The resulting network of neurons can, hence, consist of several layers of neurons, each with their own learnable weights and biases. Used in conjunction with an appropriate loss function and optimization algorithm, such a network can be used to learn any complex function, if sufficient data is available for training. Forward propagation through the network yields its prediction for a given input. This prediction is compared with the actual class label, and the loss is computed. Backward Propagation is used to compute the gradients of the loss function with respect to the parameters, which are then used by the optimization algorithm to adjust the parameters and minimize the loss over a number of iterations.

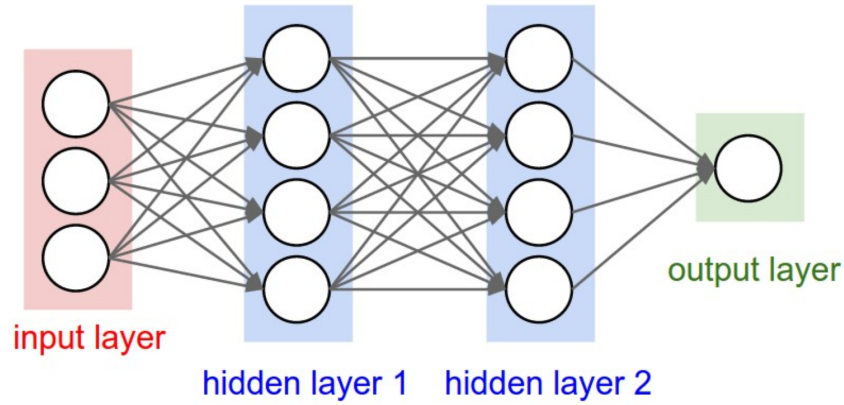


Figure 2.4: A two layer neural network with fully connected layers.  
[12]

## 2.5 Convolutional Neural Networks

Regular neural networks do not scale well to full images. If the input to the neural network is a  $200 \times 200$  RGB image, the number of weights for each neuron will be  $200 \times 200 \times 3 = 120,000$  weights. For large networks, the total number of learnable parameters become very large and lead the model to potentially overfit the training data, unless the training set is adequately large.

A convolutional neural network (CNN) is a sequence of layers. Each layers transforms an input volume (images are represented as a three dimensional matrix) of activations to another with some differentiable function which may or may not have parameters.[12, 17] These layers are of three main types:

### 2.5.1 Convolutional Layer

This is the core building block for convolutional networks. It is based on the convolution operation on images.

Each convolutional layer of a CNN consists of  $N$  kernels or filters of a certain volume of neurons sized  $f \times f \times d$ , with  $f$  being the spatial dimension and  $d$  being the number of feature channels of the kernel, which is same as the number of channels in the image at its input ( $D_i$ ). Every one of these filters is slid through the entire image of size  $H_i \times W_i \times D_i$ . Convolution refers to the summation of the element-wise dot product of the neurons in each filter with the corresponding values in the input, for each position in which the filter



is aligned with the image. Based on this notion, a convolution with a single filter at each layer results in a two dimensional output of a certain size. [12, 15, 17]

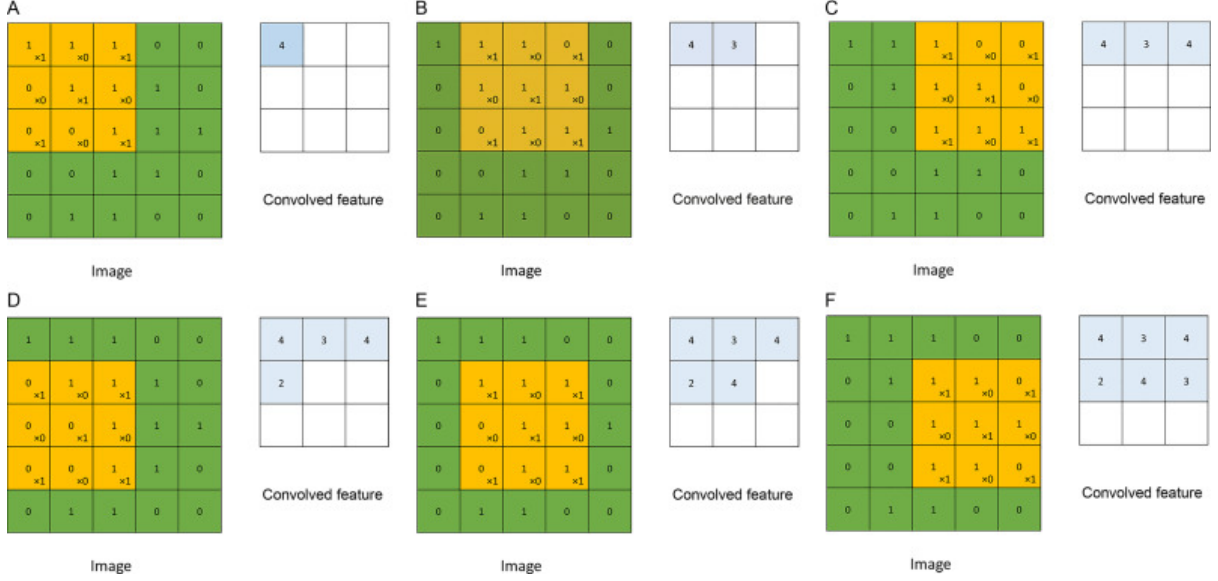


Figure 2.5: The convolution operation performed using a 3x3 filter on a 5x5x1 image. [2]

The intervals with which the filter moves in each spatial dimension is decided by the **stride**  $s$ . In order to prevent undue shrinkage of the volume along the spatial dimension, the image can be padded with pixels along the outer edges. The width of this **padding**, in pixels, is given by another hyper-parameter,  $p$ . [17, 15] The convolution operation is repeated for each of the  $N$  filters, and the resulting  $N$  activation maps are stacked together across the third dimension giving an output volume of dimensions:

$$H_o = \frac{H_i - f + 2p}{s} + 1$$

$$W_o = \frac{W_i - f + 2p}{s} + 1$$

$$D_o = N$$

In a convolutional layer, each neuron is connected to only a local region of the input volume, called the receptive field of that neuron. The extent of connectivity is limited to the filter size along the spatial dimensions, but is full along the depth axis. [12] It should also be noted that all activations belonging to a particular channel in the output volume correspond to a single filter applied on the input volume, and hence depend on the same shared parameters. Local connectivity and parameter sharing not only help reduce the

number of learnable parameters, but also make the CNN good at capturing **translation invariance**. This makes them an ideal choice for the image classification problem.

### 2.5.2 Pooling Layer

It is common to periodically insert a Pooling layer in-between successive convolutional layers in a CNN. Its function is to progressively reduce the spatial size of the representation to reduce the amount of parameters and computation in the network, and hence to also control overfitting.[12] The most common form of pooling layer in CNN architectures employs filters of size  $2 \times 2$  with a stride of 2, taking a max over 4 cells of the input image. It is worth noting that while this halves the width and height of the image, the depth remains unaffected as the max operation is applied independently to each channel of the image. This down-sampling process effectively discards 75% of the activations.

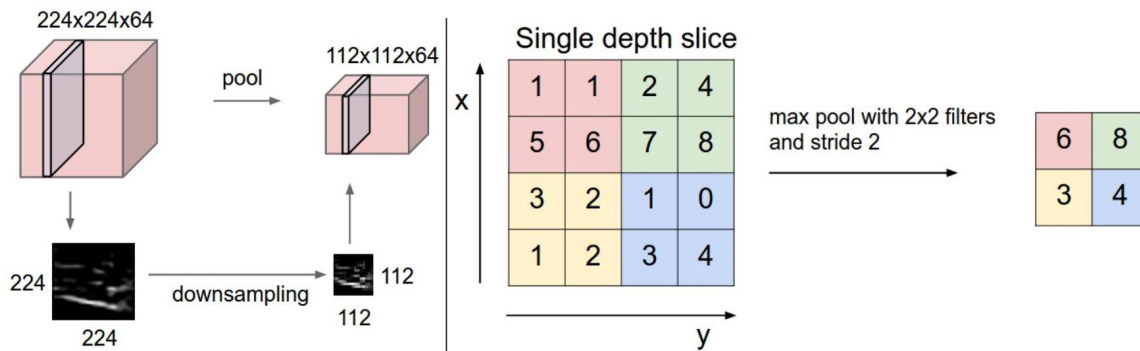


Figure 2.6: (1) A typical max pooling layer. (2) The max pooling operation.  
[12]

### 2.5.3 Fully Connected Layer

Once higher level features are detected from the preceding convolution and pooling layers, a fully connected layer is usually attached at the end of the network. This layer is fully connected to all activations in the previous layer, as in regular neural networks, allowing all the features learned by the network to be taken into account by the output layer.

All of these different types of layers can be stacked together in various ways to form a CNN.

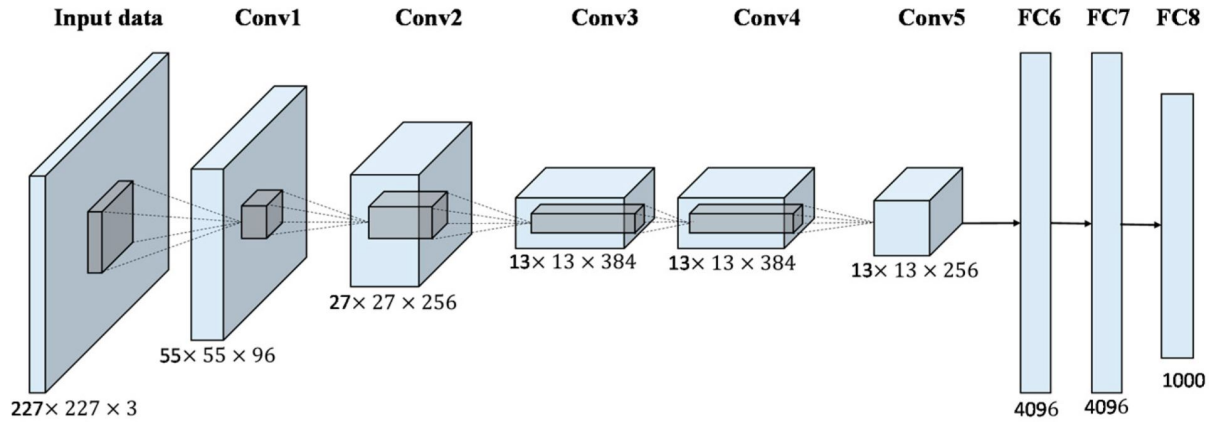


Figure 2.7: AlexNet - an influential NN architecture that popularized the use of CNNs and GPUs to accelerate deep learning.

[11, 7]

## 2.6 Semantic Segmentation

Typical CNNs used for image classification take fixed-size inputs and produce non-spatial outputs (predicted class labels). However, in case of land cover classification, the objective is not to assign a single class label to a given satellite image, but to divide the given image into segments each corresponding to one class out of a given set of land cover classes. This problem is known as semantic segmentation. There are different ways in which the existing image classification CNNs can be extended to solve this problem.

One approach is to use networks with only convolutional layers. In these networks, zero padding can be used to preserve the spatial dimensions of the input image. However, the large spatial dimensions throughout the network make the model computationally infeasible. [13, 12, 19]

While the problem requires the CNN to produce a spatial output, down-sampling is essential for reducing the need for computational resources to a feasible level. As a result, the most popular approach is to design the network as a combination of an encoder(down-sampling path) and a decoder(up-sampling path).

## 2.7 FCN Architecture

Fully connected layers have an equivalent representation as a convolutional layer having  $N$  filters with dimensions equal to those of the input image. The output of this layer will

thus be a volume of dimensions  $1 \times 1 \times N$ . This simple change allows the same CNN to be applied on images with arbitrary spatial dimensions and classify them in a single pass of forward propagation. This is far more efficient than iterating on different crops and classifying one pixel at a time. This is the basic intuition behind **Fully Convolutional Networks**. [13] For up-sampling the classification maps produced by a FCN, *Long et al.* [13] propose the use of transpose convolution.

### 2.7.1 Transpose Convolution

Ordinary convolution involves taking a dot product between the filter and the input for every receptive field at a stride of  $s$  in the input, and storing the result at a stride of 1 in the output. The resulting output image gets down-sampled by a factor of  $s$ . In transpose convolution, we use the values at a stride of 1 in the input and take their scalar product with the filter, storing the resulting matrix at a stride of  $s$  in the output. As a result the image gets up-sampled by a factor of  $s$ . Any overlapping values in the output are summed together. [13, 12] A stack of layers performing transpose convolution with an appropriate activation function can be used to learn a non-linear up-sampling. [13]

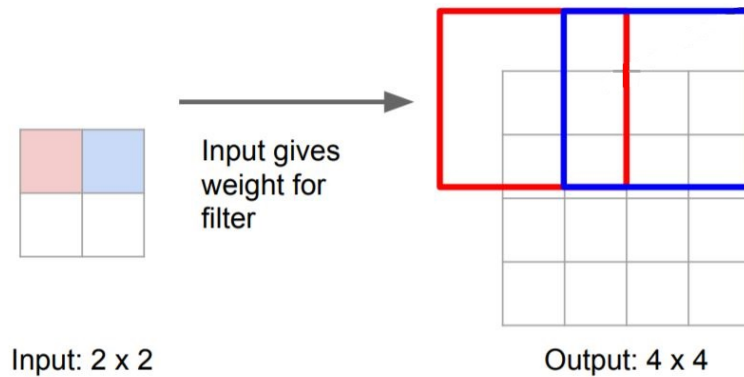


Figure 2.8: A  $3 \times 3$  transpose convolution with stride 2 and padding 1. The output gets up-sampled by a factor of 2.

[12]

### 2.7.2 Skip Connections

Semantic segmentation faces an inherent tension between semantics and location. Deep layers capture local information and resolve the semantic information that the image contains. Global information available to shallower layers can be used to capture the location. This motivates the idea of adding skip connections from shallow layers to the final prediction layer. Combining the information captured by shallow and deep layers

at the time of up-sampling lets the model make local predictions that take into account global structure and semantics. To combine activations with different spatial dimensions, they are first up-sampled individually as required using transpose convolutions and then summed up. The result is then up-sampled by a final layer, so that the output spatial dimensions match those of the input image. The addition of skip connections produces a finer and more detailed segmentation map.[13]

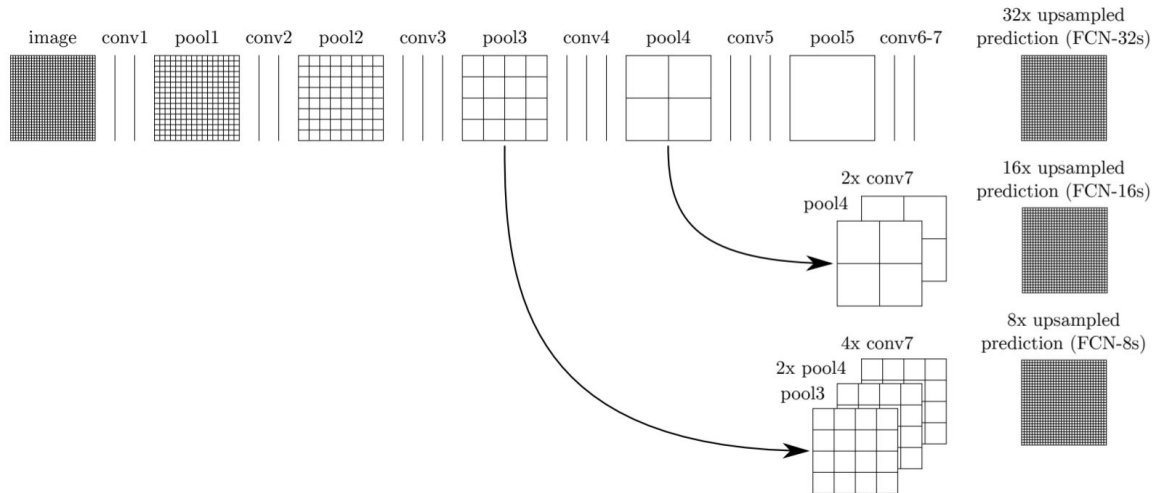


Figure 2.9: The different variants of the FCN architecture defined in [13], with different skip connections added to the base network.

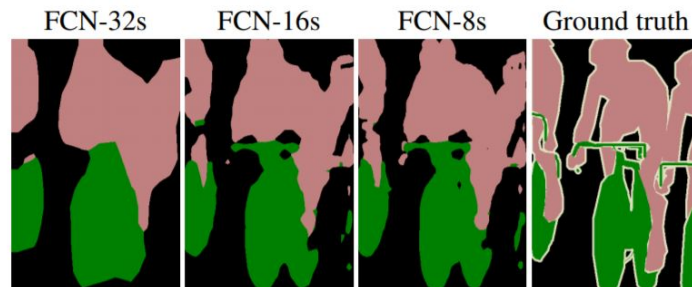


Figure 2.10: Different networks based on the FCN architecture.  
[13]

## 3. Methodology and Implementation

### 3.1 Software Overview

#### 3.1.1 Google Earth Engine

Google Earth Engine is a computing platform that allows users to run geo-spatial analysis on Google’s cloud infrastructure. It provides several ways to interact with the platform. The Code Editor is a web-based Integrated Development Environment (IDE) for writing and running scripts. The Explorer is a lightweight web app for exploring the data catalog and running simple analyses. The client libraries provide Python and JavaScript wrappers around the web Application Programming Interface (API).[6] In our project, the **JavaScript API** was used to visualize, prepare, process, and export satellite imagery and data.

#### 3.1.2 Google Colab

Google Colab is a hosted Jupyter notebook service that provides a web-based development environment for writing and executing arbitrary Python code within a browser. It is particularly well suited for machine learning and data analysis applications. It requires no setup to use, enables quick and easy collaboration and provides free access to computing resources like GPUs.[3] These computational resources are of great importance for training any classifier as they can significantly reduce training time. Since applying learning techniques is a very empirical process, the ability to iterate quickly is indispensable.

#### 3.1.3 Python

Python is an interpreted, high-level, general purpose programming language. It has an easy to learn syntax that emphasizes code readability. It supports multiple programming paradigms and has a comprehensive standard library. It also has a well-developed ecosystem of libraries for scientific computing and data science such as **NumPy**, **Pandas** and **Matplotlib**. These libraries provide data structures and methods for handling large amount of data and efficient computation. In addition to these libraries, in our project, we also used **Pillow** (Python Imaging Library) for image processing and **GDAL** (Geospatial Data Abstraction Library) for handling images in the **GeoTIFF** file format.

### 3.1.4 Scikit-learn

### 3.1.5 TensorFlow

## 3.2 Data Collection

### 3.2.1 Datasets

#### Landsat 8

We are using satellite imagery from the *Landsat 8 - Surface Reflectance - Tier 1* product. The data is collected by Landsat 8 - an American Earth Observation satellite launched in 2013 under Landsat, a joint program of the United States Geological Survey (USGS) and National Aeronautics and Space Administration (NASA). The satellite images the entire Earth's surface at a spatial resolution of 30m, once every two weeks and records multi-spectral and thermal data. The Operational Land Imager (OLI) collects passive remote sensing data from nine spectral bands including the Near Infrared and Visible bands which we intend to use for the project. The data has been atmospherically corrected in addition to standard geometric adjustments and geo-referencing. Information from the thermal infrared sensor is also used to mask clouds. The imagery, hence, provides a clear view of land. The high temporal resolution also helps to factor in temporal variations in land cover. This makes the dataset suitable for the purpose of our project.[5, 24]

#### Copernicus Global Land Cover dataset

The *Copernicus Global Land Service (CGLS) - LC100 collection 2* is earmarked as a component of the Land service to operate a multi-purpose service component that provides a series of bio-geophysical products on the status and evolution of land surface at global scale. Derived from the PROBA-V 100m time series data for the year of 2015, this dynamic land cover map at 100 m spatial resolution provides a primary land cover scheme. Next to these discrete classes, the product also includes continuous field layers for all basic land cover classes that provide proportional estimates for vegetation/ground cover for the land cover types.[4]

### 3.2.2 Study Areas

For training our classifiers, we will be using images of Agra, Mathura and the surrounding region, Uttar Pradesh. This location was chosen as it has a relatively even distribution of different land cover classes. Additionally, first-hand knowledge of the land cover of



the region has been provided by the mentor which should help us refine the data. For testing, we will use images of Gandhinagar, Gujarat because of similar reasons.

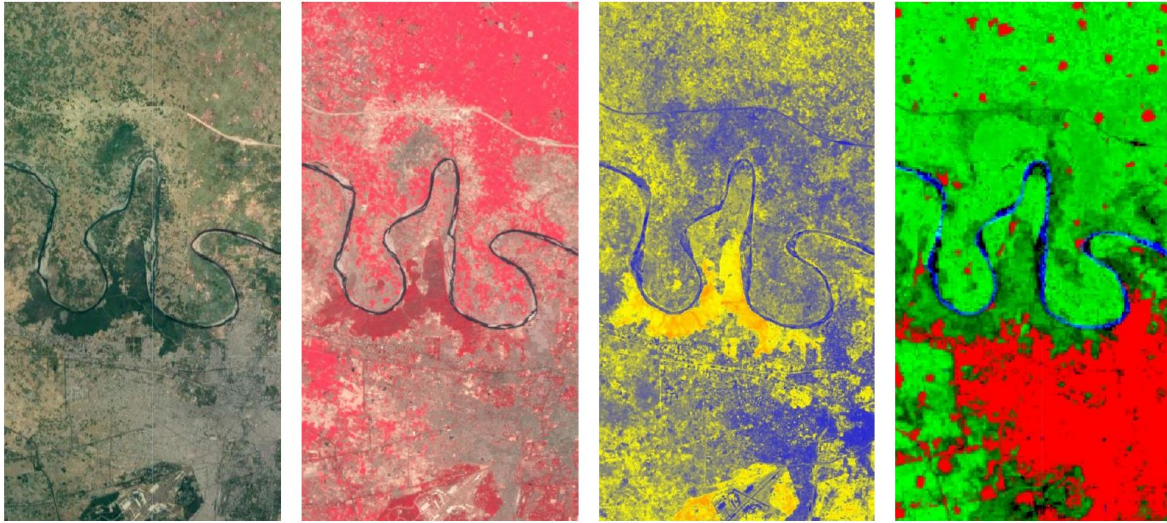


Figure 3.1: Different views of the training location - Agra. (1) True colour satellite image (2) False colour infrared image (3) Map of NDVI values (4) Ground truth land cover. [Google Earth Engine]

### 3.2.3 Data Preparation

## 3.3 Data Preprocessing

## 3.4 K Nearest Neighbors Classifier

## 3.5 Decision Tree Classifier

## 3.6 Support Vector Machines

A Support Vector Machine (SVM) is a discriminative classifier formally defined by a separating hyperplane. In other words, given labeled training data (supervised learning), the algorithm outputs an optimal hyperplane which categorizes new examples. In two dimensional space, this hyperplane is a line dividing a plane into two parts where in each class lay on either side.[18]



### 3.6.1 Types of SVMs

There are two kinds of support vector machines: Hard Margin Support Vector Machines and Soft Margin Support Vector Machines. Hard Margin Support Vector machines do not allow misclassification of data points, tend to overfit the data and are thus sensitive to outliers. Soft margin support vector machines allow slight misclassification of data points and penalize for each misclassification. For this project, soft margin support vector machines have been used to classify the data points.

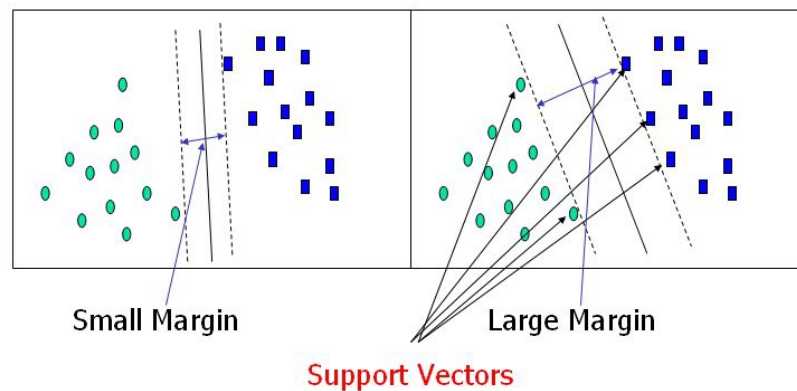


Figure 3.2: Support Vector Machine

### 3.6.2 Advantages

The advantages of using support vector machines are:

- Support vectors can deal with high dimensional data easily with the help of kernel transformations.
- They are only dependent on some part of the training data known as the support vectors which makes them very memory efficient.
- Can be used for a variety of tasks due to availability of a large number of kernel functions for decision making.
- Fast and memory efficient implementations in various languages is available via different libraries.

### 3.6.3 Disadvantages

The disadvantages of using support vector machines are:

- They do not take the spatial orientation of the pixels into account while learning.
- With wrong choice of kernel, support vector machines tend to overfit the data.
- Hard margin support vector machines are very sensitive to outliers and hence proper care should be taken while using them.

## 3.7 U-Net Architecture

Proposed by *Ronneberger et al.*[19], the U-Net architecture builds upon and extends the FCN architecture for more precise segmentation using a smaller training dataset. While the architecture was designed particularly for bio-medical segmentation applications, it has shown to work well for other domains too.

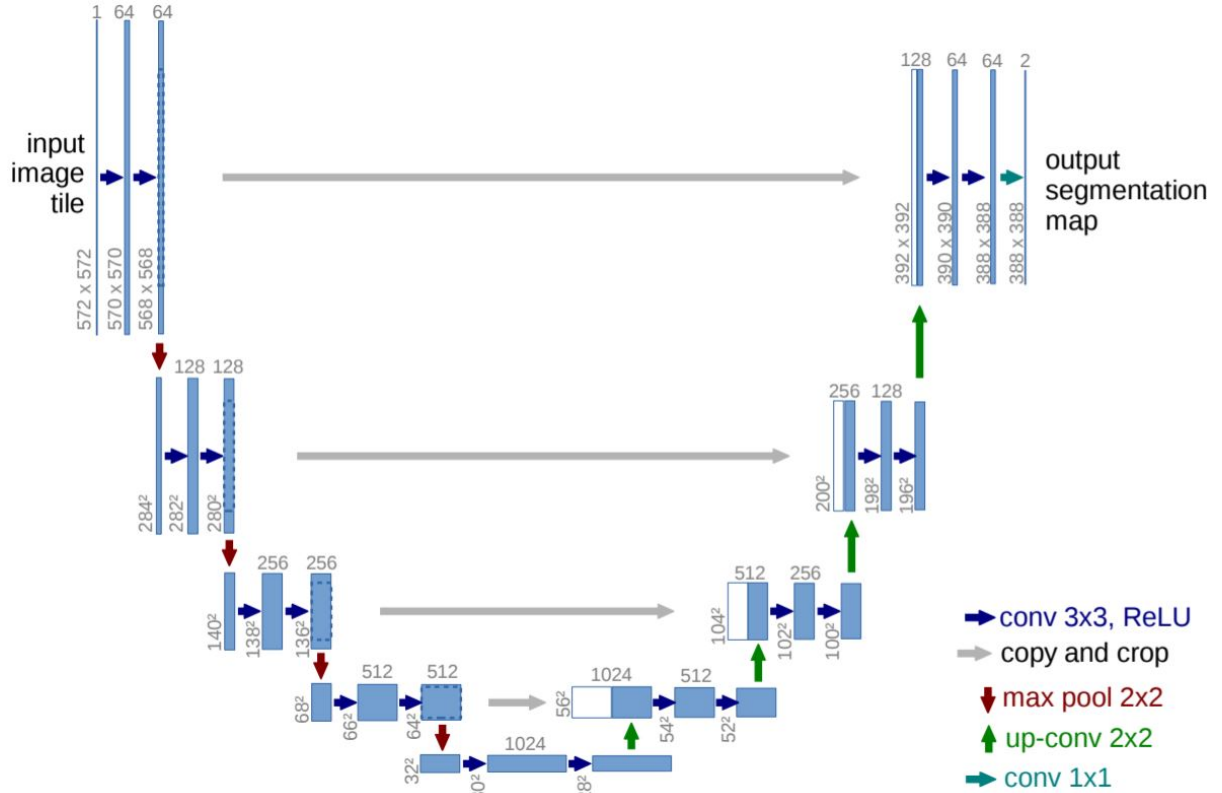


Figure 3.3: The U-Net architecture.  
[19]

### 3.7.1 Components

The U-Net architecture can be divided into three major parts:

1. The down-sampling path

2. The bottleneck
3. The up-sampling path

### **Down-sampling Path**

It is made up of four identical blocks. Each block encloses the following sequence of layers:

1. Convolutional layer ( $f = 3, s = 1, p = 0$ )
2. Convolutional layer ( $f = 3, s = 1, p = 0$ )
3. Max Pooling layer ( $f = 2, s = 2$ )

At each down-sampling step (max-pooling), the number of channels is doubled, starting with 64 channels in the first block. The down-sampling path captures not only features relevant to classification, but also contextual information about the location of different segments.

### **Bottleneck**

It is made up of 2 layers:

1. Convolutional layer ( $f = 3, s = 1, p = 0$ )
2. Convolutional layer ( $f = 3, s = 1, p = 0$ )

### **Up-sampling Path**

It is made up of four identical blocks. Each block encloses the following sequence of layers:

1. Up-sampling transpose convolution layer ( $f = 2, s = 2$ )
2. Concatenation with appropriately cropped feature map from down-sampling path
3. Convolutional layer ( $f = 3, s = 1, p = 0$ )
4. Convolutional layer ( $f = 3, s = 1, p = 0$ )

At each up-sampling step (transpose convolution), the number of channels is halved. The output volume after up-sampling is concatenated with the corresponding volume from the down-sampling path after adequate cropping. The introduction of these skip connections allows the propagation of context information about the localization from the down-sampling path to the up-sampling path which already carries semantic information. This helps the network to spread out activations correctly when up-sampling and lends the network its U-shape.

### 3.7.2 Implementation

A convolutional neural network based on the above architecture was implemented using TensorFlow in Python. However, some changes were made to enhance the performance of the model.

1. Convolutional layers down-sample the image, leading to shrinkage of the image’s spatial dimensions. This is particularly troublesome for skip-connections as concatenation of volumes at the time of up-sampling requires that the width and height of the volumes match. Using **same padding** before applying the convolution operation prevents this down-sizing and solves the problem. This also enhances the symmetry in the network, with the output layer having the same spatial dimensions as the input image, reducing the overhead of image processing which would otherwise be required for training the network.
2. As a neural network is trained, the distribution of each layer’s inputs changes continuously, as the parameters of previous layers change. This slows down learning and necessitates careful parameter initialization. This problem of internal covariate shift can be addressed by normalizing layer inputs to a learnable mean and variance, stabilizing the distribution of the activations. **Batch Normalization** allows us to use higher learning rates and makes the network more robust to the choice of hyper-parameters. Additionally, it also has a regularizing effect and helps prevent overfitting.
3. The shape of image accepted as input by the network was fixed to  $256 \times 256$ . A patch size of 256 was found to provide sufficient spatial context to the network and fit well with the computational constraints.

The model implemented has 31,051,332 parameters, of which, 31,039,556 are trainable. The arrangement of different layers and shapes of output volumes can be seen in the figure below. A more detailed plot can be found in the Appendix.



- The U-Net architecture is also particularly well suited to applications where labeled training data is not available in abundance. This makes it more attractive for our project on account of the limited computational resources and time available for training the network. We, hence, propose to use this CNN architecture for the problem of land cover classification.

### 3.7.5 Disadvantages

- Convolutional neural networks do not take temporal variations directly into account. However, it is hoped that providing the network with training data from different time periods should allow it to capture some of the temporal trends.
- The performance of a deep neural network greatly depends on the amount of training data available. We propose using techniques like data augmentation and transfer learning so as to get better results even with our small training set.

## 4. Results

## 5. Conclusion

Information of land cover classification is very crucial for applications like natural resource management, wildlife habitat protection, urban expansion, damage delineation, target land detection etc. With developments in technology, conventional method of field survey which is both inaccurate and time consuming has been replaced by remotely sensed imagery analysis which is highly accurate and covers wider range. Indian Space Research Organisation (ISRO) and Indian Institute of Remote Sensing (IIRS), Dehradun play an important role in providing information for these applications. Modern statistical and computing techniques like machine learning and deep learning minimize the human intervention in the task of analysis and classification, thus making the outcome very accurate and the process very agile.

The project seeks to compare the applicability of machine learning and deep learning techniques to the problem of land cover classification. Support Vector Machines are traditional machine learning models that provide a method for pixel-wise classification based on learning from the temporal variations in the NDVI time series. On the other hand, Convolutional Neural Networks - a class of deep neural networks, use the spatial variations in the infrared and visible bands of the satellite image for segmentation. The two, hence represent very different approaches for solving the problem. However, the project is still at a very nascent stage, and any comment regarding the comparative performance of these two approaches cannot be made yet.

### 5.1 Future Scope



# Appendix

List of dependencies

Code snippets

# References

- [1] Binod Baniya et al. “A Review of Green Roofs to Mitigate Urban Heat Island and Kathmandu Valley in Nepal”. In: 6 (Nov. 2018), pp. 137–152. DOI: 10.12691/aees-6-4-5.
- [2] Ehsan Fathi and Babak Maleki Shoja. “Deep neural networks for natural language processing”. In: *Handbook of statistics*. Vol. 38. Elsevier, 2018, pp. 229–316. (Visited on 06/02/2020).
- [3] Google. *Google Colaboratory - frequently asked questions*. URL: <https://research.google.com/colaboratory/faq.html> (visited on 06/20/2020).
- [4] Google. *Google Earth Engine - Copernicus Global Land Cover*. URL: [https://developers.google.com/earth-engine/datasets/catalog/COPERNICUS\\_Landcover\\_100m\\_Proba-V\\_Global](https://developers.google.com/earth-engine/datasets/catalog/COPERNICUS_Landcover_100m_Proba-V_Global) (visited on 06/05/2020).
- [5] Google. *Google Earth Engine - Landsat 8*. URL: [https://developers.google.com/earth-engine/datasets/catalog/LANDSAT\\_LC08\\_C01\\_T1\\_SR](https://developers.google.com/earth-engine/datasets/catalog/LANDSAT_LC08_C01_T1_SR) (visited on 06/05/2020).
- [6] Google. *Google Earth Engine - Platform Overview*. URL: <https://earthengine.google.com/platform/> (visited on 06/04/2020).
- [7] Xiaobing Han et al. “Pre-trained alexnet architecture with pyramid pooling and supervision for high spatial resolution remote sensing image scene classification”. In: *Remote Sensing* 9.8 (2017), p. 848.
- [8] *History—Indian Institute of Remote Sensing*. URL: <https://www.iirs.gov.in/historyandobjectives> (visited on 06/01/2020).
- [9] *Indian Institute of Remote Sensing*. URL: <https://www.iirs.gov.in/institute-profile> (visited on 06/01/2020).
- [10] Manideep Kolla, Aniket Mandle, and Apoorva Kumar. *Eye in the sky*. GitHub Page. 2018. URL: <https://github.com/manideep2510/eye-in-the-sky> (visited on 06/01/2020).
- [11] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. “Imagenet classification with deep convolutional neural networks”. In: *Advances in neural information processing systems*. 2012, pp. 1097–1105.
- [12] Fei-Fei Li, Justin Johnson, and Serena Yeung. *cs231n - Lecture Slides*. 2017. URL: <http://cs231n.stanford.edu/slides/2017/> (visited on 06/01/2020).

- [13] Jonathan Long, Evan Shelhamer, and Trevor Darrell. “Fully convolutional networks for semantic segmentation”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 3431–3440.
- [14] Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar. In: *Foundations of Machine Learning*, 2012.
- [15] Shivaprakash Muruganandham. *Semantic segmentation of satellite images using deep learning*. 2016.
- [16] Andrew Ng and Kian Katanforoosh. *cs229 - Lecture Notes*. 2018. URL: <http://cs229.stanford.edu/notes/> (visited on 06/01/2020).
- [17] Andrew Ng, Kian Katanforoosh, and Younes Bensouda Mourri. *Convolutional Neural Networks*. 2017. (Visited on 06/02/2020).
- [18] Savan Patel. *Chapter 2: SVM (Support Vector Machines)*. 2017. (Visited on 06/05/2020).
- [19] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. “U-net: Convolutional networks for biomedical image segmentation”. In: *International Conference on Medical image computing and computer-assisted intervention*. Springer. 2015, pp. 234–241.
- [20] Stuart J. Russell and Peter Norvig. In: *Artificial Intelligence: A Modern Approach, Third Edition*. 2010.
- [21] Jürgen Schmidhuber. “Deep learning in neural networks: An overview”. In: *Neural networks* 61 (2015), pp. 85–117.
- [22] Robert A. Schowengerdt. In: *Remote sensing: models and methods for image processing (3rd ed.)*. 2007.
- [23] Yashwant Singh. *Significance of land cover/land use*. URL: <https://www.satpalda.com/blogs/significance-of-land-use-land-cover-lulc-maps> (visited on 06/05/2020).
- [24] United States Geological Survey. *Landsat 8*. URL: <https://www.usgs.gov/land-resources/nli/landsat/landsat-8> (visited on 06/05/2020).

## List of Abbreviations

<b>CNN</b>	Convolutional Neural Networks
<b>ET</b>	Evapotranspiration
<b>FCN</b>	Fully Convolutional Neural Networks
<b>GEE</b>	Google Earth Engine
<b>GPU</b>	Graphical Processing Unit
<b>IIRS</b>	Indian Institute of Remote Sensing
<b>NDVI</b>	Normalised Difference Vegetation Index
<b>NIR</b>	Near InfraRed
<b>NN</b>	Neural Networks.
<b>SVM</b>	Support Vector Machines

## Glossary

**Chlorophyll** Green photosynthetic pigment found in plants

**Convolution** mathematical way of combining two signals to form a third signal

**Kernel** Kernel functions.

**Pixel** a minute area of illumination on a display screen, one of many from which an image is composed.

**Pooling** process of extracting features from image output of a convolution layer

**Semantic Segmentation** the process of associating each pixel of an image with a class label

**Temporal** relating to time