SNOW COVER MAPPING IN INDIAN HIMALAYAS

JITHU J
ELECTRONICS AND ELECTRICAL ENGINEERING DEPARTMENT
IIT GUWAHATI

END SEMESTER REPORT

22 April 2024

Abstract

Accurate mapping of snow cover in mountainous regions is crucial for various applications, including hydrology, climate modeling, and disaster management. Satellite imagery offers a valuable resource for monitoring snow cover over large areas; however, traditional methods for snow cover mapping often face challenges in complex terrains such as the Indian Himalayas. I this project we try to Map the snow cover using machine learning algorithms and compute the total area.

Our methodology involves preprocessing of multi-spectral satellite imagery to extract relevant features, including spectral bands, topographic attributes, and meteorological data. We then employ machine learning algorithms identify the snow pixels and snowless pixels.

we combine our results with NDSI index generated results to get a better output.Overall, this study highlights the potential of machine learning techniques for enhancing snow cover mapping in challenging mountainous regions like the Indian Himalayas.

Contents

1 Introduction **3**

1.1 Objectives 3

2 Background **5**

3 Design **9**

4 Implementation **13**

5 Results **19**

6 Conclusions **20**

6.1 Bibiliography 20

Chapter 1

Introduction

The Indian Himalayas, characterized by their rugged terrain and extreme climatic conditions, are home to vast expanses of snow-covered landscapes, playing a critical role in regional hydrology, ecology, and livelihoods. Accurate monitoring and mapping of snow cover in this region are essential for various applications, including water resource management, climate change studies, and hazard mitigation. Satellite remote sensing offers a powerful tool for observing snow cover dynamics over large spatial extents; however, traditional methods for snow cover mapping often encounter challenges in complex mountainous terrains.

In recent years, the advent of machine learning (ML) techniques has provided new avenues for improving the accuracy and efficiency of snow cover mapping from satellite imagery. ML algorithms have demonstrated remarkable capabilities in image classification tasks, leveraging complex patterns and spectral information to accurately delineate snow-covered areas.

Here we are trying to use ML techniques to identify the pixels which contains snow. We make use of landsat 8 data which is freely available on internet.

1.1 Objectives

Segment the image and find amount of snow cover in the image.

- Download and combine satellite images for state of sikkim and nearby areas
- Use ML algorithms to identify snow and snowless pixels in the image.
- Calculate the area in sq kms based on resolution

Chapter 2

Background

FRS(Fractional Snow Cover)::

Fractional snow cover refers to the proportion of a specific area that is covered by snow. It is often expressed as a fraction or a percentage, indicating the amount of ground or surface area covered by snow compared to the total area being observed. Fractional snow cover is a critical parameter in various fields such as climatology, hydrology, and ecology, as it affects local climate, water availability, and habitat suitability for various organisms. It can be monitored and measured using various techniques, including satellite imagery, ground observations, and remote sensing technologies.

NDSI (Normalized Difference Snow Index)::

It is a spectral index commonly used in remote sensing to detect the presence of snow in satellite imagery. NDSI is designed to highlight the contrast between snow and other surfaces, such as vegetation or soil, based on their reflectance properties in different spectral bands. Higher NDSI values typically indicate a higher likelihood of snow cover, while lower values indicate less snow cover or the presence of other surface types.

NDFSI (Normalised Difference Forested Snow Index)::

It is a spectral index proposed in the paper "An effective fractional snow cover Estimation method using deep feature snow Index" by Yuhan Wang, Lingjia Gu[1]. This index is a derivative of NDSI index but instead of using visible bands this index makes use of Near infrared band and SWIR band of landsat 8 data to predict the snow pixels.

K-Means Clustering::

K-means clustering is a popular unsupervised machine learning algorithm used for clustering data into distinct groups or clusters based on similarity. The algorithm aims to partition the data points into k clusters, where each data point belongs to the cluster with the nearest mean (centroid).

C-NN (Convolutional Neural network)::

A Convolutional Neural Network (CNN) is a type of deep learning neural network that is primarily used for analyzing visual imagery. CNNs are highly effective in tasks such as image classification, object detection, and image segmentation. CNNs are structured to automatically and adaptively learn spatial hierarchies of features from the input images. They achieve this through a series of layers.

Landsat 8 and USGS Earth Explorer::

Landsat 8 is a satellite launched by NASA (National Aeronautics and Space Administration) and the USGS (United States Geological Survey) as part of the Landsat program. It was launched on February 11, 2013, and continues the legacy of Landsat satellite missions, which have been capturing Earth observation data for several decades.

The USGS Earth Explorer is an online platform provided by the United States Geological Survey (USGS) for accessing and downloading a wide range of Earth observation data, including satellite imagery, aerial photography, and other geospatial datasets. It offers access to data from various satellite missions, including Landsat, Sentinel, and other remote sensing programs. We can download satellite data from USGS explorer as spectral bands.

	Bands	Wavelength (micrometers)	Resolution (meters)
Landsat 8 Operational Land Imager (OLI) and Thermal Infrared Sensor (TIRS) Launched February 11, 2013	Band 1 - Coastal aerosol	0.43 - 0.45	30
	Band 2 - Blue	0.45 - 0.51	30
	Band 3 - Green	0.53 - 0.59	30
	Band 4 - Red	0.64 - 0.67	30
	Band 5 - Near Infrared (NIR)	0.85 - 0.88	30
	Band 6 - SWIR 1	1.57 - 1.65	30
	Band 7 - SWIR 2	2.11 - 2.29	30
	Band 8 - Panchromatic	0.50 - 0.68	15
	Band 9 - Cirrus	1.36 - 1.38	30
	Band 10 - Thermal Infrared (TIRS) 1	10.60 - 11.19	100
	Band 11 - Thermal Infrared (TIRS) 2	11.50 - 12.51	100

Figure 2.1: landsat 8 bands

QSGS (Quantum Geographic Information System)

QGIS is a geographic information system software that is free and open-source. QGIS supports Windows, macOS, and Linux. It supports viewing, editing, printing, and analysis of geospatial data in a range of data formats. QGIS was previously also known as Quantum GIS.

UNET

U-Net is a convolutional neural network that was developed for biomedical image segmentation at the Computer Science Department of the University of Freiburg.[1] The network is based on a fully convolutional neural network[2] whose architecture was modified and extended to work with fewer training images and to yield more precise segmentation

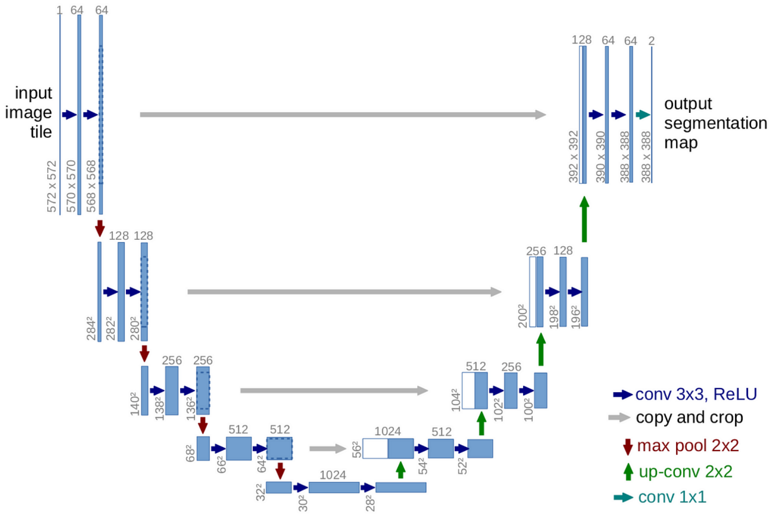

Figure 2.2: Unet Architecture

Chapter 3

Design

Acquiring data sets::

The primary data used in the project is landsat 8 satellite data. We can download such satellite data from USGS earth explorer. The data from landsat is recieved as multiple bands. At present we are interested in bands 2,3,4,5 and 6.

band 2:: Blue band
band 3:: Green band
band 4:: Red band
band 5:: Near-Infrared band
band 6:: SWIR 1 - Short-Wave Infrared band 1

Now the process is to use QGIS software to import the said bands. There is a function in QGIS called raster calculator with which we can perform arithematic operations on the imported band data. We use this fuction to calculate the NDSI and NDFSI indices and create the respective masks.

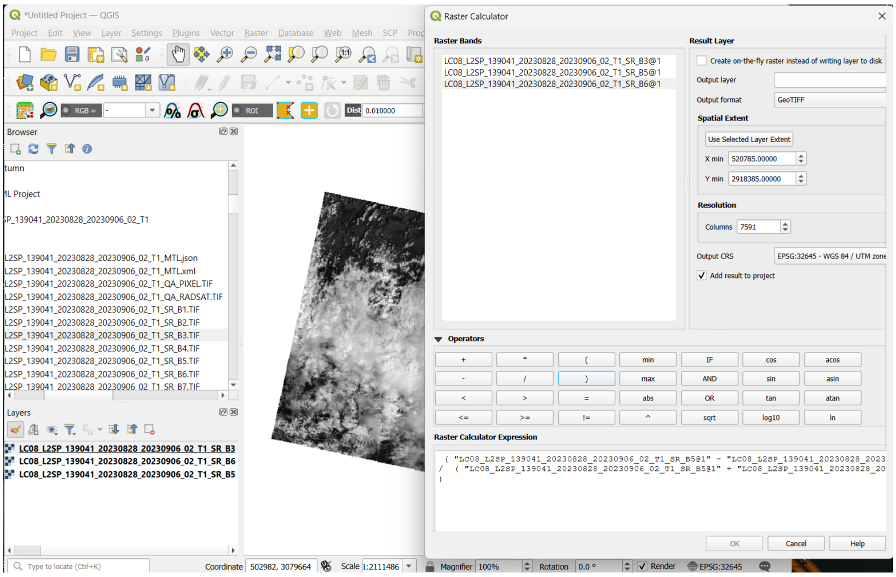

Figure 3.1: QGIS interface with NDFSI equation coded into Raster calculator

K-means Clustering::

The RGB image produced will have snow as white pixels and vegetation and soil an green and brown pixels. Since snow is white which will have a higher pixel intensity value than others we can cluster them using k-means clustering. Here we plan to use 3 clusters.

But since this is satellite images there will be clouds present in the image which will also be in white pixels. That means the cluster with snow present will also have clouds in them.

NDSI-Normalized Difference Snow Index::

NDSI is a technique to classify pixels which have snow present in them. It uses green band and SWIR-1 band to predict if the pixel contains snow or not. We can use the following Equation for calculating NDSI. The Ndsi threshold is usually set to be 0.4. The equation for Normalized Difference Snow Index (NDSI) is given by:

$$\text{NDSI} = \frac{\text{Green} + \text{SWIR}}{\text{Green} - \text{SWIR}} \tag{3.1}$$

NDFSI-

Is a spectral index used to estimate fractional snow cover when the area of interest is a forested region. It employs non-visible spectral indices such as near infrared band(band 5) and Short Wave infrared band (band 6). in this project we have used 0.4 as threshold for NDFSI. A sample mask from qgis software is shown below. We use the following Equation for NDSFI calucula-tion.

$$\mathrm{NDSI} = \frac{\mathrm{NIR} + \mathrm{SWIR}}{\mathrm{NIR} - \mathrm{SWIR}} \tag{3.2}$$

U-Net model::

We have developed a dataset consisting of landsat 8 bands from many dates and along with them we have generated NDFSI and NDSI masks correspond-ing to this data. From multiple research papers listed at the end we have come to a conclusion that the green, NIR and SWIR bands contains most informa-tion regarding presence of snow. We use these 3 bands as input to the U-net architecture and mask as output to train the model. The trained model is then used to predict new masks.

Calculating The Area of Snow Cover::

From the generated final images we can calculate how many pixels are classi-fied as snow. The spacial resolution of landsat 8 images are 30m that means each pixel is (30*30)m^2 We can use the equation below for finding the area.
The equation for the total area is given by:

$$\text{total area} = \text{number of pixels} \times 900\,\mathrm{m}^2 \tag{3.3}$$

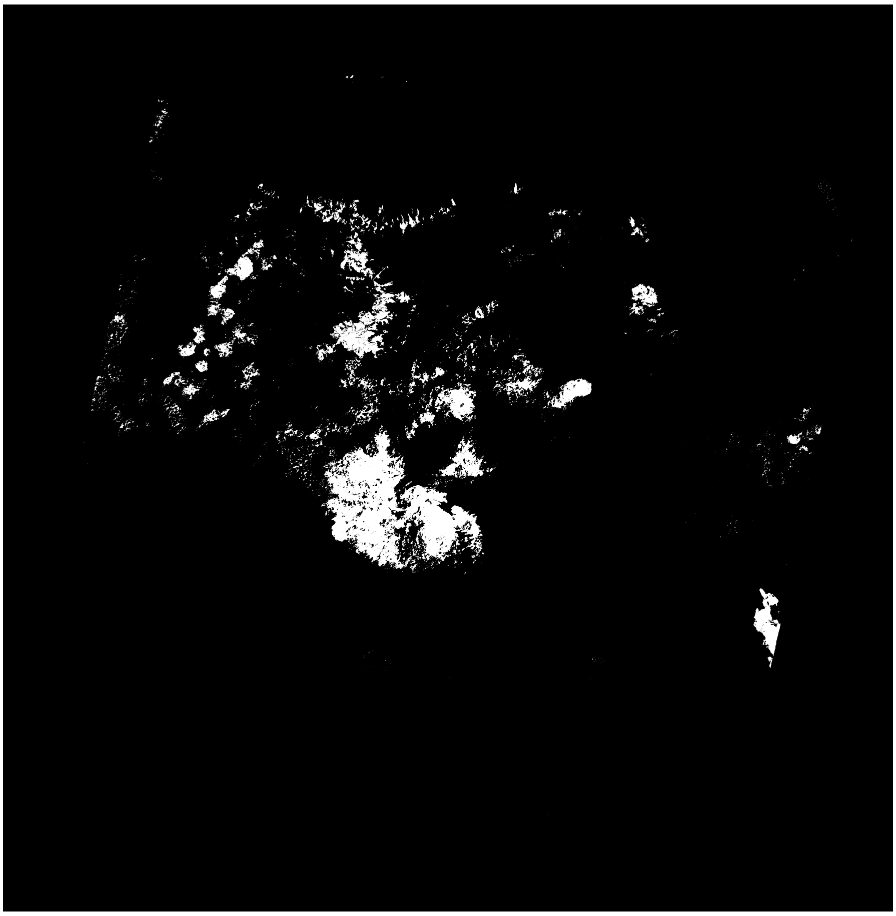

Figure 3.2: NDFSI mask sample

Here from analysing the properties of all the indices we came to a conclusion that NDFSI index will be most suited as a mask here, as the area under consideration is a highly forested region.

Chapter 4

Implementation

Data Set::

For developing evaluation methods the spectral data from landsat 8 satellite on multiple dates are downloaded. Each patch of landsat 8 contains 12 different bands. Out of this bands 2,5,6 bands are loaded into QGIS and using raster calculator the ndsi and ndfsi masks are generated and saved into folders. This process is briefly shown in one of the previous images along with one of the generated masks.

segmentation Using U-net::

A unet architecture is designed for the segmentation process. The code is shown in the image below. The generated dataset is used to train the model. The network summary, training results and other training results are shown below.

The dataset is loaded and split into training and testing data. this is then fed into the network for training. The trained model is then used to predict a snow mask. we then calculate percentage of snow in the image.

```
def unet(input_size=(256, 256, 3)):
    inputs = tf.keras.Input(input_size)

    # Encoder
    conv1 = layers.Conv2D(64, 3, activation='relu', padding='same')(inputs)
    conv1 = layers.Conv2D(64, 3, activation='relu', padding='same')(conv1)
    pool1 = layers.MaxPooling2D(pool_size=(2, 2))(conv1)

    conv2 = layers.Conv2D(128, 3, activation='relu', padding='same')(pool1)
    conv2 = layers.Conv2D(128, 3, activation='relu', padding='same')(conv2)
    pool2 = layers.MaxPooling2D(pool_size=(2, 2))(conv2)

    conv3 = layers.Conv2D(256, 3, activation='relu', padding='same')(pool2)
    conv3 = layers.Conv2D(256, 3, activation='relu', padding='same')(conv3)
    pool3 = layers.MaxPooling2D(pool_size=(2, 2))(conv3)

    conv4 = layers.Conv2D(512, 3, activation='relu', padding='same')(pool3)
    conv4 = layers.Conv2D(512, 3, activation='relu', padding='same')(conv4)
    pool4 = layers.MaxPooling2D(pool_size=(2, 2))(conv4)

    conv5 = layers.Conv2D(1024, 3, activation='relu', padding='same')(pool4)
    conv5 = layers.Conv2D(1024, 3, activation='relu', padding='same')(conv5)

    # Decoder
    up6 = layers.concatenate([layers.Conv2DTranspose(512, (2, 2), strides=(2, 2), padding='same')(conv5), conv4], axis=3)
    conv6 = layers.Conv2D(512, 3, activation='relu', padding='same')(up6)
    conv6 = layers.Conv2D(512, 3, activation='relu', padding='same')(conv6)

    up7 = layers.concatenate([layers.Conv2DTranspose(256, (2, 2), strides=(2, 2), padding='same')(conv6), conv3], axis=3)
    conv7 = layers.Conv2D(256, 3, activation='relu', padding='same')(up7)
    conv7 = layers.Conv2D(256, 3, activation='relu', padding='same')(conv7)

    up8 = layers.concatenate([layers.Conv2DTranspose(128, (2, 2), strides=(2, 2), padding='same')(conv7), conv2], axis=3)
    conv8 = layers.Conv2D(128, 3, activation='relu', padding='same')(up8)
    conv8 = layers.Conv2D(128, 3, activation='relu', padding='same')(conv8)

    up9 = layers.concatenate([layers.Conv2DTranspose(64, (2, 2), strides=(2, 2), padding='same')(conv8), conv1], axis=3)
    conv9 = layers.Conv2D(64, 3, activation='relu', padding='same')(up9)
    conv9 = layers.Conv2D(64, 3, activation='relu', padding='same')(conv9)

    outputs = layers.Conv2D(1, (1, 1), activation='sigmoid')(conv9)

    model = models.Model(inputs=[inputs], outputs=[outputs])
```

Figure 4.1: U-NET code

```
Model: "model"
__________________________________________________________________________________________________
 Layer (type)                Output Shape                 Param #   Connected to
==================================================================================================
 input_1 (InputLayer)        [(None, 256, 256, 3)]        0         []

 conv2d (Conv2D)             (None, 256, 256, 64)         1792      ['input_1[0][0]']

 conv2d_1 (Conv2D)           (None, 256, 256, 64)         36928     ['conv2d[0][0]']

 max_pooling2d (MaxPooling2  (None, 128, 128, 64)         0         ['conv2d_1[0][0]']
 D)

 conv2d_2 (Conv2D)           (None, 128, 128, 128)        73856     ['max_pooling2d[0][0]']

 conv2d_3 (Conv2D)           (None, 128, 128, 128)        147584    ['conv2d_2[0][0]']

 max_pooling2d_1 (MaxPoolin  (None, 64, 64, 128)          0         ['conv2d_3[0][0]']
 g2D)

 conv2d_4 (Conv2D)           (None, 64, 64, 256)          295168    ['max_pooling2d_1[0][0]']

 conv2d_5 (Conv2D)           (None, 64, 64, 256)          590080    ['conv2d_4[0][0]']

 max_pooling2d_2 (MaxPoolin  (None, 32, 32, 256)          0         ['conv2d_5[0][0]']
...
Total params: 31031745 (118.38 MB)
Trainable params: 31031745 (118.38 MB)
Non-trainable params: 0 (0.00 Byte)
__________________________________________________________________________________________________
```

Figure 4.2: U-NET Summary

```
Epoch 1/10
2/2 [==============================] - 12s 3s/step - loss: 0.6839 - accuracy: 0.5842 - val_loss: 0.5296 - val_accuracy: 0.9684
Epoch 2/10
2/2 [==============================] - 6s 3s/step - loss: 0.4794 - accuracy: 0.9716 - val_loss: 0.3454 - val_accuracy: 0.9684
Epoch 3/10
2/2 [==============================] - 6s 3s/step - loss: 0.3536 - accuracy: 0.9716 - val_loss: 0.2688 - val_accuracy: 0.9684
Epoch 4/10
2/2 [==============================] - 6s 3s/step - loss: 0.2738 - accuracy: 0.9716 - val_loss: 0.2471 - val_accuracy: 0.9684
Epoch 5/10
2/2 [==============================] - 6s 3s/step - loss: 0.2094 - accuracy: 0.9716 - val_loss: 0.1974 - val_accuracy: 0.9684
Epoch 6/10
2/2 [==============================] - 6s 3s/step - loss: 0.1710 - accuracy: 0.9716 - val_loss: 0.1561 - val_accuracy: 0.9684
Epoch 7/10
2/2 [==============================] - 6s 3s/step - loss: 0.1359 - accuracy: 0.9716 - val_loss: 0.1272 - val_accuracy: 0.9684
Epoch 8/10
2/2 [==============================] - 6s 3s/step - loss: 0.1136 - accuracy: 0.9716 - val_loss: 0.1184 - val_accuracy: 0.9684
Epoch 9/10
2/2 [==============================] - 6s 3s/step - loss: 0.1059 - accuracy: 0.9716 - val_loss: 0.1161 - val_accuracy: 0.9684
Epoch 10/10
2/2 [==============================] - 6s 3s/step - loss: 0.1050 - accuracy: 0.9716 - val_loss: 0.1167 - val_accuracy: 0.9684
1/1 [==============================] - 0s 360ms/step - loss: 0.1167 - accuracy: 0.9684
Validation Loss: 0.11671236902475357
Validation Accuracy: 0.9683990478515625
```

Figure 4.3: Model training

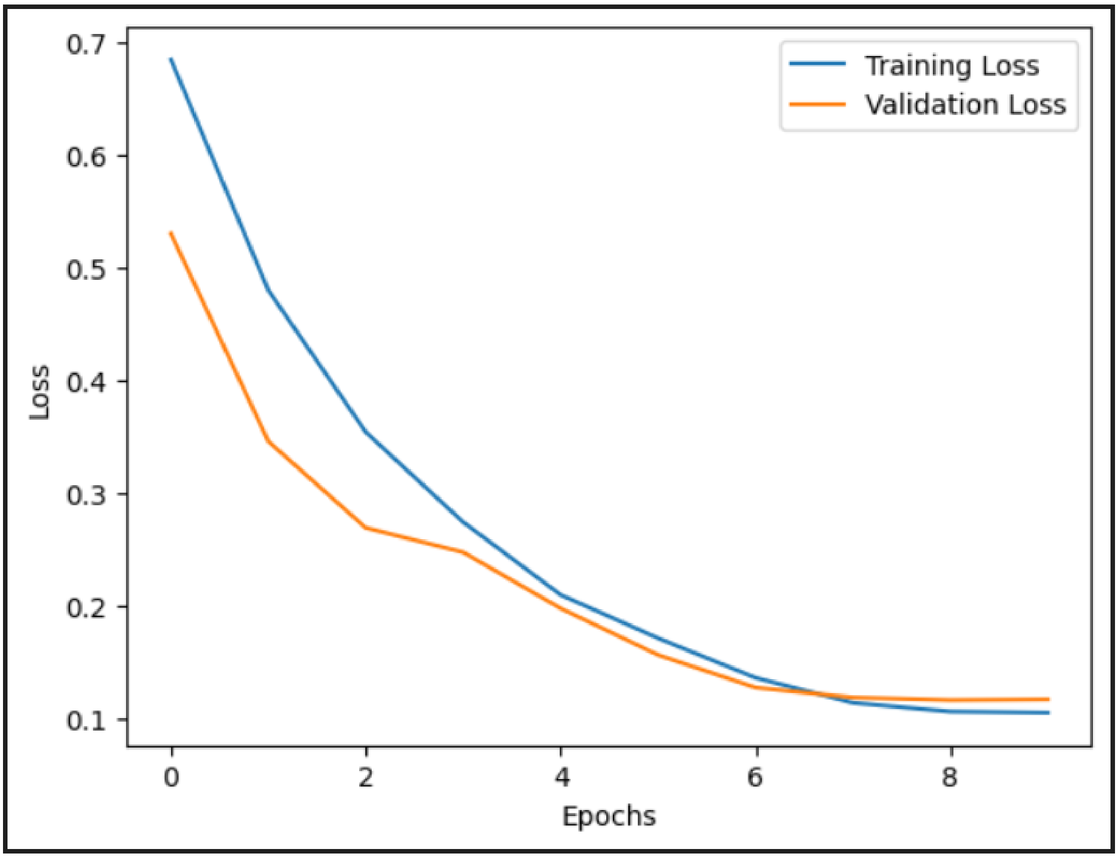

Figure 4.4: Training and validation losses

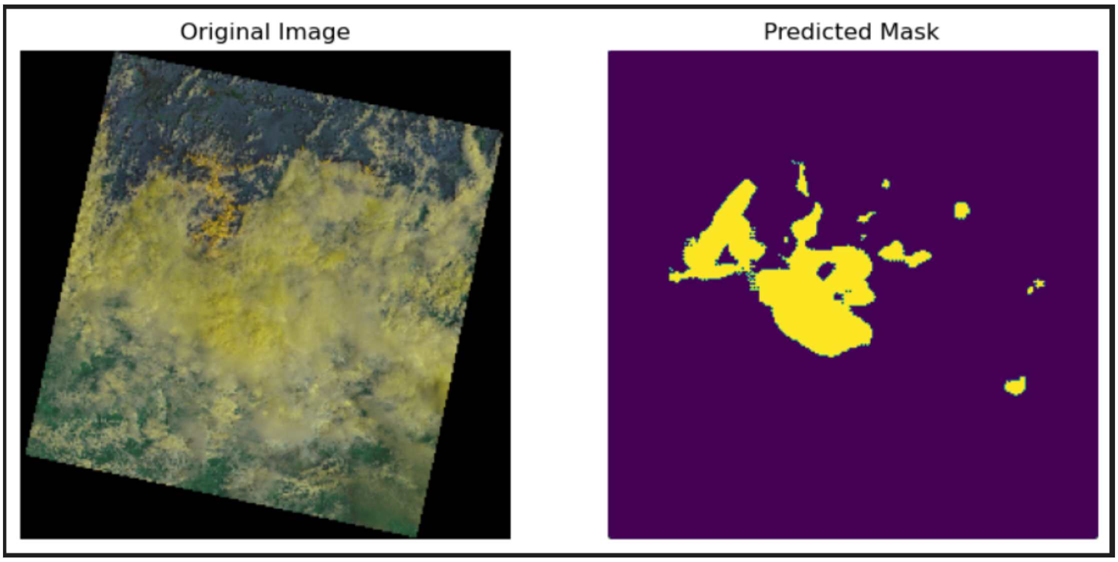

Figure 4.5: Prediction of mask

Chapter 5

Results

Segmented image::

The segmented image is generated by the model. np.sum is applied to image as it is binary and then divided with 255 to get number of snow pixels in the image. It is then divided with total number of pixels to obtain fractional snow cover value. it is then multiplied by total area of the patch to get an estimated snow cover in the given patch of data.

$$\text{total snow cover} = \text{Fractional snow cover percentage} \times \text{total area} \tag{5.1}$$

We get 4371 pixels. When divided with 256*256 which is the size of the image. we get 6.67% . The total are covered by the patch is 185*180 km which is equal to 33300 sq.km. on multiplying the percentage with the total km we get 2221.09 sq km.

Snow cover estimate::
The patch chosen was of August 20th 2023. above Sikkim and nearby territories of China, Nepal and Bhutan. The Snow cover in these region during this date is estimated to be 6.67% or 2221.09 sq km.

Chapter 6

Conclusions

Fractional snow cover estimation above the Indian Himalayas is done for date August 20th 2023.

6.1 Bibiliography

1. Y. Wang, L. Gu, X. Li and X. Fan, "An Effective Fractional Snow Cover Estimation Method Using Deep Feature Snow Index," in IEEE Geoscience and Remote Sensing Letters, vol. 21, pp. 1-5, 2024, Art no. 2000605, doi: 10.1109/LGRS.2024.3365792 keywords: Power capacitors;Snow;Estimation;Feature extraction;Forestry;Geoscience and remote sensing;Indexes;Deep learning (DL);fractional snow cover (FSC) estimation;linear regression;snow index,

2. E. H. Bair, T. Stillinger and J. Dozier, "Snow Property Inversion From Remote Sensing (SPIReS): A Generalized Multispectral Unmixing Approach With Examples From MODIS and Landsat 8 OLI," in IEEE Transactions on Geoscience and Remote Sensing, vol. 59, no. 9, pp. 7270-7284, Sept. 2021, doi: 10.1109/TGRS.2020.3040328. keywords: Snow;MODIS;Remote sensing;Earth;Cloud computing;Artificial satellites;Grain size;Albedo;Landsat;moderate-resolution imaging spectroradiometer (MODIS);multispectral imaging;snow;spectral mixing,

3.G. Wang, L. Jiang, J. Shi and X. Su, "A Universal Ratio Snow Index for Fractional Snow Cover Estimation," in IEEE Geoscience and Remote Sensing Letters, vol. 18, no. 4, pp. 721-725, April 2021, doi: 10.1109/LGRS.2020.2982053. keywords: Snow;Power capacitors;Remote sensing;Earth;Artificial satellites;Indexes;MODIS;Fractio snow cover (FSC);moderate resolution imaging spectroradiometer (MODIS);normalized difference snow index (NDSI);universal ratio snow index (URSI),

4.W. Luan, X. Zhang, P. Xiao, H. Wang and S. Chen, "Binary and Fractional MODIS Snow Cover Mapping Boosted by Machine Learning and Big Landsat Data," in IEEE Transactions on Geoscience and Remote Sensing, vol. 60, pp. 1-14, 2022, Art no. 4305714, doi: 10.1109/TGRS.2022.3198508. keywords: Snow;Remote sensing;MODIS;Earth;Artificial satellites;Power capacitors;Training;Fractional snow cover;Landsat;machine learning;Moderate Resolution Imaging Spectroradiometer (MODIS);random forest (RF);snow cover mapping;Tibetan Plateau,