CONVOLUTIONAL NEURAL NETWORKS TO DETECT CLOUDS AND SNOW IN OPTICAL IMAGES

DEBVRAT VARSHNEY
March, 2019

SUPERVISORS:
Mr. P. K. Gupta
Dr. C. Persello
Dr. B. R. Nikam

CONVOLUTIONAL NEURAL NETWORKS TO DETECT CLOUDS AND SNOW IN OPTICAL IMAGES

DEBVRAT VARSHNEY
Enschede, The Netherlands, March, 2019

ABSTRACT

Studying snow cover is integral in monitoring a country's hydrological resources and assessing climate change. Satellite remote sensing can support this as it covers a large spatial extent and reduces the need for human excursions. Optical remote sensing is specifically advantageous as it measures the albedo and snow surface properties, giving an accurate assessment of the area. However, the visible bands of such images show high reflectance values for both clouds and snow, often leading to misclassification and unreliable results. Shortwave infrared (SWIR) band, on the other hand, is extremely reflective for clouds, compared to snow. But due to their large wavelengths, SWIR sensors are generally not available in high spatial resolutions.

In order to use SWIR to discriminate between clouds and snow in high resolution Visible-Near Infrared (VNIR) images, our study proposes the use of convolutional neural networks (CNNs). CNNs provide an efficient way of deep feature extraction using contextual learning. We use the fully convolutional approach to achieve pixel-wise classification through semantic segmentation. Moreover, we apply a novel way of resampling the SWIR within the CNN architecture and fusing it with the VNIR bands. This fusion based convolutional strategy gave an average snow-and-cloud F1 score of 0.95 compared to a score of 0.85 by a non-fusion based network. We performed all our experiments on the multi-resolution data available from Resourcesat-2 satellite, of the Indian Remote Sensing Program, using visually labeled reference pixels. We also compared the classification output from a subsidiary model with a pre-built cloud mask tool of Resourcesat-2. We found that our model achieved an F1 score of 0.91 for clouds compared to 0.65 by the pre-built tool. The proposed model thus showed an advantage in detecting clouds in high resolution optical images, captured over snow covered regions. This highlights the possible use of such methods for other multi-sensor fusion problems, in the future.

Keywords:
Fully Convolutional Networks, Multi-resolution fusion, Deep learning, LISS-IV, Cloud detection, Snow

ABBREVIATIONS

AWiFS	Advanced Wide Field Sensor
ANN	Artificial Neural Networks
CFB	Concatenated Feature Block
CSMG	Cloud and Shadow Mask Generator
CNN	Convolutional Neural Networks
FCC	False Colour Composite
FCN	Fully Convolutional Network
LISS	Linear Imaging Self Scanner
NRSC	National Remote Sensing Centre, Hyderabad
OA	Overall Accuracy
PA	Producer Accuracy
RS-2	Resourcesat-2
SCA	Snow Cover Area
SFB	SWIR Feature Block
SWIR	Shortwave-Infrared
UA	User Accuracy
VNIR	Visible and Near-Infrared
VFB	VNIR Feature Block

ACKNOWLEDGEMENTS

I'm grateful to a lot of people who have directly, or indirectly contributed to this research work. I'd first like to thank Prasun Sir, for floating this topic and providing an opportunity to enter and explore the wonderful world of Machine Learning. He has been extremely approachable and kind with his advice. I'd also like to thank Dr. Persello for seeing the novelty in this work and trusting my abilities. He has always been patient and generous with his ideas. Furthermore, having the faith and encouragement from Dr. Nikam is what kept this project alive.

I'm grateful to National Remote Sensing Centre, Hyderabad, for providing data captured by Resourcesat-2. I'd also like to thank the faculty and staff of both IIRS and ITC, especially Mr. Avdhesh and Mr. Ajay from the CMA department, who have been extremely prompt and unperturbed in providing technical assistance. A big shout out to the ITC Hotel for organizing a peaceful stay in Enschede. More importantly, this whole academic joyride would not have been possible without the purview and guidance of Dr. Sameer Saran and Dr. Tolpekin; both of whom have been instrumental in coordinating this course smoothly. Also, it has been quite a privilege to have met Prof. Stein, as his inspiring and joyful personality has left quite a mark on me.

A special thanks to Khairiya Mudrik, for literally saving me in Module 13, and to Sheetabh Gaurav for all the discussion sessions. I'm not sure if I would have been able to grasp the CNN fundamentals without these two.

I cannot express enough gratitude to have such a loving, caring and a disciplined family like mine. Their faith and push is why I joined this course in the first place, and it has become one of the few decisions of my life, that I can actually be proud of.

Lastly, I'll forever be indebted to my classmates at IIRS, for all the good times that we had.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

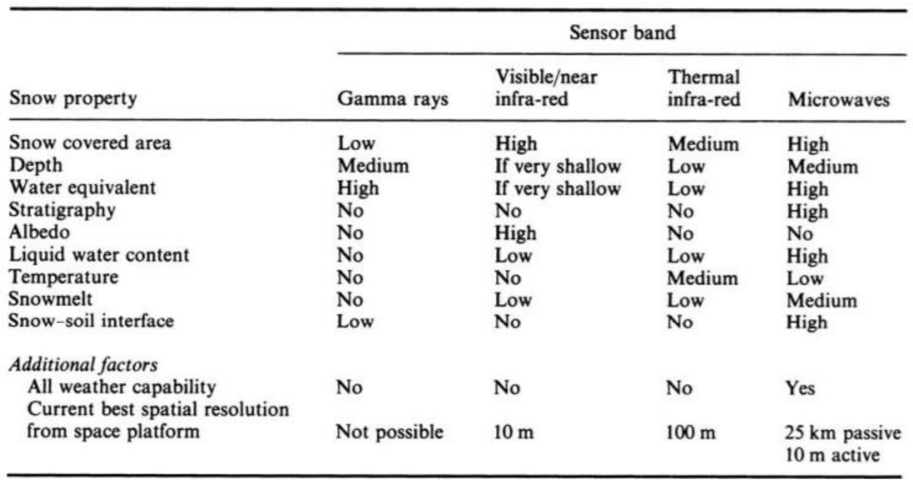
1.1. Snow and Cloud Similarity

Snow is an important feature of our environment. It helps in balancing the heat flow between the Earth surface and atmosphere. Its presence in a basin also affects surface moisture, thereby contributing to water runoff (Maurer, Rhoads, Dubayah, & Lettenmaier, 2003). It has been found that analyzing snow cover area (SCA) plays an extensive role in managing water resources; while studying the snowmelt can help us assess water requirements for agricultural & other societal needs (Tekeli, Sönmez, & Erdi, 2016; National Snow and Ice Data Center [NSIDC], 2017). Apart from hydrological aspects, detailed snow cover maps are also utilized in weather forecasting and military operations (Miller, Lee, & Fennimore, 2005). Thus, studying the spatial extent of snow has wide applications.

Such spatial understanding has historically been made through snow surveys, which are mainly point measurements, and thus do not provide good estimates of the areal cover. Furthermore, as snow is present in a mountainous (rough/undulating) terrain, the measurement excursions can easily translate into becoming quite a labor intensive, expensive and hazardous activity (Man, Guo, Liu, & Dong, 2014). This is where Remote Sensing comes into the picture. The large extent, and high spatial resolution, captured by a remotely sensed imagery can help us make accurate predictions of characteristics like snow cover area and snow water equivalent.

Optical remote sensing of snow brings its own challenges. While studies like Rango (1993) show that the Visible/Near-Infrared (VNIR) bands are quite helpful in capturing the albedo and areal extent of snow (Table 1), Nikam et al. (2017) mention that as clouds have similar reflectance values in the VNIR range, they become quite a hindrance while mapping snow in this spectral range. Miller et al. (2005) have further noted that the shortwave infrared (SWIR) band (1.6 to 2.2 μm) is a better alternative to discriminate between clouds and snow. The lower reflectance of snow, as compared to clouds, in the SWIR range can help in SCA estimation. The spectral difference between snow and clouds in the SWIR region is further portrayed in Figures 1 and 2.

Table 1: Sensor responses to various snow properties (Rango, 1993)



	Sensor band			
Snow property	Gamma rays	Visible/near infra-red	Thermal infra-red	Microwaves
Snow covered area	Low	High	Medium	High
Depth	Medium	If very shallow	Low	Medium
Water equivalent	High	If very shallow	Low	High
Stratigraphy	No	No	No	High
Albedo	No	High	No	No
Liquid water content	No	Low	Low	High
Temperature	No	No	Medium	Low
Snowmelt	No	Low	Low	Medium
Snow–soil interface	Low	No	No	High
Additional factors				
All weather capability	No	No	No	Yes
Current best spatial resolution from space platform	Not possible	10 m	100 m	25 km passive 10 m active

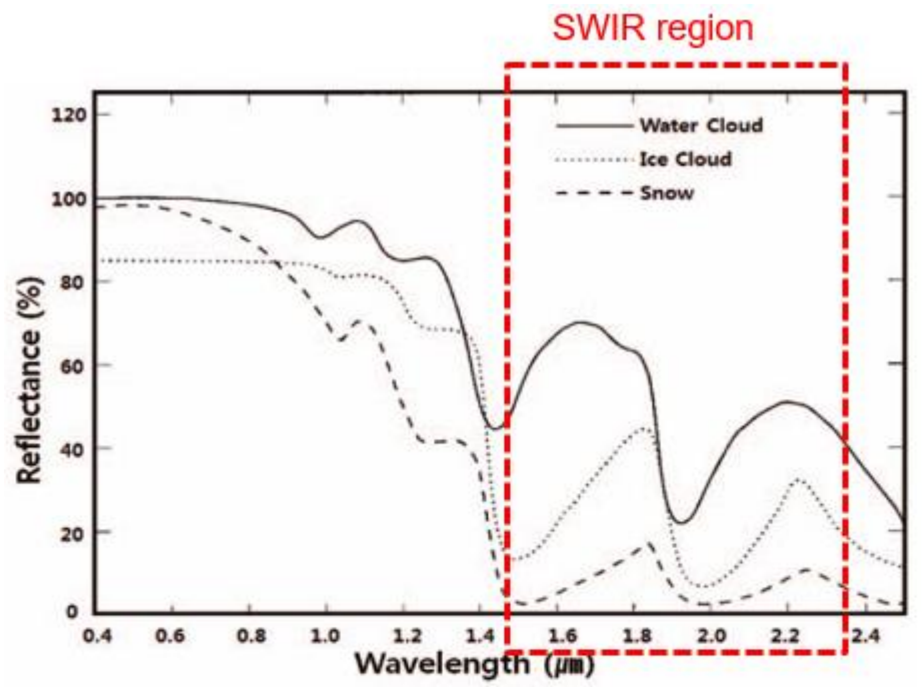


Figure 1: Reflectance curves of water cloud, ice cloud and snow. While both snow and clouds have similar reflectance values in the lower wavelength regions, the gap in their reflectance values increases as we move towards higher wavelengths. The SWIR region (marked in red) is where snow exhibits near-zero reflectance values, whereas clouds exhibit high reflectance values, in sharp contrast. (Gao, Han, Tsay, & Larsen, 1998)

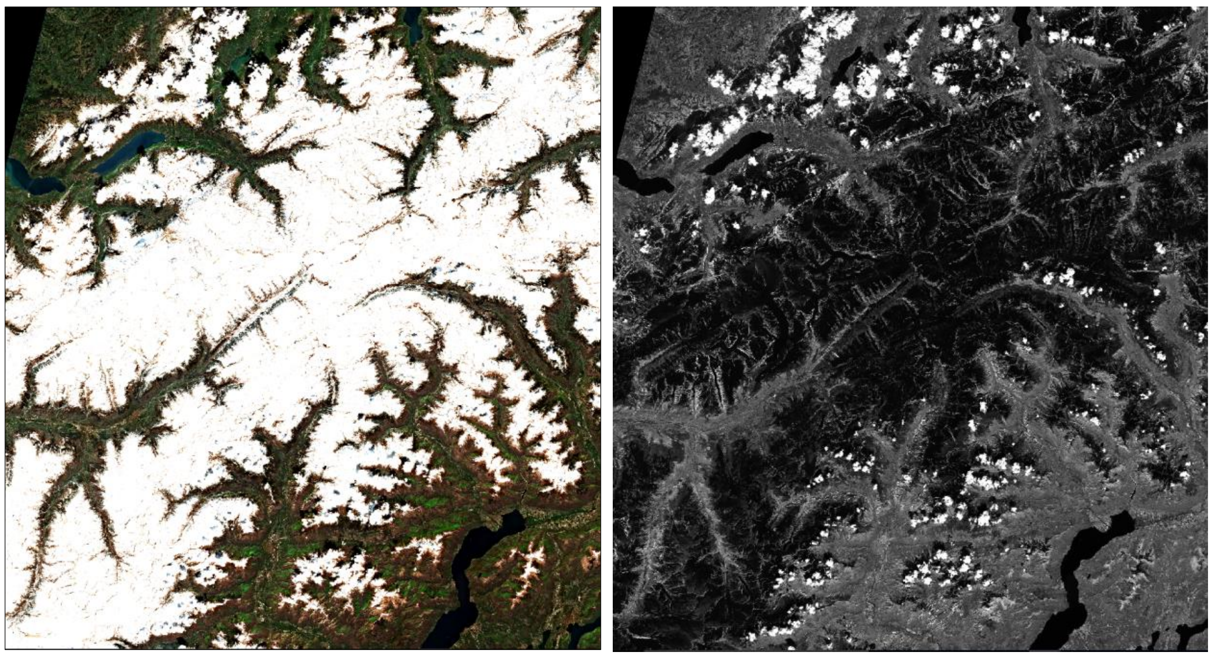


Figure 2: A True Colour Image (on the left) of the Swiss Alps, taken by Sentinel-2A. It is very hard to detect clouds in such an image. In the image on the right, the SWIR channel of the same satellite helps in highlighting clouds with bright pixels

The characteristics of SWIR have been widely used by satellite sensors like **L**inear **I**maging **S**elf **S**canner (LISS) – III and **A**dvanced **Wi**de **F**ield **S**ensor (AWiFS) for snow mapping purposes (Birajdar, Venkataraman, & Samant, 2016; Kulkarni, Singh, Mathur, & Mishra, 2006; Srinivasulu & Kulkarni, 2004). But, their spatial resolution is low (23.5m and 56m of LISS-III and AWiFS, respectively). Also, the current cloud masking software available for LISS-III is rudimentary in nature (National Remote Sensing Centre [NRSC], 2017). Moreover, satellites such as Landsat and Sentinel provide cloud masks in their Level-2 products, which is missing for any Indian Remote Sensing product.

Bühler, Meier, & Ginzler (2015) report that cloud-free Near-Infrared (NIR) images, of high spatial resolution, have potential in measuring the small scale spatial variability of snow properties. Thus, in order to achieve an effective cloud mask at a higher spatial resolution, we can incorporate the characteristics of the LISS-IV sensor, present on the same satellite as the two sensors mentioned earlier. All the three sensors (AWiFS, LISS-III and LISS-IV) are available on Resourcesat-2 and work in the same VNIR range. LISS-IV has the highest spatial resolution (5.8m), whereas AWiFS and LISS-III carry an additional SWIR band. Moreover, these sensors are nadir looking and hence capture a geographic area at the same time.

Our study aims to combine the characteristics of LISS-III and LISS-IV, in order to obtain a high resolution robust cloud mask over snow regions for Resourcesat-2 satellite. In order to implement this, we propose to utilize neural networks for image classification, but first, in the next section, we take a glimpse into traditional techniques of cloud detection.

1.2. Feature Detection

Detecting clouds in optical satellite images has traditionally been carried out through thresholding techniques, such as those used by Lyapustin, Wang, & Frey (2008) and Z. Zhu & Woodcock (2012). These techniques majorly involve arithmetic computations over a variety of bands, followed by taking thresholds like calculating Normalized Difference Cloud Index (NDCI), or a Normalized Difference Snow Index (NDSI) (Tang et al., 2010). Although pixel-wise thresholding can be fast, and computationally less intensive, these techniques majorly remain ineffective when detecting features in a spatio-contextual sense (Guirado, Tabik, Alcaraz-Segura, Cabello, & Herrera, 2017).

Clouds can be of various types, depending upon their thickness (such as thick clouds, cirrus clouds), and all these types can have varying spectral reflectance values. Also, their texture and shape can vary depending on the time of the day, and wind speed. Moreover, the spectral signatures of clouds can be easily confused with other highly reflective land surfaces such as concrete, snow or ice (X. Zhu & Helmer, 2018). Thus clouds form complex feature sets which can be extremely tough to detect using primitive thresholding techniques.

There has been growing interest in using Artificial Neural Networks, and specifically Convolutional Neural Networks, which can perform efficient feature detection. The higher computational complexity that they involve is often ignored to achieve accurate results over large datasets. This has led to exponential work on applying neural networks for the purpose of cloud detection, such as those by Mateo-García, Gómez-Chova, & Camps-Valls (2017). We further discuss about such Artificial Neural Networks in the next section.

1.3. Artificial Neural Networks

Artificial Neural Networks (ANNs) are computing frameworks which mimic the functioning of a biological brain. These frameworks can *learn* to perform tasks similar to how a human or an animal performs. The framework tries to map a set of inputs to a given set of outputs and tries to come up with predictions which are as close to the target set as possible. Similar to any biological neural network, this learning, or training, takes place iteratively, and the network tries to come closer to the desired output with each iteration.

For creating image segments, a network is fed with an image and a corresponding set of pre-labeled pixels. Once the network learns attributes such as texture, tone and spatial correlation of the labeled pixels, it can classify the rest of the unlabelled pixels with this information. Such a trained network can then be used on an entirely new image, in order to classify it.

Apart from object detection and image classification, these frameworks also help in other complex tasks such as speech recognition, stock predictions etc. With silicon chips becoming faster and cheaper, these frameworks have significantly helped in extracting information from vast amounts of datasets, especially those being produced by remote sensing products nowadays.

The basic functioning of an artificial neural network, and its iterative learning process, is explained in brief in the following subsections.

1.3.1. The Perceptron

The most fundamental unit of an artificial neural network is the perceptron. It is also referred to as a node, or a neuron. It takes a weighted sum of inputs and applies a non-linear activation function to it. The weighted sum can also have an additional constant, a biased term, added to it. This is implemented by introducing an extra input having a constant value of 1 where the weight on this input is called a bias.

The green oval in Figure 3 highlights the perceptron. It is made up of two functions denoted by circles inside.

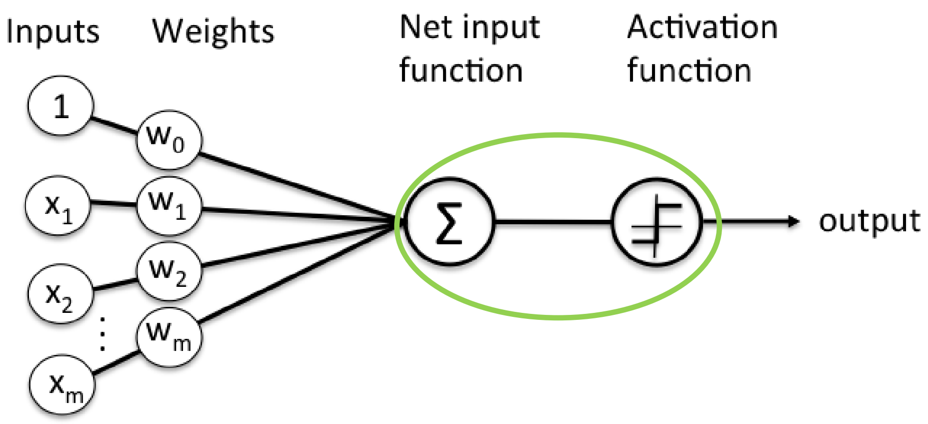


Figure 3: A perceptron, shown in green, taking a weighted sum of inputs $\{x_1, x_2 \dots x_m\}$ along with w_0 as bias and applying an activation function to the entire sum (Raschka, 2015)

The output of this perceptron is given in Equation 1, where 'f' is a non-linear activation function, and the summation is performed over 'm' inputs. The activation function is generally of the likes of a sigmoid, or a hyperbolic tangent function.

$$y = f\left(\sum_{i=1}^{m} w_i x_i + w_0\right) \tag{1}$$

1.3.2. Multi Layer Perceptron

Any number of such perceptrons can be used with a varied set of weights. Figure 4 shows a simple neural network with two perceptrons sharing three inputs.

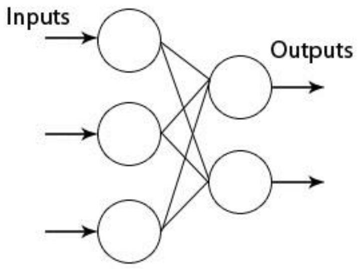


Figure 4: A simple network having two perceptrons. Every input-output connection will have a unique weight associated with it ("Perceptron," 2014)

These perceptrons can be stacked into multiple layers to build a denser, and a computationally more intensive, network. Such ANNs are referred to as Multi Layer Perceptrons (MLP). Figure 5 shows a three-layered neural network. The layer between the input and output layers is referred to as the hidden layer.

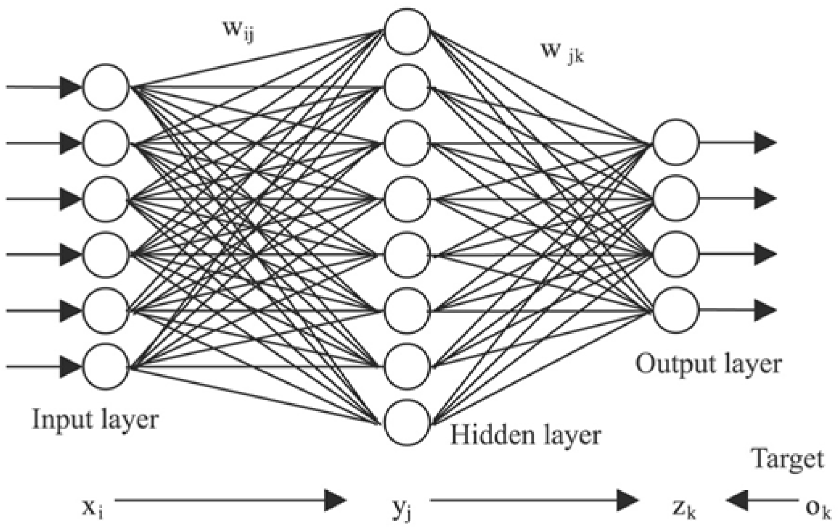


Figure 5: An Artificial Neural Network containing input nodes (x_i), mapped to output nodes (z_k) via intermediate hidden nodes (y_j). Every connection between two stages of nodes (a neural connection) has a certain 'weight' (w) associated with it, which is initialized randomly at first. After each forward pass of the input data, the predicted outputs z_k are compared with actual target outputs o_k. The error between this predicted set and the target set of outputs is passed back to the network so that the weights can be modified to decrease the error. (Templeton, 2015)

In Figure 5, weight w_{ij} acts upon the ith input and the jth node of the hidden layer. Similarly, weight w_{jk} acts upon the jth node of the hidden layer and the kth output. Assuming an unbiased network, and the same activation function at each layer, the hidden node will produce y_j and the output node will produce z_k as given in Equations 2 and 3 respectively.

$$y_j = f\left(\sum_i w_{ij} \cdot x_i\right) \tag{2}$$

$$z_k = f\left(\sum_j w_{jk} \cdot y_j\right) \tag{3}$$

1.3.3. Backpropagation and Gradient Descent

The primary purpose of a neural network is to find an optimum combination of weights, which can translate a given set of input data to a set of output which is as close to the desired (target) set of outputs as possible. Hence, for a given set of weights, the performance of such a network can be judged by the total error it produces between the predicted and the desired sets of outputs (Rumelhart, Hinton, & Williams, 1986). This total error is defined as in Equation 4. Here m is the number of samples with which the network is trained.

$$E = \frac{1}{2}\sum_{k=1}^{m}(z_k - o_k)^2 \tag{4}$$

In order to reduce this error, we need to find a set of weights which can minimize it. As z_k is an outcome of every weight w_{ij} and w_{jk}, we can use partial derivative of the error with respect to every weight. We can update the weight with the help of a simple gradient descent (Rumelhart et al., 1986) given in Equation 5. Here 'η' is known as the learning rate of the network.

$$\Delta w = -\eta \frac{\partial E}{\partial w} \tag{5}$$

This automatic weight modification (the training) is carried out till the error reaches a minimum. Literature suggests that any task performed by such neural networks, especially those related to semantic segmentation, can lead to high levels of accuracy; like the one used by Shelhamer, Long, & Darrell (2017)

1.4. Research Prospect

This research aims to use the VNIR information of a high resolution sensor along with the SWIR information of a medium resolution sensor, in order to segregate clouds from snow effectively. The novelty of this work is to build a robust neural network architecture especially designed for this purpose. The resultant cloud mask should be effective enough on a variety of snow covered regions. Moreover, such a cloud mask should be applicable for high resolution Indian Remote Sensing product, which has been lacking till now. The objectives to be achieved, and the corresponding questions to be explored and answered, are as follows.

1.4.1. Research Objectives

1. To utilise the SWIR information from LISS-III and fuse it with the corresponding LISS-IV image using a novel neural network architecture that can generate a high resolution classified map of clouds and snow.

2. To analyze if the requirement of the additional SWIR band was beneficial or not

3. To compare the performance of the proposed architecture with more prominent cloud mask solutions involving traditional techniques

4. To compute the SCA in a given image of a snow region, and calculate the percentage of clouds present.

1.4.2. Research Questions

The questions to be addressed with respect to the above objectives are as follows.

Objective 1:

a) What is the classification accuracy obtained by the proposed network?
b) How can the network be improved to increase the accuracy?

Objective 2:

a) Was there any advantage in introducing and fusing a Shortwave Infrared band?

Objective 3:

b) Does the proposed network perform better than traditional techniques?
c) Are the extensive computations involved in the proposed network justified?

1.4.3. Thesis Structure

This chapter was to set a background of the research prospect, highlight the motivation, the objectives and give a glimpse of the methodology that would be adopted. **Chapter 2** gives a background on a special type of ANN, and explains why it would be beneficial for our problem statement. This chapter also gives an overview of some of the prominent cloud masking utilities available as of now. Furthermore, **Chapter 3** explains the characteristics inherent in our network, while **Chapter 4** guides through the process of building an optimum architecture. The dataset used in this study is explained in **Chapter 5**, and **Chapter 6** is where we observe the network's performance and make a comparative analysis. Finally, we conclude the thesis in **Chapter 7**, highlighting the scope of further research.

2. BACKGROUND

This chapter discusses some of the current state-of-the-art cloud mask utilities available with the remote sensing community, and then subsequently gives an idea about Convolutional Neural Networks, which form the backbone of our cloud detection algorithm.

2.1. Cloud Mask Utilities

The inspiration behind this work was the Fmask algorithm created by Z. Zhu & Woodcock (2012). The algorithm builds over the traditional ACCA algorithm (Irish, Barker, Goward, & Arvidson, 2006) to detect clouds, cloud shadows along with semi-transparent clouds and their shadows on Landsat imagery. This algorithm was found to be very effective on a large set of freely available Landsat images and thus was an asset to the remote sensing community. Usability of this algorithm led to its further improvement and application on Sentinel-2 data as well (Z. Zhu, Wang, & Woodcock, 2015). The algorithm detects clouds by a series of spectral tests to generate a cloud probability mask, while it uses thresholds on NDSI and Brightness Temperature to create a snow layer as well.

The algorithm detects cloud shadows by incorporating a couple of geometry based techniques, which can match a shadow region to that of the nearest cloud object. To calculate this, the algorithm heavily depends upon the satellite's metadata, apart from the actual images. The metadata carries information about the sensor's view angle, solar zenith angle and solar azimuth angle, which are required for the aforementioned geometrical techniques. Altogether, Fmask applies a scene based threshold to all the pixels in a neighbourhood, and classifies the pixels into clouds, cloud shadows, and snow; in that priority. It fails to understand the spectral-spatial difference among the different class objects on its own, as it highly depends on the threshold values which have been applied.

Although there exists a cloud mask utility for AWiFS and LISS-III products (National Remote Sensing Centre, 2017), this utility is not built for high resolution LISS-IV images. Moreover, it uses spectral attributes to only detect clouds and cloud shadows from the input image. Like the Fmask software described above, it also requires separate meta files to perform the classification.

Thus, as these utilities are sensor specific, heavily dependent on the associated metadata files, and majorly use spectral thresholds for cloud determination, there lies a scope to build more flexible and robust algorithms which can segregate between snow and cloud features in a more spatio-contextual sense. Convolutional Neural Networks, a variation of Artificial Neural Networks, help in such a feature detection scenario. The next section discusses how such networks operate.

2.2. Convolutional Neural Networks

A Convolutional Neural Network (also called as a CNN or a ConvNet) is a special type of Neural Network where the hidden neurons 'convolve'. Each neuron in the hidden layer of a ConvNet is at a time exposed to only a small region of the previous layer. It performs the weighted sum, followed by activations, for this small region, and then slides, or convolves, onto a neighbouring region using the same set of weights. This procedure is carried out till the entire previous layer (an image, in our case) has been covered by this neuron.

The procedure described above is unlike the one followed in a regular neural network, where the hidden neuron is 'fully connected' to all the neurons of the previous layer. The difference in neural connections between the network structures is highlighted in Figure 6. We can see that the number of weights which need to be learnt, with respect to every hidden neuron, in a CNN is lesser as compared to a regular neural net. This greatly reduces the computations required over an image.

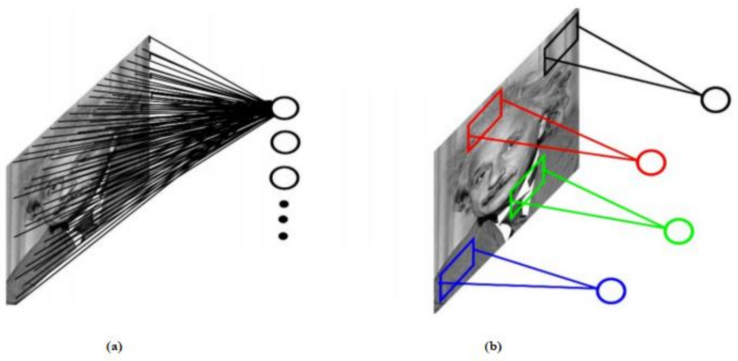


Figure 6: Difference in neural connections between (a) a regular neural net, where the hidden neuron is connected to all the pixels of the previous layer i.e. it is fully connected; and (b) a Convolutional Neural Network where the hidden neuron is at a time connected to only a small region of the previous layer. The different colours depict different positions of that neuron, where the weights used remain the same (Santos, 2019)

The weights of this convolving neuron are also often visualized as a two dimensional matrix, called a kernel, or a filter. This kernel performs a weighted sum at a position, and generates a pixel for the next layer. It then slides to cover the rest of the input image and generates a feature map as the next layer. The local region to which a filter is exposed to is known as the 'receptive field'. This is better portrayed in Figure 7 and 8.

(a)

-1	-2	-1
0	0	0
1	2	1

(b)

0	0	0	0	0	0	0
0	0	0	75	80	80	0
0	0	75	80	80	80	0
0	0	75	80	80	80	0
0	0	70	75	80	80	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0

Figure 7: A 3x3 kernel (a), which will convolve upon a 5x5 input image (b). Different shades of green represent different DN values.

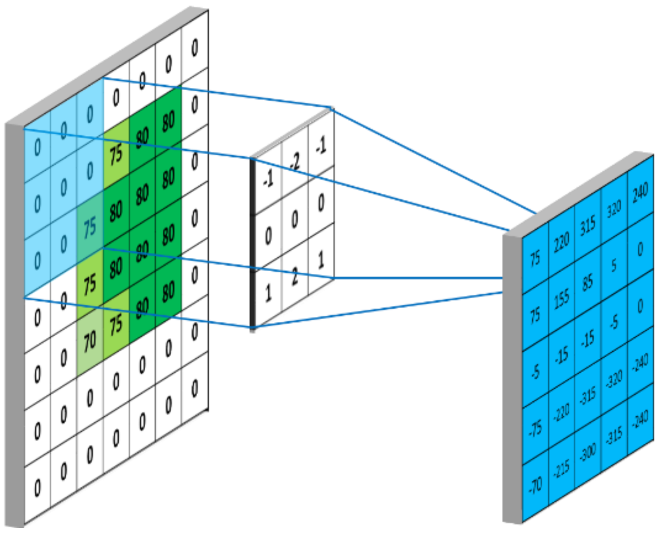


Figure 8: The kernel acting upon the input image. Light blue is the receptive field, whereas the pixel formed after the weighted sum, is in blue ("Convolutional Neural Networks - Basics," 2017)

2.2.1. Layers in a CNN

Apart from the convolutional layer, a CNN is majorly made up of activation layers and pooling layers. Depending on the need, a network can also be 'regularized' with certain layers.

2.2.1.1. Activation Layer

The activation layer applies a non-linear function to the previous layer. The non-linearity is maintained so that the output from this layer is differentiable, and we can obtain a gradient from it. Different types of activation functions are given in Equations 6-10:

1. Sigmoid Function:

$$y = \frac{1}{1 + e^{-x}} \tag{6}$$

2. Hyperbolic Tangent:

$$y = \tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{7}$$

3. Rectified Linear Unit (ReLU):

$$y = \begin{cases} x, & x \geq 0 \\ 0, & x < 0 \end{cases} \tag{8}$$

Using ReLU, compared to other conventional non-linear functions such as the hyperbolic tangent, decreases training time considerably (X. X. Zhu et al., 2017).

4. Leaky ReLU:

$$y = f(x) = \begin{cases} x, & x \geq 0 \\ ax, & x < 0 \end{cases} \tag{9}$$

Here ‘a’ is a positive constant, generally less than 1

5. Softmax:

$$y_i = f(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{k} e^{x_j}} \tag{10}$$

This layer is generally used at the end of the network, for classification. y_i represents the activation for the ith class out of a total of ‘k’ classes. The output channels in the previous layer should be equal to the number of classes required, k.

2.2.1.2. Pooling Layer

This layer is used to select a specific activation from a window. It can be of two types: Max pooling, and average pooling. For a given window size, max pooling will give the maximum activation as the output, whereas in average pooling, the mean of all activations will be given as output (Figure 9).

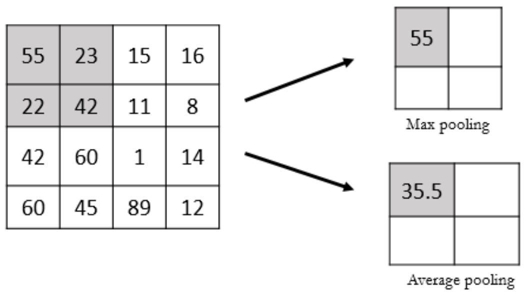


Figure 9: Pooling operation being performed on an image (on the left). A window of 2×2 (shown in grey area) is pooled at a time, to give the output on the right.

2.2.1.3. Regularization Layer

Regularization is a process to prevent overfitting of the training data. Apart from techniques such as weight decay, and early stopping, we can incorporate certain types of layer which inhibit overfitting in neural networks. Such layers are:

1. Dropout:

Dropout was introduced as a regularizer in Srivastava, Hinton, Krizhevsky, Sutskever, & Salakhutdinov, 2014. The authors introduced a model where the units are present with a probability ‘p’. They hypothesize that this model prevents units from getting adapted to each other, and therefore also reduces the effect of noise present in the training data. By using this approach, the test accuracy increases for large datasets.

2. Batch Normalization:

The data over which a neural network needs to be trained (called a batch), is not fed as a whole. It is fed in mini-batches, and the training, through gradient descent, takes place one batch at a time. A Batch Normalization (BN) layer (Ioffe & Szegedy, 2015) first normalizes the output of the previous layer, using the mean and standard deviation of the particular batch, and then linearizes all the outputs with the help of learnable parameters γ and β. This approach reduces the need for dropout.

In Equation 11, x_i is the ith output from the previous layer which has been normalized with respect to the batch mean and batch standard deviation to produce $\hat{x}_i$.

$$y_i = BN_{\gamma,\beta}(x_i) = \gamma \hat{x}_i + \beta \tag{11}$$

2.2.2. CNNs as Feature Detectors

CNNs provide hierarchical feature learning. This means that the initial layers of a network extract basic features such as edges first, and the activations from these layers are pooled to form more complex features (such as objects) in the deeper layers. The fact that the filters focus on only one small region at a time, a convolutional layer helps in understanding the local relationship between pixels. It then auto-correlates this information to form edges, or objects, depending upon the depth of the layer. Comparatively, a fully connected layer in a regular MLP, looks at the entire image, and tries to understand a more global relationship among pixels. Hence CNNs become better feature detectors than regular MLPs (Ben Driss, Soua, Kachouri, & Akil, 2017).

Although research in neural networks has been taking place since the 1980s, one of the pioneering convolutional network was LeNet-5 in 1998. Since 2010, ImageNet Large Scale Visual Recognition Challenge (ILSVRC) started taking place annually, and varied research teams across the academia and industry forayed into developing their own network topologies on the ImageNet database. With AlexNet (Krizhevsky, Sutskever, & Hinton, 2012) winning the 2012 ILSVRC, and matrix multiplications becoming easier with GPUs, there has been extensive reliability on deep, convolutional networks for solving image classification problems. This eventually led to the development of networks such as GoogLeNet, VGGNet and ResNet. Libraries and toolboxes, such as Caffe, TensorFlow and MatConvNet helped implement these networks.

The advantage of this research boom has been that we can use such networks (pre-trained on large datasets), fine tune and deploy them for the problem, and data of our choice. We now see extensive use of such deep learning, computer vision frameworks for varied domains such as medical image processing, and remotely sensed images.

2.2.3. CNNs for Remote Sensing and Cloud Detection

Over the past decades, remotely sensed images have largely become open sourced, contributing to extensive research on automated image classification techniques on such images. From k-means clustering, to object-based analysis, researchers have now started applying neural nets to classify remotely sensed images. Such automated, machine learning methods have made large scale data classification easier, compared to traditional methods of spectral-spatial classification (Maggiori, Tarabalka, Charpiat, & Alliez, 2017).

Along with being applied for SAR data (Cozzolino, Martino, Poggi, & Verdoliva, 2017; Geng, Wang, Fan, & Ma, 2017; Li et al., 2017; Mullissa, Persello, & Tolpekin, 2018), CNNs have shown high classification result for LiDAR (Savchenkov, Davis, & Zhao, 2017; Yang et al., 2017) and especially for High Resolution optical imagery (Bergado, Persello, & Stein, 2018; Wiratama, Lee, Park, & Sim, 2018; Zhang, Niu, Dou, & Xia, 2017). With this as a motivation, we further explored how deep convolutional networks could be applied for cloud detection purposes.

Studies such as Hughes & Hayes (2014) have used exhaustive amounts of training data to create deep neural nets for clouds, cloud shadows, and snow detection, but these networks are not convolutional in nature, and require post-processing for snow-cloud correction. Other studies like Le Goff, Tourneret, Wendt, Ortner, & Spigai, 2017 and X. Zhu & Helmer, 2018 have incorporated deep neural nets for cloud detection. But these are either not developed for snow covered areas, or most of them fail to distinguish snow and cloud pixels efficiently. Mohajerani, Krammer, & Saeedi (2018) have also developed a CNN for cloud detection, but then again use a separate snow/ice removal framework in the pre-processing stage. Zhan et al., (2017) have further developed a deep convolutional network for distinguishing clouds from snow which majorly uses a multiscale prediction strategy, combining low-level feature maps with high level feature maps. But their intermediate feature maps are of varying spatial resolutions, and they interpolate the maps separately before combining. We propose that similar networks can become robust on snow covered regions by incorporating a SWIR channel on the input dataset, and resampling it within the convolutional architecture. Thus, our major study will focus on how we can build an effective convolutional network that works on a VNIR-SWIR composite.

3. NETWORK CHARACTERISTICS

This chapter highlights the traits of the convolutional neural networks adopted for the study.

3.1. Filter Parameters

The network is made up of a variety of hidden layers. Each neuron of a convolutional layer is represented as a stack of filters sliding over an input stack of image channels. This neuron produces a single, unique band as a feature map (Figure 10). Thus, the number of neurons in a convolutional layer also defines the number of channels (feature maps) that the layer will produce as its output. Each filter stack is given by the dimensions $D \times F \times F$, where F is the width of a square filter, and D is the depth of the filter stack, which is the same as the number of input channels to the layer.

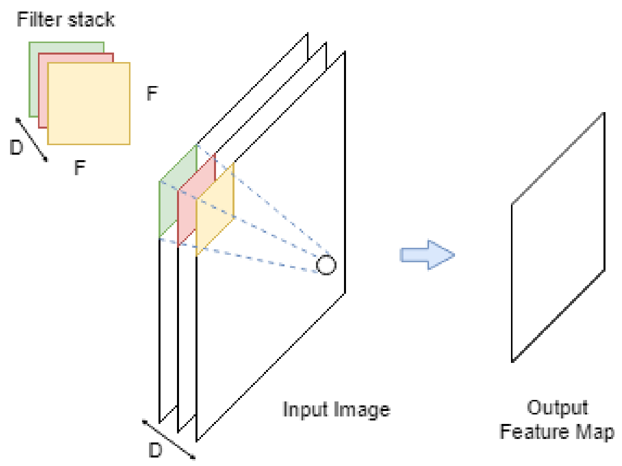


Figure 10: A filter stack corresponding to a convolving neuron, producing a feature map. 'D' filters, each having dimensions of $F \times F$, have unique trainable weights.

To put this together, we represent a convolutional layer as $D \times F \times F \times K$, where K is the number of neurons, representing the number of output channels that it produces. The number of pixels by which a filter slides across a two-dimensional image matrix is called the 'stride', S. Horizontal and vertical stride is kept the same in this study. As the filter stack working on an $F \times F$ receptive field produces just one pixel as an output, the final feature map produced is smaller in dimension, compared to the input feature. In order to keep the dimensions of the input and output feature maps the same, we sometimes add additional zero-valued rows and columns as the outskirts to the image matrix. This phenomenon is called padding. For our study, the number of such rows and columns added are the same, denoted by P.

For an input image band of dimensions $H \times W$, a convolutional layer produces an output feature map of dimensions $H' \times W'$ given in Equation 12.

$$H' = \frac{H - F + 2P}{S} + 1 \tag{12a}$$

$$W' = \frac{W - F + 2P}{S} + 1 \tag{12b}$$

In some convolutional layers, we also apply a dilation factor to the filter. This increases the spatial support of the filter, without increasing the number of trainable weights per layer (Persello & Stein, 2017). The dilation is achieved by inserting zeros between filter elements, as shown in Figure 11.

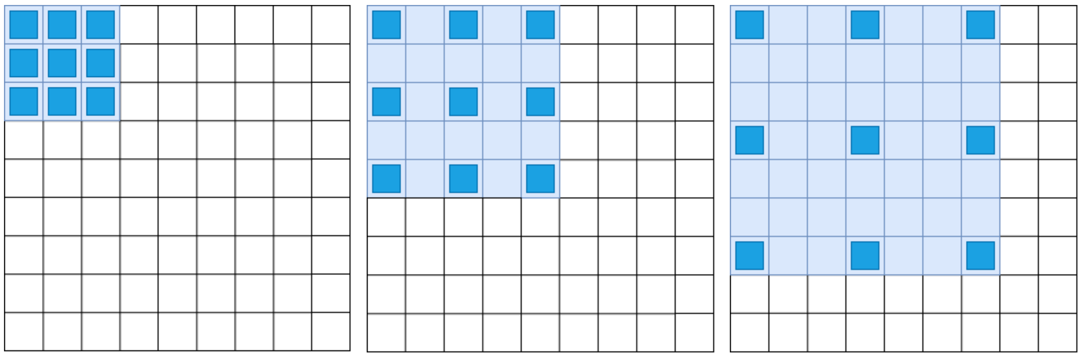


Figure 11: Filters with increasing spatial support. From left to right - filters having a dilation factor of 1, 2 and 3 respectively. Light blue region depicts the receptive field, whereas weights are applied only on the dark blue pixels. Rest of the pixels on the receptive field have a weight value 0 associated with them.

A dilation factor of d applied to $F \times F$ weights would resize the filter to a dimension F' given in Equation 13. When dilation is used, F' will replace F in Equation 12.

$$F' = d(F - 1) + 1 \tag{13}$$

For simplicity, we train the network on a square input image, of dimension M, i.e. $H = W = M$. By keeping the stride for each convolution as 1 pixel, we use an appropriate combination of F, P, and d to keep the dimensions of input-output feature maps the same at every layer. This is done so that we can achieve pixel-wise classification in a fully convolutional sense (Shelhamer et al., 2017). Each convolutional layer is then followed by a Batch Normalization layer and a leaky ReLU activation function.

3.2. Merging and Pooling

During our experiments, a max pooling layer is also applied. It carries a window of size $S_p \times S_p$ pixels, and moves with a stride of S_p pixels. We do not use padding in these layers. By keeping $P = 0$ and $F = S = S_p$ in Equation 12, we see that these layers can downsample an input feature map by a factor S_p.

A SWIR band of an optical satellite data might not be in the same resolution as the other VNIR bands. In order to use these bands for cloud cover analysis, we need to fuse, or merge them together by concatenation. This requires all bands to be in the same dimensions. To carry this out, we resample the bands either by max pooling or transposed convolutions (explained in the next section), which is then followed by a concatenating operation. The resampling and the concatenation is incorporated within the CNN architecture, and the network is trained in an end-to-end manner. This approach has shown higher accuracy as compared to traditional methods where resampling and fusion precede and are performed separately from the CNN training (Bergado et al., 2018).

In order to achieve classification maps in the same dimensions as the input, high resolution VNIR images, we have adopted the fully convolutional approach for semantic segmentation, as proposed by Shelhamer et al., 2017. As our CNN can reduce the dimensions of an intermediate feature map by a factor of S_p, or it might consist of a band in a lower dimension (such as SWIR), we use transposed convolutions which bring all bands and feature maps to a common, higher dimension.

3.3. Transposed Convolutions

Transposed convolutions are reverse operations compared to a regular, forward convolution. This means that, from the output feature of a regular convolution, a transposed convolution can help achieve a feature map which is of the same dimensions as the input of the regular convolution. Thus, for a forward convolution decreasing the dimensions of an input feature map by a factor S, a transposed convolution will *increase* the dimensions of a given feature by the factor S. The utility of such an operation is that it can help decode compressed feature maps, or help in upsampling any given feature channel.

To maintain the same connectivity pattern as its corresponding regular convolution, a transposed convolution often involves adding multiple rows and columns of zeros to a feature map. This acts as a disadvantage because it involves a lot of unnecessary zero-valued multiplications (Dumoulin & Visin, 2016).

A transposed convolutional layer having an input feature of dimensions $M \times M$, will produce an output feature map of dimensions $M' \times M'$, given by Equation 14. Here, p is called the cropping factor, and all other terms have the same meaning as used earlier.

$$M' = S(M-1) + F - 2p \tag{14}$$

3.4. Learning Algorithm

The network is trained by minimizing a cross entropy loss function. The objective of the training is to reduce the error computed by this (loss) function, for a weight vector $\boldsymbol{w}$ used by the network. This loss function is given in Equation 15.

$$E_N(\boldsymbol{w}) = -\sum_{i}^{N} \boldsymbol{z}_i \cdot \log(\boldsymbol{y}_i) \tag{15}$$

Here, N are the number of training samples (pixels) in a mini-batch, $\boldsymbol{z}$ and $\boldsymbol{y}$ are one-dimensional vectors having size equal to the number of classes in the input image. $\boldsymbol{z}$ is made up of zeroes, except at the index corresponding to the pixel's labeled class, which has a value of one. $\boldsymbol{y}$ is made up of normalized values coming from softmax output of the final classification layer (Equation 10 in Section 2.1.1). For each iteration of a mini-batch, the weights are modified in the direction of decreasing error, as given by Equation 5 in Section 1.2.3. The weight modification in every new sweep (through a mini-batch) can be accelerated if we incorporate the modified weights from the previous sweep (Rumelhart et al., 1986). This is shown in Equation 16, where t represents the t^{th} sweep through a mini-batch, α is the momentum and η is the learning rate. Both α and η range between 0 and 1. Such gradient descent methods have shown better generalization than adaptive methods such as Adam and AdaGrad (Wilson, Roelofs, Stern, Srebro, & Recht, 2018).

$$\Delta \boldsymbol{w}_t = -\eta \frac{\partial E_N(\boldsymbol{w})}{\partial \boldsymbol{w}_t} + \alpha \Delta \boldsymbol{w}_{t-1} \tag{16}$$

In order to avoid overfitting, the loss function of Equation 15 is penalized by a squared L^2 norm of the weight vector $\boldsymbol{w}$. The contribution of this norm is controlled by a parameter λ, known as weight decay. The modified loss function is given as $Q_N(\boldsymbol{w})$ in Equation 17.

$$Q_N(\boldsymbol{w}) = E_N(\boldsymbol{w}) + \lambda \, ||\boldsymbol{w}||_2^2 \tag{17}$$

4. METHODOLOGY

This chapter first talks about a general configuration that we used to build our network. The configuration was inspired by the FuseNet architecture proposed by Bergado et al., (2018). The chapter further talks about a baseline architecture that we started off with. Subsequently, in Section 4.4, it shows how the parameters of the baseline architecture were fine-tuned. This was done in order to achieve optimum performance measures, like those explained in Section 4.5. Finally, in order to understand the relevance of SWIR, and to highlight the usability of CNNs over traditional thresholding algorithms, Section 4.6 explains how the network models were modified in this regard.

4.1. General Configuration

A general configuration of the network models experimented with is shown in Figure 12. A Batch Normalization layer followed every convolutional and transposed convolutional layer. The convolutional layers further had a leaky ReLU (with a=0.1) activation function. The activation used towards the end of the Concatenated Feature Block (CFB) was a softmax function, which could segment the data into four classes. This was done because we wanted our data to be segmented into Clouds, Snow, Shadows and Rest of the region. Moreover, all the convolutional layers were designed such that the output feature maps of a layer were of the same dimensions as the layer's input feature maps.

The output of the softmax activation (the predicted map of Figure 12), and a set of manually (visually) labeled reference pixels were then supplied to a cross entropy loss function, to calculate the error between predicted and the true (reference) class of every pixel. The feature blocks were fed with (or trained by) image patches, as explained in the next section.

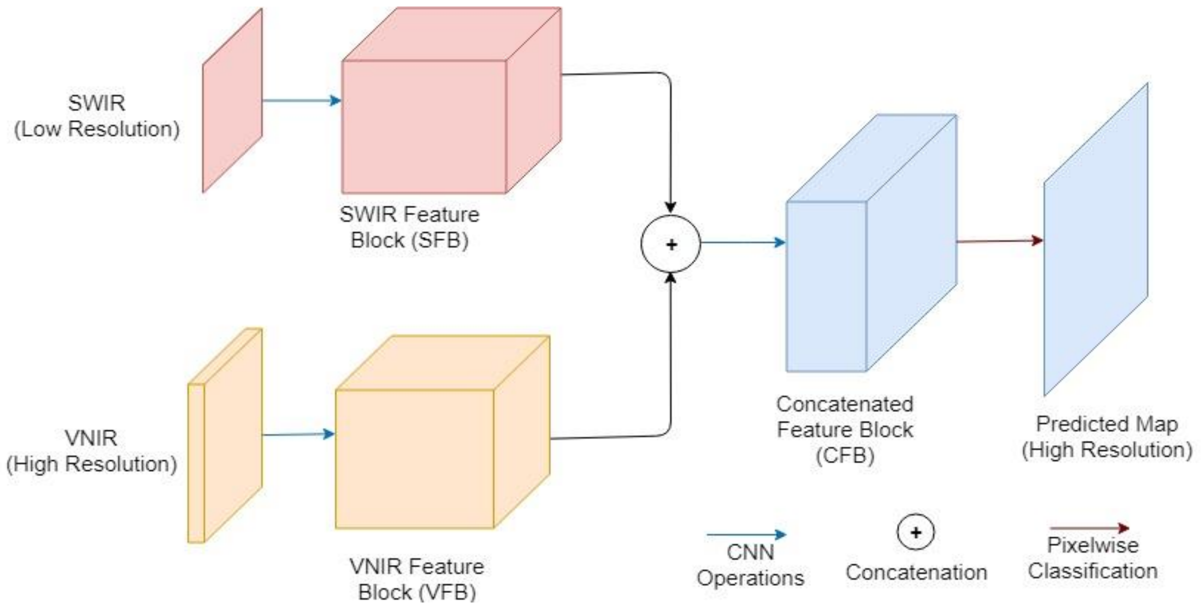


Figure 12: General CNN structure for cloud detection adopted from Bergado et al. (2018)

4.2. Training Setup

Initially, all the available dataset was normalized. To train our neural networks, 2000 'patches' were randomly chosen across each of the training tiles (for tiles and dataset, refer Chapter 5). These patches were fed in a mini-batch size of 32 i.e., 32 patches per iteration (forward and backward pass) and a total of 250 such iterations to make one 'epoch'. For every M×M patch selected from the SWIR band, corresponding 4M×4M patches were selected from the three VNIR bands, and fed to the SWIR Feature Block (SFB) and VNIR Feature Block (VFB), respectively (Figure 13). As our networks were built to make predictions at the higher (VNIR) resolution, a corresponding 4M×4M patch was also selected from the visually labeled reference map (Chapter 5). This helped in calculating the loss function, and hence in training the networks. As these patches were chosen randomly, it is possible that they overlapped amongst themselves and incorporated a sense of redundant learning of contextual information.

Furthermore, to assess the training, 500 patches were randomly chosen across the same tiles for validation purposes. The loss function and its convergence over the validation set was used to analyze the networks' robustness. We trained the networks for 200 epochs initially, and then gradually reduced the number of epochs to 70, and then 40. This was because the loss function had converged significantly way before the 40th epoch, and there was no further drop in its value. The learning rate was logarithmically reduced between 10^{-6} and 10^{-7}, with a step size equal to the number of epochs. The weight decay and momentum were kept as 5×10^{-4} and 0.9, respectively. The filter weights had a normalized initialization (Glorot & Bengio, 2010) and all the experiments were carried out using the MatConvNet library version 1.0-beta-23, compiled with CUDA 10.0 and cuDNN 7.4.

4.3. Baseline Architecture

A baseline architecture was developed, called Fuse1, so that the fusion takes place at the lower (SWIR) resolution. The VFB was made by operating two layers of convolutions on the input VNIR image. The leaky ReLUs in the VFB were followed by a max pooling layer each. Two layers of max pool were introduced to bring the VFB at the resolution of SWIR. Parallelly, the SWIR band was convolved with 1×1 convolutions to make the SFB. Both blocks were concatenated at the same (lower) resolution. The CFB further involved two layers of convolutions, with different dilation factors and was finally upsampled using two layers of transposed convolutions. This was done so that the predictions could be made at the higher (VNIR) resolution. Figure 13 shows how the feature maps transition in the baseline architecture, whereas Table 2 specifies the intricacies of the CNN layers used.

Table 2: CNN parameters for the baseline architecture (Fuse1). The VFB, SFB, and CFB correspond to VNIR Feature Block, SWIR Feature Block and the Concatenated Feature Block, respectively

VFB	SFB	CFB
Conv9-1-8 maxpool Conv9-1-16 Maxpool	Conv1-16	Conv5-1-64 Conv5-2-64 TConv4-2-1-64 TConv4-2-1-64 Conv1-1-4

In Table 2, every convolutional layer is represented as Conv<filter width>-<dilation factor>-<number of filters>. Example: a Conv5-2-64 layer means 64 filters of size 5×5, with a dilation factor of 2. All convolutional filters move with a stride of 1. Appropriate padding was applied to keep the dimensions of the input and output feature maps the same. Max pooling layers are represented as maxpool, all of them having a 2×2 window, moving with a stride of 2. Transposed Convolutions are represented as TConv<filter width>-<stride or upsampling factor>-<cropping factor>-<number of filters>. The filter width, the upsampling factor (or stride) and the cropping factor are related as in Equation (14). As we wanted our network to semantically segment the data into four classes, the last convolutional layer had four channel outputs.

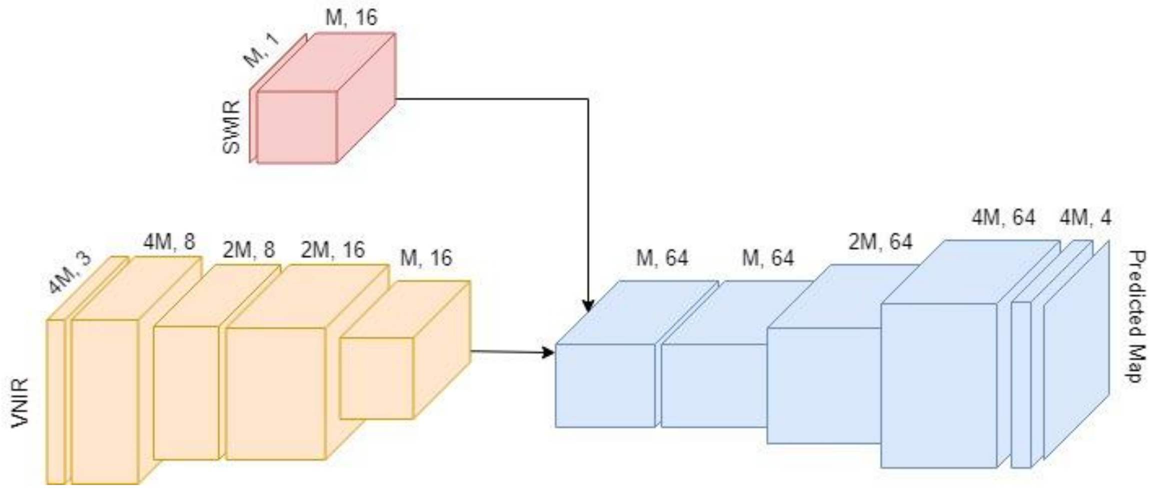


Figure 13: Features transitioning in the baseline architecture. Features are represented as A, B which means B bands of size A×A.

4.4. Experiments on the Baseline Architecture

Initially, we kept the patch size (value of M, in Section 4.2) as 32 pixels. We then fine-tuned our model to achieve higher classification accuracy in a manner listed below.

1. We first made a comparative study on the two types of pooling. This was to observe the effect of pooling on VNIR feature maps. All the max pooling layers in the VFB were changed to average pooling layers. We then experimented downsampling with evenly-strided convolutional layers, to understand if the learnable filter weights bring any advantage or not.

2. We then experimented with multiple fusion strategies. Instead of downsampling VNIR and concatenating at the lower resolution, we concatenate at the higher resolution. This was done by upsampling the baseline SFB with two layers of TConv4-2-1-16. This helped in doubling the SWIR resolution twice. In this model, the TConv layers and the maxpool layers in the original CFB and VFB, respectively, were discontinued with. Table 3 showcases the structure of this model, called Fuse2.

Table 3: CNN structure for Fuse2

VFB	SFB	CFB
Conv9-1-8	Conv1-1-16	Conv5-1-64
Conv9-1-16	TConv4-2-1-16	Conv5-2-64
	TConv4-2-1-16	Conv1-1-4

Further ahead, we designed two more models called Fuse6 and Fuse7. Fuse6 had the same SFB and CFB as Fuse2 (Table 3), where the SFB was concatenated with the original VNIR bands directly. In Fuse7, the VFB of Fuse1 was concatenated with the original SWIR band and the CFB remained the same.

3. Next up, we used the baseline architecture of Fuse1, and modified the filter sizes in the VFB. We replaced the 9×9 filters with filters of size 3×3, 5×5, 7×7, 11×11 and 13×13.

4. Finally, the effect of changing the patch size was also studied, by modifying the value of M to 20, 50 and then 70.

Although the network was prepared to predict four classes, we majorly focussed on attaining high accuracies (as explained in the next section) for Clouds and Snow. Hence, we refer to our optimum architecture as CloudSNet.

4.5. Performance Metrics

We analyzed all our experimental network models of Section 4.4, based on a combination of metrics. These were as follows:

Overall Accuracy
Overall Accuracy (OA) is the total number of correctly predicted pixels, divided by the total number of labeled reference pixels. We compute the overall accuracy on all the image tiles.

Producer's Accuracy
Producer's Accuracy (PA) is the number of pixels correctly predicted for a class divided by the total number of reference pixels for that class.

User's Accuracy
User's Accuracy (UA) is the number of pixels correctly predicted for a class divided by the total number of predicted pixels of that class. We focus on the PA and UA of Clouds and Snow only. Moreover, we look at PA of Clouds, UA of Snow, and PA of Snow as we want our networks to detect as much of the true cloud pixels, and the least amount of false-snow pixels.

F1-Score
OA can be highly biased if there is an uneven class distribution in the image. PA and UA help in this regard by highlighting the effectiveness of a class's prediction. Thus, the harmonic mean of the PA and UA, known as the F1 score, acts as a useful metric to assess any classifier's performance. We use the average F1 score of Snow and Clouds to compare our network models.

Visual Inspection
Human, visual inspection is handy when comparing the usefulness of different class maps. We majorly checked if the classified outputs were smooth and free of noise. Moreover, we noted if and why multiple classes were getting confused with each other.

Computation Time

An important measure of performance is the computation time involved. As different type of CNNs and their layers involve complex matrix multiplications, it becomes relevant to understand how much time is being spent on training the network models. All our processing took place on a desktop with an Intel Xeon CPU E5-2695 v2 having 128GB RAM and working at 2.4GHz. The training process was accelerated by an NVIDIA Tesla K20Xm GPU.

4.6. Network Comparisons

In order to study the effectiveness of the optimum network obtained through Section 4.4 we compared its performance in a manner described in this section.

4.6.1. Fully Convolutional Networks

Our central hypothesis is that using a resampled SWIR should help in an easier detection of clouds (over snow) in high-resolution optical images. To test this hypothesis, we use the optimum model of Section 4.4 and compare its performance with similar fully convolutional networks that only take the VNIR bands or the SWIR band as the input. We refer to these networks as FCN_{VNIR} and FCN_{SWIR}, respectively.

4.6.2. Cloud and Shadow Mask Generator for RS-2

We also compared our network structure with Resourcesat-2's Cloud and Shadow Mask Generating (CSMG) software of National Remote Sensing Centre (2017), which uses the traditional threshold-based algorithms. This software tool works on LISS-III data, not on LISS-IV, and classifies the input raster into Clouds, Shadows, and Rest. In order to compare our network model with this software tool, we modified CloudSNet in the following aspects:

1. We used Band 5 from AWiFS (in a manner described in Section 5.3.1) as the SWIR input, and for every M×M patch used for this band, we took a 2M×2M patch on the VNIR bands of LISS-III.

2. As the final classification should be on the higher resolution (i.e. 2M×2M in this case), $CloudSNet_2$ was prepared by removing an appropriate upsampling/downsampling layer from CloudSNet.

3. The original output from $CloudSNet_2$ had four classes. We combined the 'Snow' with the 'Rest' pixels so that the comparison with CSMG could be more viable.

5. DATASET

This chapter talks about the data that we've used, and how it has been made compatible for the CNN classifier.

5.1. Satellite and Sensors

The dataset used was that of Resourcesat-2 satellite, from the Indian Remote Sensing programme. The satellite carries three multispectral pushbroom scanners, majorly meant for monitoring crops, providing assistance to farming activities, and managing water resources. The satellite operates in a sun-synchronous orbit 817 km above the Earth, with all the sensors looking at nadir.

The three sensors in their decreasing spatial resolutions are AWiFS, LISS-III and LISS-IV. All sensors acquire data in the same spectral bandwidth of visible and near-infrared bands. The AWiFS and LISS-III additionally capture short-wave infrared signals, whereas LISS-IV has an off-nadir viewing capability. The details of the sensors are specified in Table 3. Band 5 corresponds to the SWIR band, while bands 2, 3 and 4 correspond to the VNIR bands.

Table 4: Sensor specifications of Resourcesat-2 (National Remote Sensing Centre, 2003)

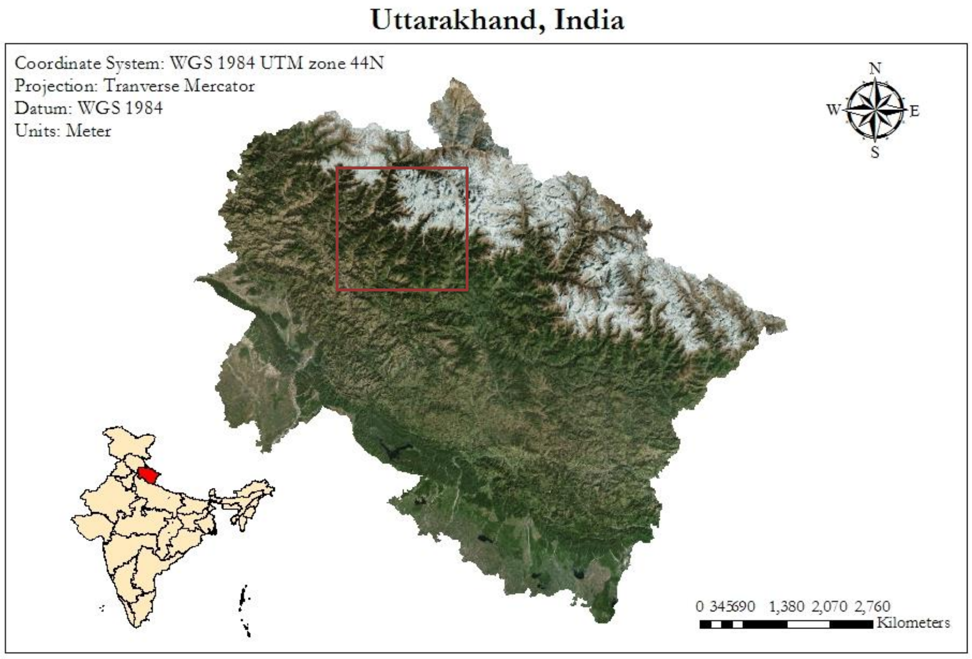
Specification	AWiFS	LISS-III	LISS-IV
Input Resolution (m)	56	23.5	5.8
Output Resolution (m)	56	24	5
Spectral Bands (μm)	B2 0.52 – 0.59 B3 0.62 – 0.68 B4 0.77 – 0.86 B5 1.55 – 1.70	B2 0.52 – 0.59 B3 0.62 – 0.68 B4 0.77 – 0.86 B5 1.55 – 1.70	B2 0.52 – 0.59 B3 0.62 – 0.68 B4 0.77 – 0.86
Swath (km)	740	140	70
Revisit (days)	5	24	5

5.2. Study Area

The area we chose for our study was the state of Uttarakhand in India. The northern part of the state has a mountainous region expanding to nearly 47,000km^2 ("Uttarakhand," 2017), with most of it lying in the Greater Himalayan (Himadri) range. The region has some of the highest and the most rugged mountains in the world, which are covered with thick snow throughout the year. Thus, the area provides a wide variety of snow-covered regions, fit to be analyzed for our problem statement.

Recent studies using LISS-IV have found that the state is home to some of the most vulnerable glaciers in the country (Rawat, 2018). Since major glaciers such as Pindari and Gangotri are situated here, their potential vulnerability can risk in massive floods in the adjoined areas. Hence developing cloud-free snow cover maps becomes essential for water resource management, as well as for predicting floods and assessing potential risk. Thus, our study can contribute to such a scenario.

We chose Path 97, Row 49 from the orbital pass of Resourcesat-2. The satellite captures this scene at around 5:34 in the morning. Figure 14 highlights the area of study.



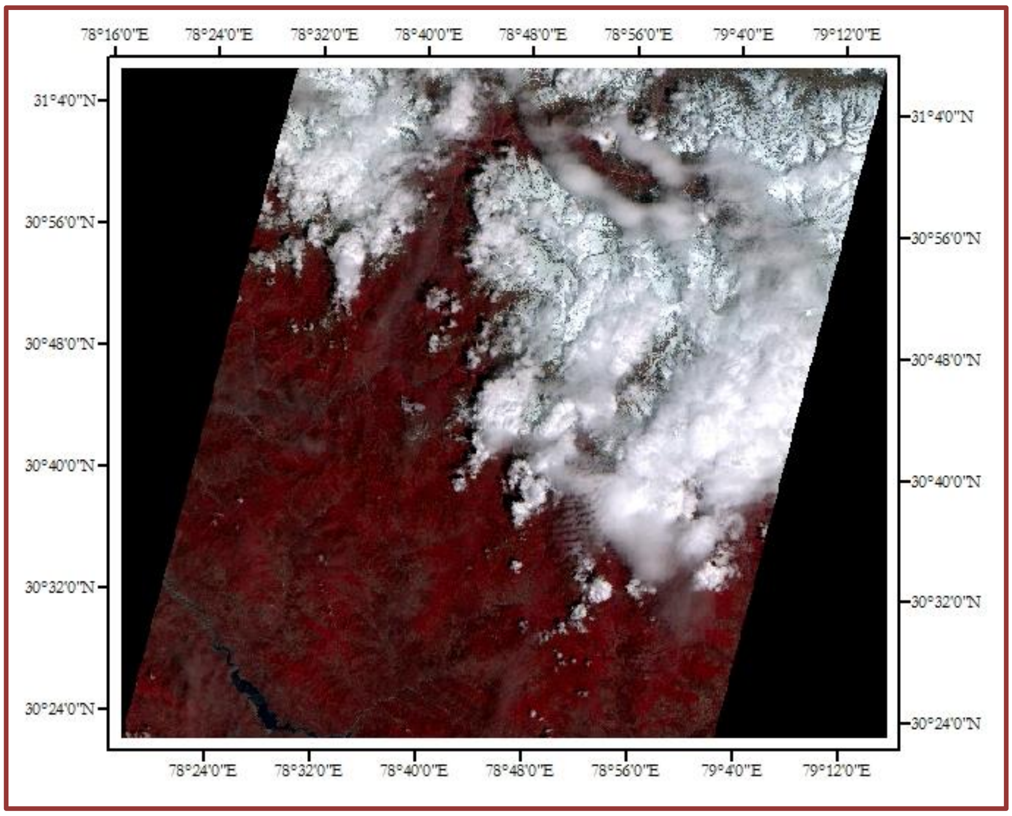


Figure 14: Top - The red box indicating the study area, in the state of Uttarakhand. Above – A LISS-IV False Colour Composite belonging to Path 97 Row 49 of Resourcesat-2, captured on 13th May, 2015

5.3. Data Preparation

To train and build our Neural Network, we looked for images from various dates where LISS-IV data was present. The scenes we chose were captured on 9th October 2014 and 13th May 2015. These consisted of a homogenous amount of clouds and snow, where the features were not easy to distinguish from each other. The VNIR bands of LISS-III and LISS-IV were separately stacked together to form False Coloured Composites.

Eight square regions, of 100km^2 each, were selected from these two scenes. Four of these regions were kept for training and validating the CNN classifier, while the remaining four were used to test the classifier's performance. Figure 15 highlights the eight regions.

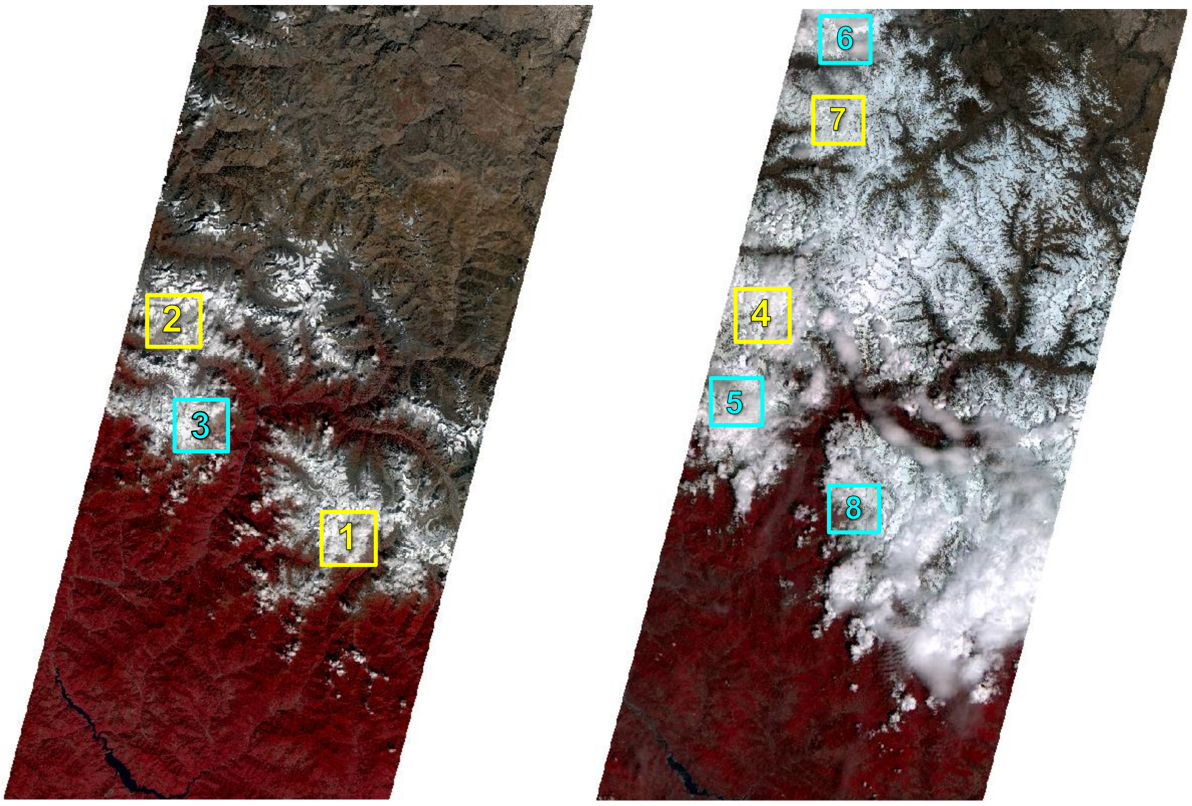


Figure 15: Scenes from 9th October, 2014 (left) and 13th May, 2015 (right). Regions marked with yellow were used for training and validation, while the regions marked with blue were used for testing the classifier.

5.3.1. SWIR Resampling

For training the classifier, we used a lower resolution SWIR band. To classify LISS-IV, we used Band 5 from LISS-III of the same date. And to perform classification on LISS-III data, we used Band 5 from AWiFS, even though LISS-III has its own SWIR.

The original SWIR bands of LISS-III and AWiFS are in 24m and 56m respectively (Table 3), whereas the VNIR bands of LISS-IV and LISS-III are in 5m and 24m respectively. As our classifier required the SWIR channel's resolution to be an even multiple of VNIR's resolution, we resampled the SWIR of LISS-III and AWiFS to 20m and 48m, respectively, using nearest neighbor interpolation.

5.3.2. Reference Labels

As our CNN carries out supervised classification, we provided it with reference data, for learning. Reference labels were created visually for tiles selected in Figure 2. The labels created were at the higher resolution from the available bands. They segment the tiles into Clouds, Snow, Shadows and Rest of the region. Figure 16 shows the reference labels created for tile 1. The number of pixels per class (at the resolution of LISS-IV) for every tile has been described in the Appendix.

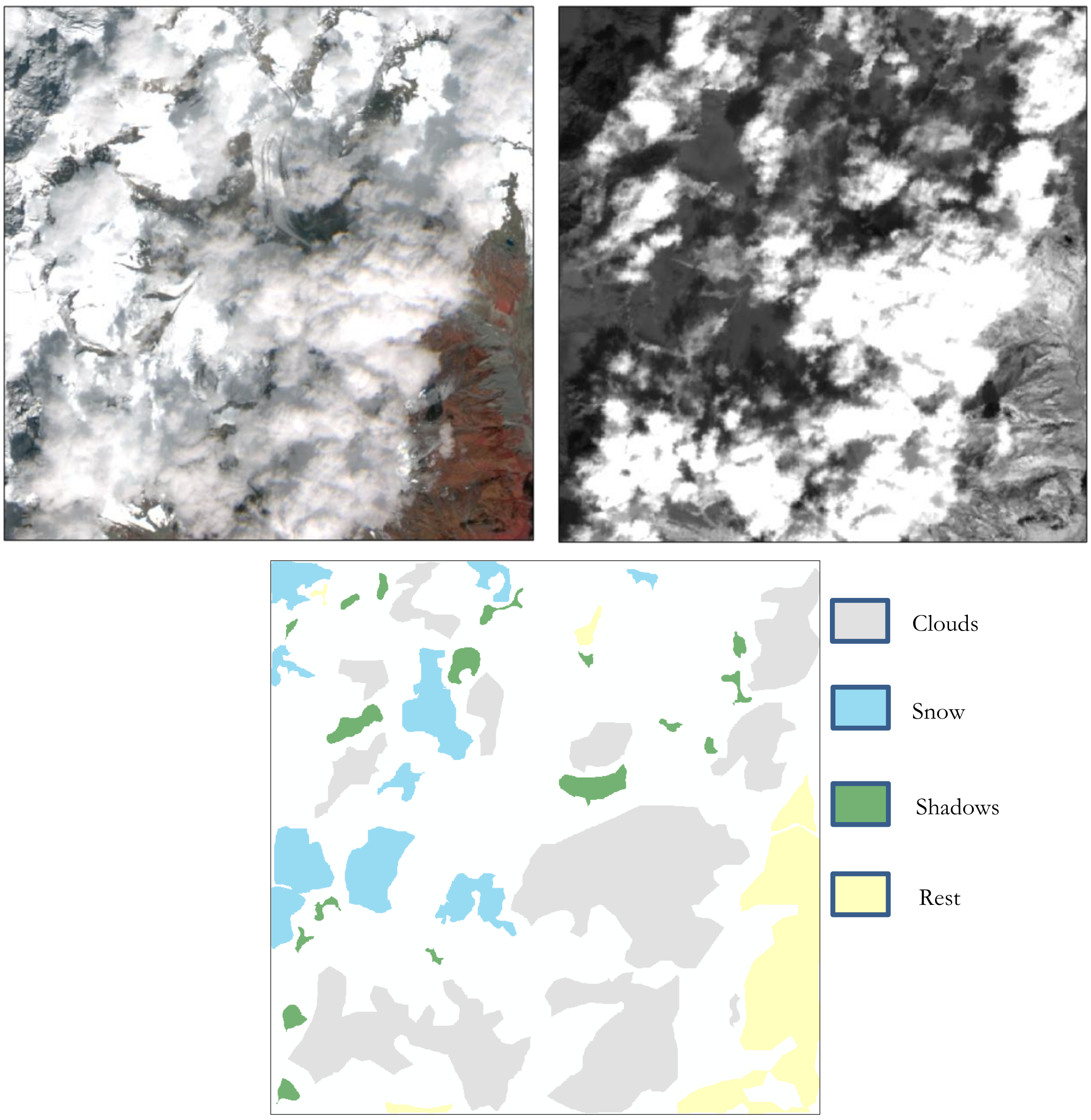


Figure 16: Tile No. 1, Clockwise from top left - False Colour Composite of LISS-IV, Band 5 (SWIR) of LISS-III of the same area and Reference labels created for this tile

6. RESULTS AND DISCUSSION

This chapter is divided into three sections. In the first section, we first understand how changing different parameters and network training strategies affect the performance metrics, to build a CloudSNet. The following section compares the performance of CloudSNet with the similarly-structured FCN_{VNIR} and FCN_{SWIR} to understand the relevance of SWIR resampling. Finally, we compare $CloudSNet_2$ and compare it with the CSMG.

6.1. Sensitivity Analysis

In this section we study the effect of changing the downsampling operation, the fusion style, the filter size(s) and the patch size (M), in that order.

6.1.1. Downsampling Operations

We analysed the effect of pooling operations in the VFB. Figure 17 shows the overall accuracy obtained through different pooling operations on the image tiles.

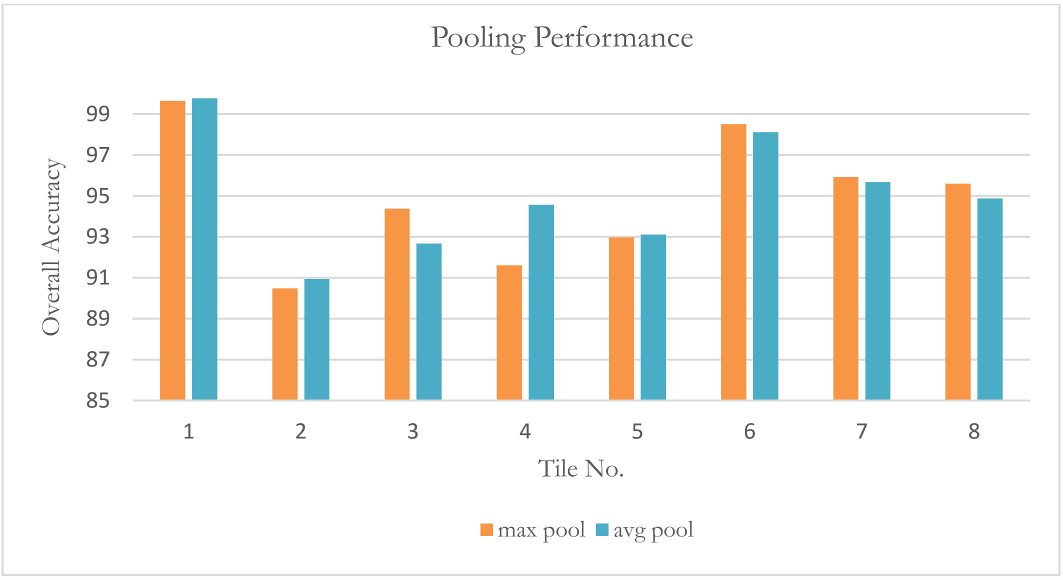


Figure 17: Overall Accuracy of different tiles, using different pooling strategies

Both the pooling operations gave similar performance over most of the tiles. Average Pooling gave a significantly higher accuracy over Tile 4, whereas Max pooling gave higher accuracy over tiles 3 and 8. One possible reason is that Tiles 3, and 8 are spatio-spectrally diverse, whereas Tile 4 is spectrally uniform and has fewer variations. Hence, detecting the maximum activations from a local neighborhood of tiles 3 and 8 helps in feature detection, whereas averaging over the local activations is optimum for Tile 4. Additionally, the final predicted maps through max pool have smoother class boundaries, compared to average pooling (Figure 18). This is because, at the boundaries of different spatial features in an image, filters detecting maximum activations in a local neighborhood will help segregate the features; whereas averaging over the local neighborhood at the boundaries will not help in distinct segregation.

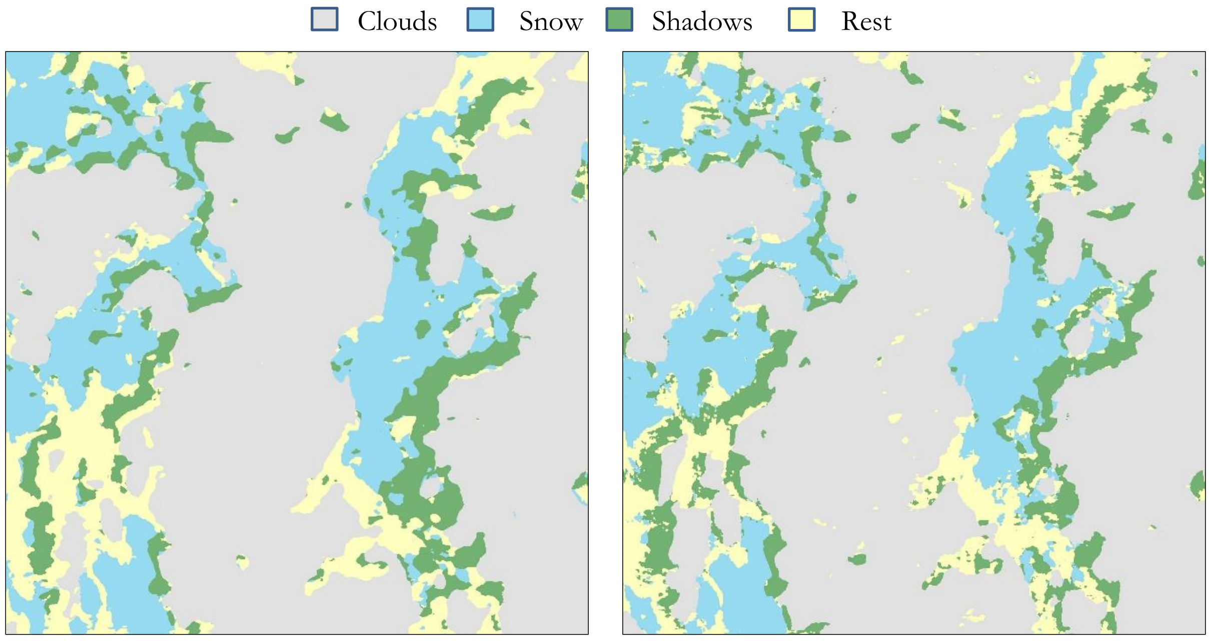


Figure 18: Classified maps of Tile No. 4 incorporating Max pooling (left) and Average Pooling (right). The predicted maps through Max Pooling have smoother boundaries.

We also experimented downsampling with Convolutional layers having an even stride. In this case, the learned downsampling gave more or less the same PA and UA for clouds and snow, as the pooling layers did (Figure 19). Hence, the additional trainable parameters that the convolutional layers involved did not help improve the accuracy much. Furthermore, average pooling gave a higher total PA for clouds.

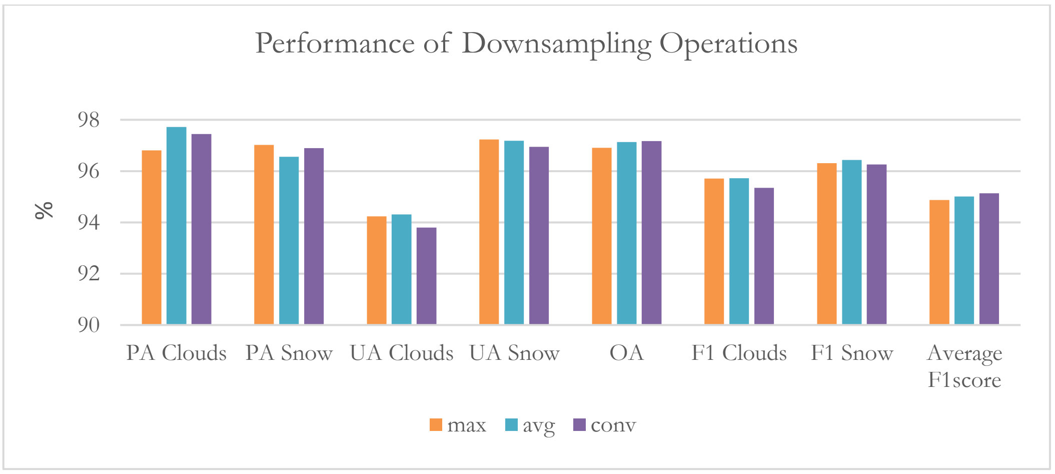


Figure 19: Comparing the performance of various downsampling operations. 'max' is max pooling, 'avg' is average pooling, and 'conv' is downsampling with convolutions having even stride.

Thus, we chose max pooling for all our subsequent experiments, as it helped in smoother class predictions, and did not involve trainable parameters.

6.1.2. Fusion Strategy

The different fusion strategies we experimented with are explained in Table 4. Fuse1 was the baseline architecture, and all other network models were derived from it after slight modifications. Fuse2 and Fuse6 involve transposed convolutions on SWIR and there are no downsampling operations involved here before the concatenation. Comparatively, Fuse1 and Fuse7 involve concatenation at the lower resolution, and hence their VFBs involve downsampling. The fusion styles described in Table 5 were majorly used to assess the total Producer's Accuracy of Clouds and Snow across all the tiles. These are shown in Figure 20.

Table 5: Fusion Experiments

Network	Style	Resolution at Concatenation
Fuse1	SFB merged with VFB	Low
Fuse2		High
Fuse6	SFB merged with VNIR	High
Fuse7	SWIR merged with VFB	Low

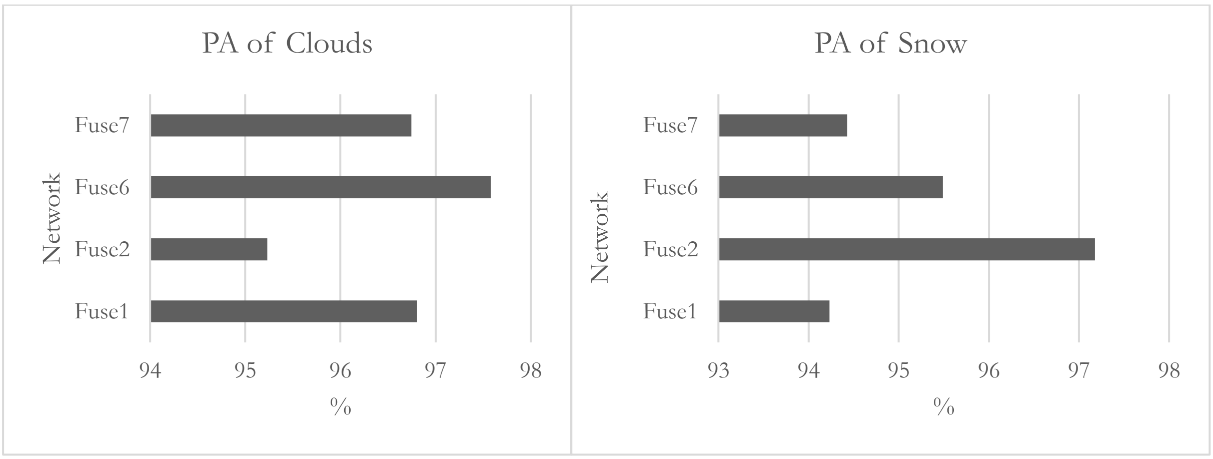
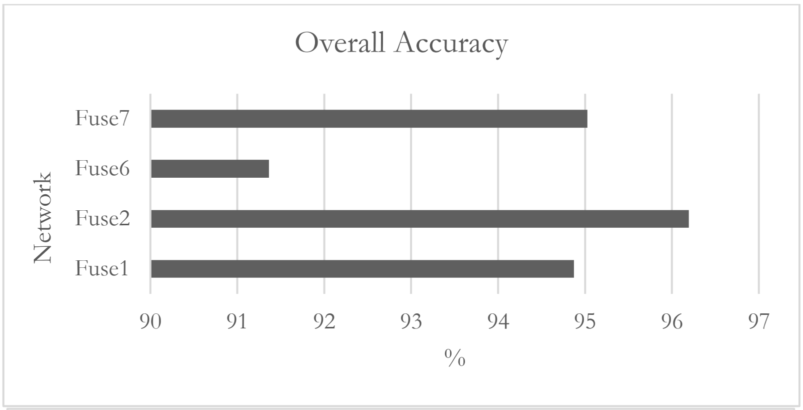


Figure 20: Producer's Accuracy of Clouds (left) and Snow (right) through different fusion styles

From Figure 20 we see that Fuse6 gave a higher PA for Clouds, while Fuse2 gave a higher PA for Snow. Both these networks employ the learned upsampling of SWIR, extracting information which helped discriminate between the snow and cloud features. Comparatively, the downsampling of VNIR bands in Fuse1 and Fuse7 gave lesser accuracy in this regard. Moreover, clouds could be detected easily by concatenating the convolved SWIR directly with the original VNIR bands, and not requiring additional VFB (Fuse6 gave a higher cloud PA than Fuse2).

Furthermore, Fuse2 gave the highest overall accuracy, jointly followed by Fuse1 and Fuse7 (Figure 21). The latter two networks, both concatenated at the lower resolution, gave the highest average F1 score. The overall accuracy was computed over four classes (Clouds, Snow, Shadows and Rest) and the F1 score was averaged over two classes (Clouds and Snow). As the VNIR bands are more *feature rich*, compared to the SWIR, multiple convolutions on them before the concatenation helped in an accurate prediction of all four classes. Fuse1 and Fuse7 also gave a higher User Accuracy for the two main classes (Appendix), and thus, the final network (fusing style) selection depends from case to case. A network could be selected depending on the priority given to either PA or UA, for either Snow, Clouds or both.



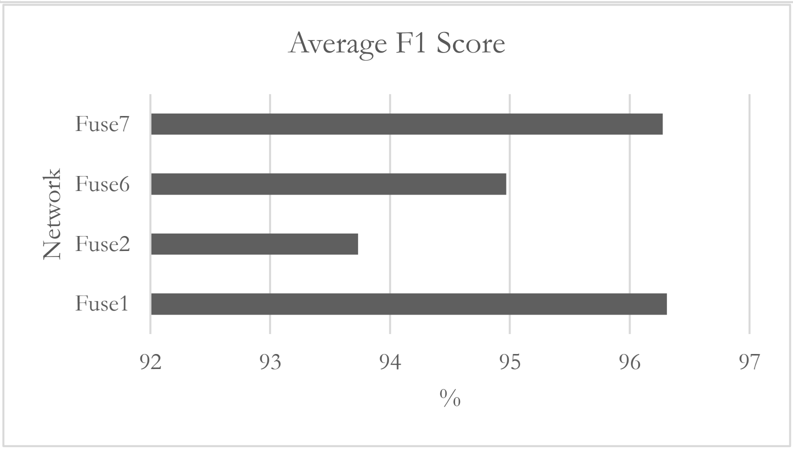


Figure 21: Overall Accuracy (top) of different concatenation strategies, and their average F1 scores (above)

There were some noticeable disadvantages of using transposed convolutions on the SWIR. From Figure 22 we see that Fuse6 and Fuse2, both made up of upsampled SWIR, take nearly twice as much time for training, compared to other network structures. Also, the final class boundaries produced by them are quite uneven. Comparatively, the maxpool layers of Fuse7 and Fuse1 give much smoother boundaries, and the classes are more uniform (Figure 23). The lack of uniformity in transposed convolutions is because of the insertion of additional zeros to their input feature maps, which dilates the local spatial neighborhood.

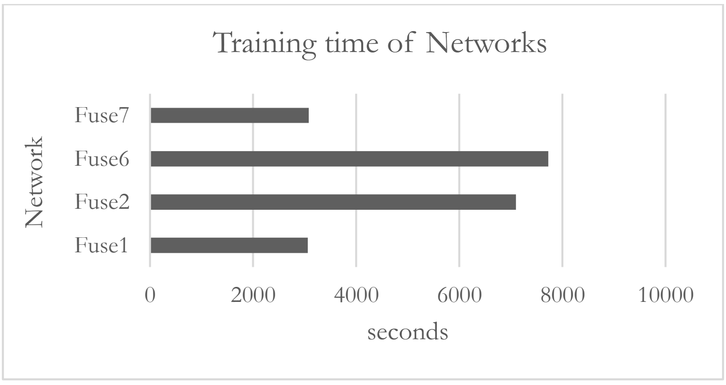


Figure 22: Training time for different fusion strategies

From these experiments, we see that although having dense convolutions on the SWIR helps in cloud detection, upsampling through transposed convolutions can make the classes 'patchy', with uneven class boundaries. Moreover, multiple convolutional layers on the VNIR bands certainly helped in snow estimation (as the PA for snow is higher for Fuse2, compared to Fuse6 in Figure 20).

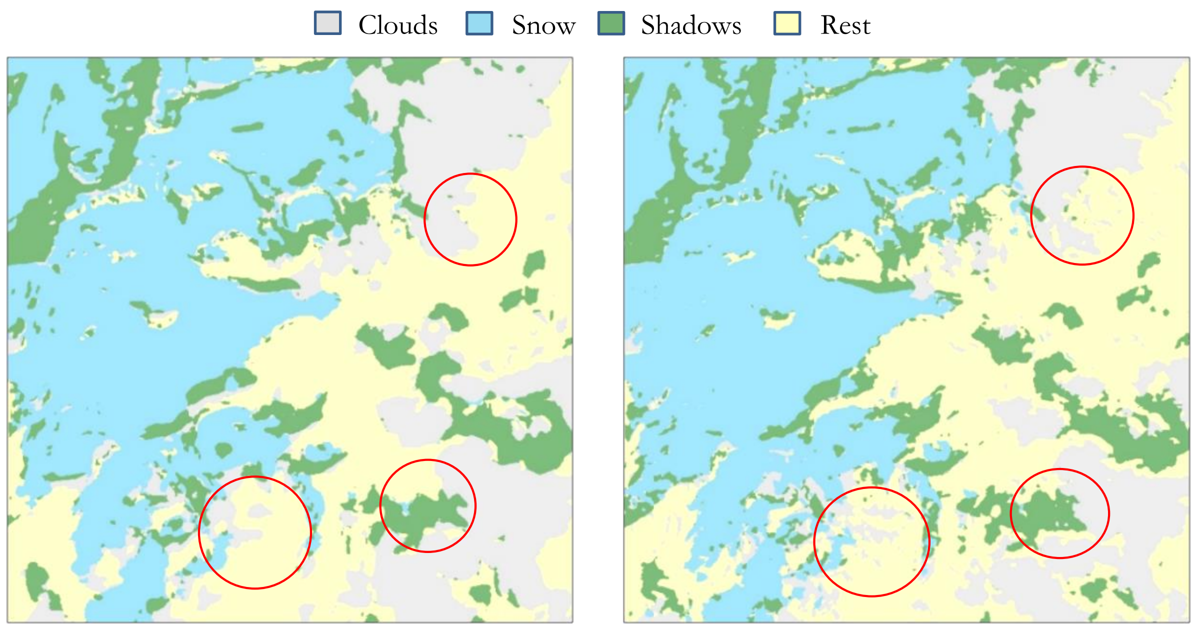


Figure 23: Tile no. 3 (top), classified map by Fuse1 (above, left) and classified map by Fuse2 (above, right). Regions marked with red have mixed, non-uniform classes for Fuse2, compared to Fuse1.

6.1.3. Filter Size

We compared different filter sizes on the VFB. As this decides the receptive field and the spatial window sliding over an area, the filter size becomes an important parameter in extracting and understanding spatial features, especially from VNIR bands. The VNIR bands give us most of the information related snow extent, presence of shadow regions, and the prevalence of the 'Rest' class. The information from these bands also helps in segregating 'Rest' regions, and thin Clouds, as both these features have similar spectral properties in the SWIR band. Thus, optimizing the convolutions on the VNIR bands help in understanding (and sharpening) class boundaries.

The baseline architecture had 9×9 filters in the VFB. We experimented with different square filters of odd dimensions, starting from 3×3 to 13×13. The most optimum filter size we found was that of 5×5. This is because it showed the maximum performance across most of the metrics – F1 scores for Clouds, snow, average F1 score, Producer's Accuracy of Clouds and Overall Accuracy. All these metrics were computed for the entire dataset shown to the networks, i.e. training, as well as the test tiles. Figures 24 to 26 show the performance of different filter sizes in the VFB against various metrics.

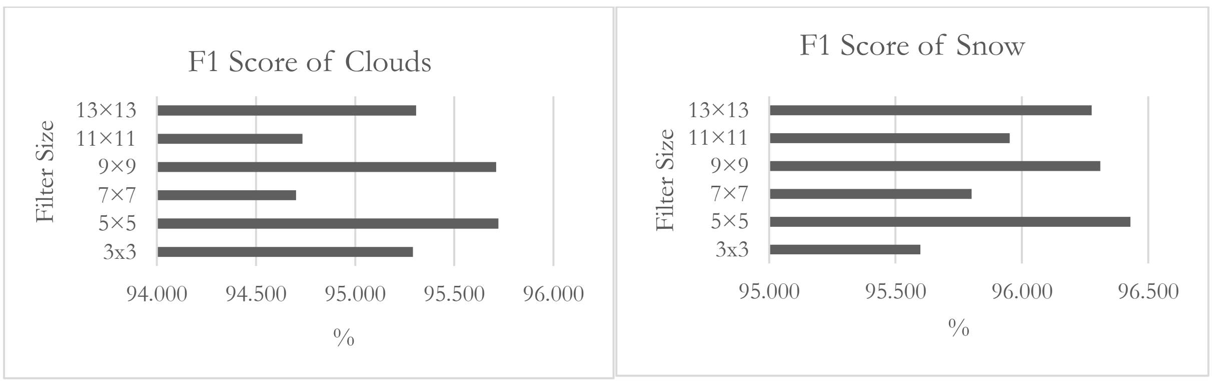
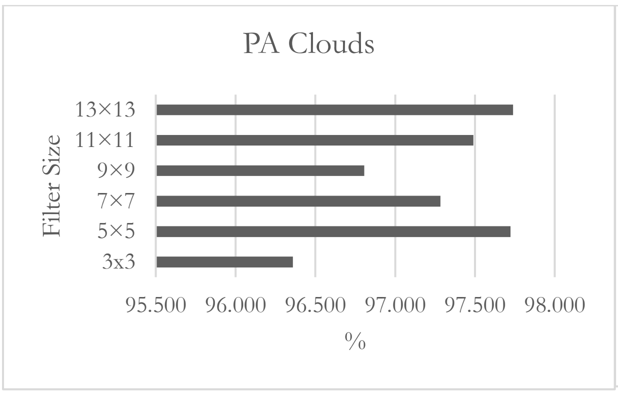


Figure 24: Figures showing the F1 score of Clouds (left) and Snow (right) for different filter sizes in the VFB. Filters of dimension 5×5 and 9×9 give the highest scores for these metrics



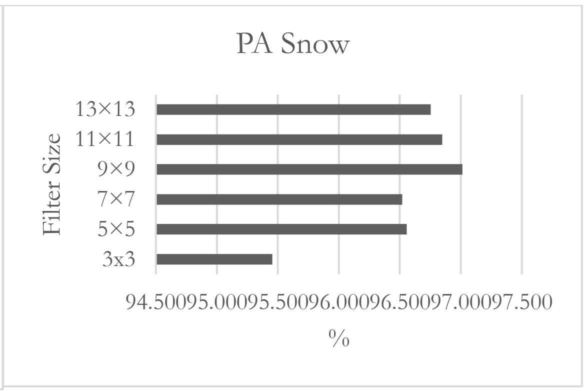
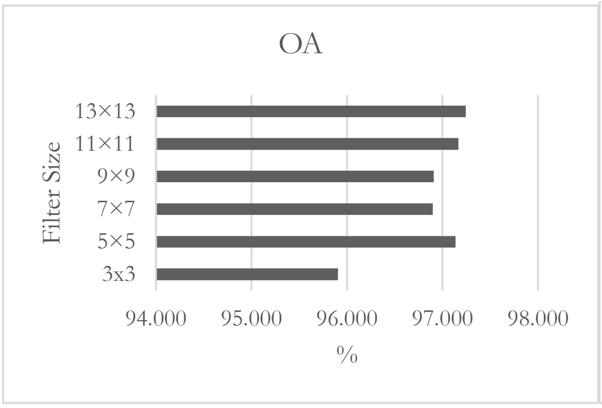


Figure 25: Figures showing the PA of Clouds (left) and Snow (right). 5×5 filter gave the highest PA for clouds, while 9×9 filter gave the highest PA for Snow



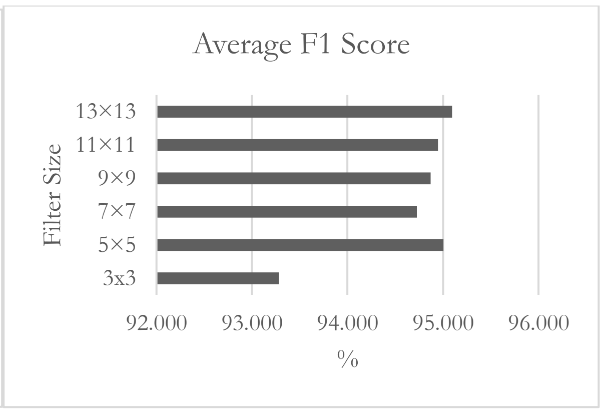


Figure 26: Overall Accuracy (left) and the Average F1 score (right) obtained through different filter sizes

All these networks with varying filter sizes took similar time (approximately 3000s) for training. Filters of size 5×5 and 7×7 showed some salt and pepper noise in a few regions. Moreover, working with 5×5 filters was optimum as they involve some of the least number of trainable parameters, compared to other filters. Also, the spatial window that these filters look at, is big enough to understand the contextual information.

6.1.4. Patch Size

Increasing the patch size increases the contextual information that a network can learn, and hence should increase the performance accuracies. But this also depends on the average size of the objects that we are trying to extract from the images. The performance accuracies might start saturating or decreasing after a certain (patch size) value.

Hence, we analyzed the effect of varying the size of image patches (*M*×*M*) used for training our baseline model. We experimented with value of M as 20, 32, 50 and 70. For every M×M patch selected on the SWIR band, a 4M×4M patch was selected on the VNIR bands. The training time increased with increasing patch size, as expected. We further computed overall performance metrics for the training tiles (Figure 27, left), and only for the test tiles (Figure 27, right). From Figure 27 we see that the performance metrics across the training tiles showed an increasing trend, peaking at M equal to 50. Even for the test tiles, a patch size of 50×50 on SWIR gave most of the high performance values. Hence a patch size of 50 seemed to be of optimum value. Furthermore, we noticed that increasing patch size also increased salt and pepper noise in the final images.

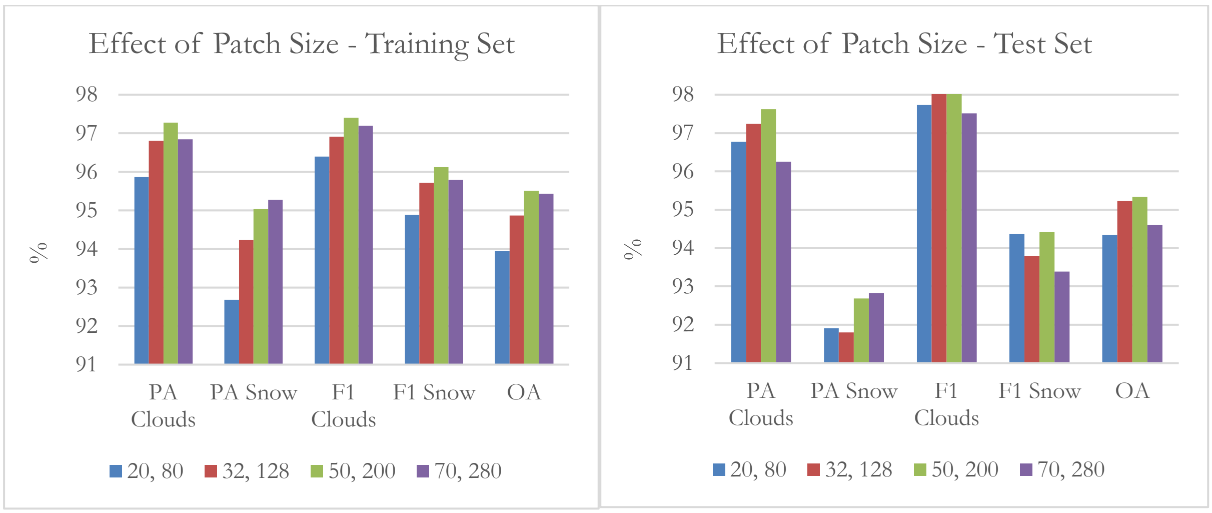


Figure 27: Effect of varying the patch size on performance metrics across training tiles (left) and across the test tiles (right). A patch size of (A, B) corresponds to A×A patches on the SWIR band, and B×B patches on the VNIR bands.

6.2. Network Assessment

Taking the analysis from the previous section into consideration, we built CloudSNet. As the transposed convolutions on SWIR took twice the time for training, we kept the SFB dense, but with regular convolutions. We used 5×5 filters in the VFB, downsampling the feature maps with maxpool layers prior to the concatenation, and used a patch size with M equal to 50, to train our model. We assessed the credibility of the architecture in two ways. First, we compared it with FCN_{VNIR}. As FCN_{VNIR} has the same structure as CloudSNet but without an SFB, comparing with it is beneficial to our study. It helps us understand if introducing a SWIR, by resampling and fusing it inside an FCN-on-VNIR, would be really advantageous in detecting clouds (over snow) as compared to a regular FCN-on-VNIR. We also composed an FCN-on-SWIR (FCN_{SWIR}) and observed the differences in the three models. Second, we compared CSMG with $CloudSNet_2$. As the CSMG tool performs unsupervised classification using only spectral attributes, by comparing it with $CloudSNet_2$, we can analyze the effect of incorporating spatial information for classification. Table 6 shows the network architectures of CloudSNet (along with $CloudSNet_2$), FCN_{VNIR} and FCN_{SWIR}.

Table 6: Table showing the architecture of CloudSNet (left), FCN_{VNIR} (middle) and FCN_{SWIR} (right). $CloudSNet_2$ was made by removing a layer of transposed convolution, and the second maxpool from CloudSNet.

CloudSNet		**FCN_{VNIR}**	**FCN_{SWIR}**
VFB	**SFB**		
Conv5-1-8	Conv1-1-8	Conv5-1-8	Conv1-1-8
maxpool	Conv3-1-16	maxpool	Conv3-1-16
Conv5-1-16	Conv5-1-32	Conv5-1-16	Conv5-1-16
maxpool		maxpool	Conv5-1-32
CFB		Conv5-1-32	Conv5-2-64
Conv5-1-64		Conv5-2-64	TConv4-2-1-64
Conv5-2-64		TConv4-2-1-64	TConv4-2-1-64
TConv4-2-1-64		TConv4-2-1-64	Conv1-1-4
TConv4-2-1-64		Conv1-1-4	
Conv1-1-4			

6.2.1. Comparison with FCN_{VNIR} and FCN_{SWIR}

Here, we show the comparative results of the three networks i.e. CloudSNet, FCN_{VNIR} and FCN_{SWIR}. We divide the performance metrics into two categories – major and minor. The major metrics are the accuracies and scores associated to clouds and snow; whereas the minor metrics are the class accuracies for shadows, and the Rest, along with the overall accuracy. All the metrics reported in this subsection are the cumulative metrics upon all the four test tiles. Table 7 shows the major performance metrics, while Table 8 shows the minor performance metrics.

Table 7: Major performance metrics (in %) of CloudSNet, FCN_{VNIR}, and FCN_{SWIR}

Network	PA Clouds	UA Clouds	PA Snow	UA Snow	F1 Score Clouds	F1 Score Snow	Average F1 Score
FCN_{SWIR}	94.80	98.06	86.90	87.57	96.40	87.23	91.82
FCN_{VNIR}	82.86	96.28	**90.94**	72.16	89.07	80.47	84.77
CloudSNet	**96.90**	**98.45**	90.40	**94.73**	**97.67**	**92.51**	**95.09**

Table 8: Minor performance metrics (in %) of CloudSNet, FCN_{VNIR}, and FCN_{SWIR}

Network	PA Shadows	UA Shadows	PA Rest	UA Rest	OA
FCN_{SWIR}	48.35	55.82	84.53	62.97	88.88
FCN_{VNIR}	78.05	80.72	**98.01**	**82.81**	85.99
CloudSNet	**86.25**	**80.89**	94.50	79.21	**94.31**

From the tables above, we see that, CloudSNet gives a definite advantage for cloud detection (over snow), compared to a regular FCN. Moreover, we see that SWIR plays a major role in the discrimination, as using it in an FCN (whether as FCN_{SWIR} or inside CloudSNet) gives nearly 91% of UA for Snow, and nearly 96% of PA for Clouds. FCN_{VNIR} on the other hand, gives the highest PA for Snow; which means that fetching out bright pixels of snow from the three VNIR bands is beneficial than fetching out dark pixels from SWIR.

Table 7 further shows that FCN_{VNIR} gives the least UA of snow, which means that the snow pixels which it predicted were the least reliable among the three networks. The confusion matrix of FCN_{VNIR}, given in Table 9 shows that most of these false predictions occur on the actual cloud pixels (also seen in Figure 28). This is because clouds have the same reflectance values as snow in the VNIR bands. Hence, using CloudSNet becomes beneficial as it reduces the misclassification of snow, by incorporating SWIR.

Table 9: Confusion Matrix of FCN_{VNIR}. Approximately 98% of the false predictions for snow occur on true-cloud pixels (Columns show the actual classes, while the rows depict the predicted classes)

Class	Clouds	Snow	Shadows	Rest
Clouds	2269877	42077	40151	5395
Snow	417752	1106632	8165	1049
Shadows	17748	35943	231986	1709
Rest	34140	32203	16942	401153

Coming to the minor metrics in Table 8, we see that FCN_{VNIR} is the most beneficial in classifying the 'Rest' class. This is because the 'Rest' pixels show the least reflectance in Red and Green bands, easily contrasting with the bright neighbouring pixels of snow or clouds, and helping the CNN, in their classification. For this class, both FCN_{SWIR} and CloudSNet show a high misclassification rate of Clouds as 'Rest', and vice-versa (Table 10 and Table11). This is because 'Rest' pixels have significantly high reflectance values in SWIR, almost the same as that of thin clouds leading to their misclassification (as shown in Figure 29). Moreover, CloudSNet is better able to predict the Shadows, as the dark pixels from SWIR also add to the information, making the predictions more accurate. The network is also able to give a higher overall accuracy, showing that it was the most successful in segmenting the data semantically.

Table 10: Confusion Matrix of FCN_{SWIR}.
Columns show the actual classes, while the rows depict the predicted classes

Class	Clouds	Snow	Shadows	Rest
Clouds	2597165	345	3312	47655
Snow	11472	1057466	124332	14277
Shadows	2893	109446	143715	1392
Rest	127987	49598	25885	345982

Table 11: Confusion Matrix of CloudSNet.
Columns show the actual classes, while the rows depict the predicted classes

Class	Clouds	Snow	Shadows	Rest
Clouds	2654499	5829	13718	22335
Snow	45016	1100082	16163	64
Shadows	14022	46419	256372	113
Rest	25980	64525	10991	386794

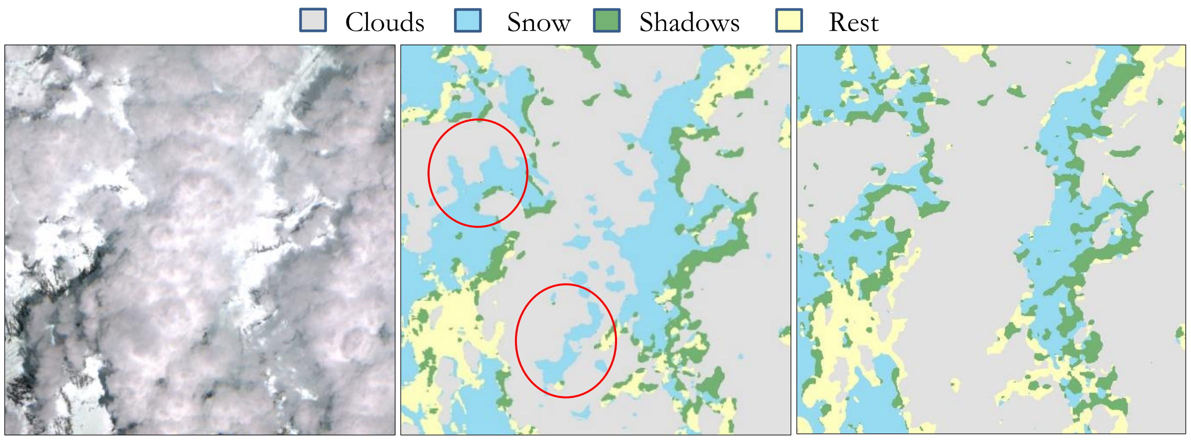


Figure 28: False predictions by FCN_{VNIR}. From left – original Tile No. 4, its FCN_{VNIR} classification and CloudSNet classification. FCN_{VNIR} falsely classifies some of the bright regions as snow (marked in red), which were actually clouds.

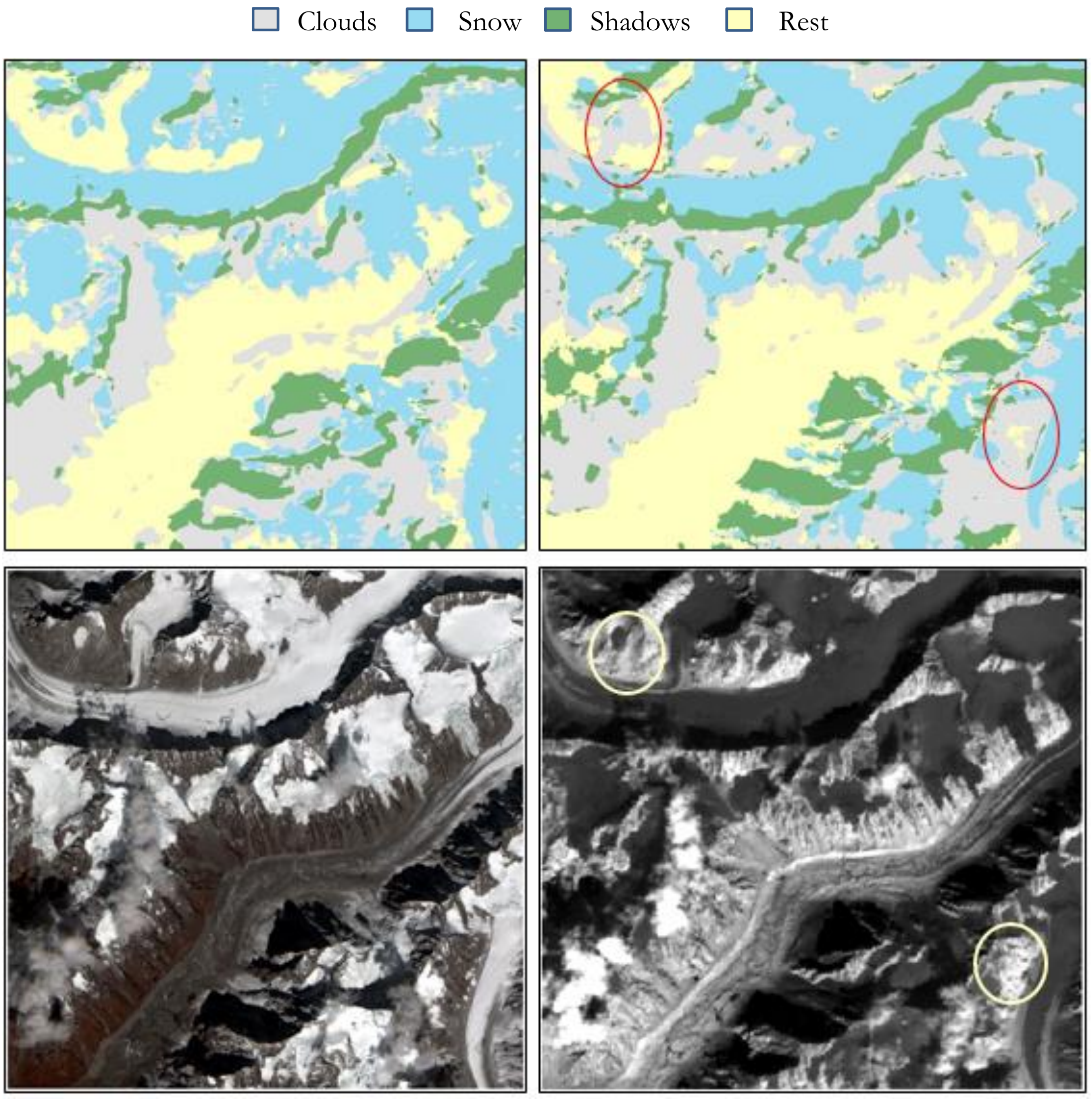


Figure 29: False predictions by CloudSNet. Clockwise from above, left – original tile no. 2, FCN_{VNIR} classification, CloudSNet classification and the corresponding SWIR band for the tile. 'Rest' regions marked in yellow have significantly high reflectance values in SWIR. Result being, the marked regions have been misclassified as clouds by CloudSNet (highlighted in red), which is not the case for FCN_{VNIR}.

Thus, we see that fusing a medium resolution SWIR band to high resolution optical VNIR images can bring an advantage in segregating clouds from snow. Working with these set of resolutions separately cannot achieve the task as effectively as their fused product can. Our network is still far from perfect, as a lot of 'Rest' regions are being misclassified as clouds. This means that the VFB was not able to learn enough spectral information from the VNIR bands, to be able to segregate the 'Rest' regions properly. Such an issue can only be rectified by training the network on more, diverse dataset and optimizing the network parameters in the VFB.

6.2.2. Comparison with Cloud and Shadow Mask Generator

The Cloud and Shadow Mask Generator (CSMG) for RS-2 takes only spectral information into account, and lacks in exploiting the spatial, contextual information of the image. Hence, comparing the utility's output with our proposed network architecture helps in appreciating the advantage CNNs bring in semantic segmentation of remote sensing images. Figure 34 shows the comparative performance of $CloudSNet_2$ and CSMG software.

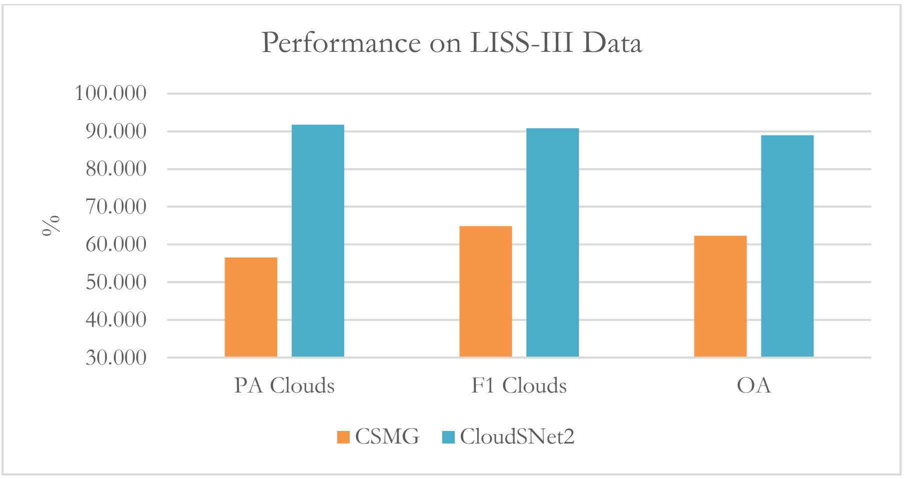


Figure 30: PA and F1 Score of Clouds, along with the overall accuracy across all the LISS-III tiles

From the figure above, we see that CloudSNet$_2$ is beneficial for cloud detection. Moreover, the maps predicted by the software tool are quite 'blocky' in nature, whereas those predicted by CloudSNet$_2$ are seamless (Figure 35). Figure 35 shows the classification only for Tile 1, but the 'blockiness' was observed across all the tiles, as shown in the Appendix.

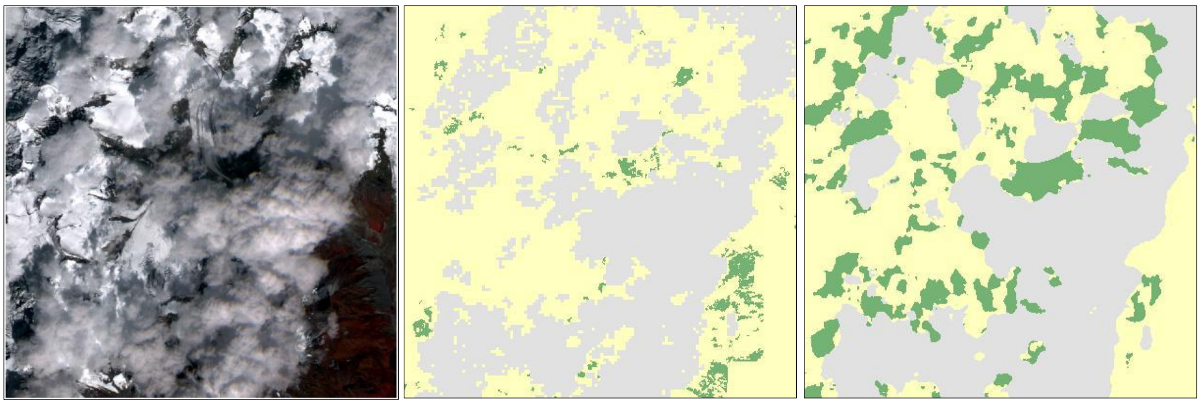


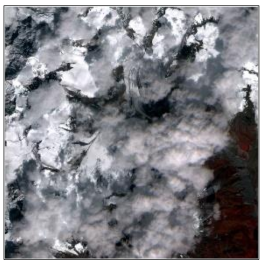
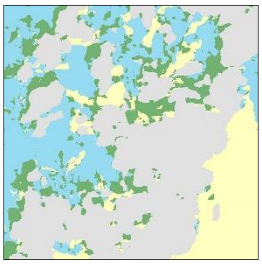
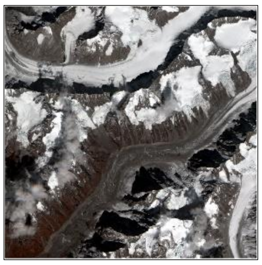
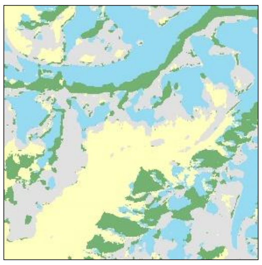
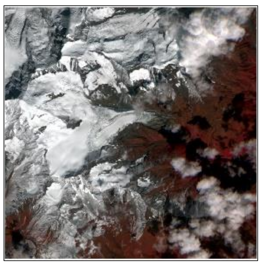
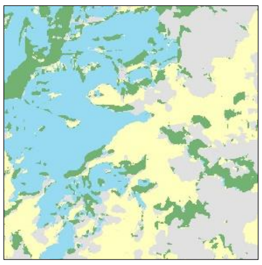
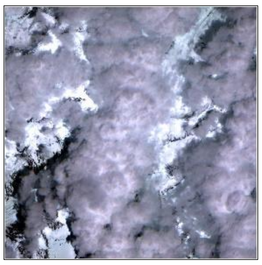
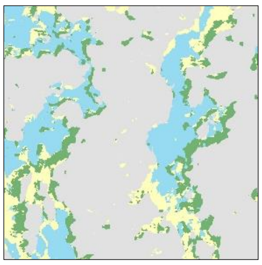
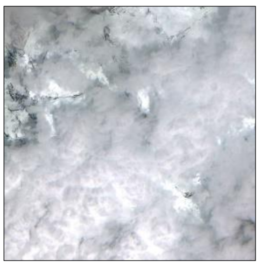
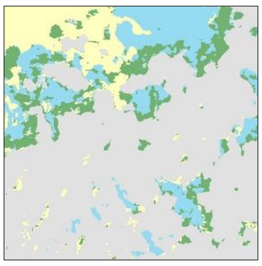
Figure 31: Segmentation of Tile 1 into three classes. From left – LISS-III False Colour Composite, output from the CSMG, and output from CloudSNet$_2$. The pre-built tool is able to detect most of the cloud regions correctly, but the transition between classes is not smooth resulting in sharp edged boundaries and a 'blocky' look. On the other hand, CloudSNet$_2$ is able to understand the contextual information of the pixels, giving a more realistic picture with a higher accuracy.

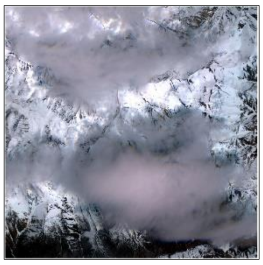
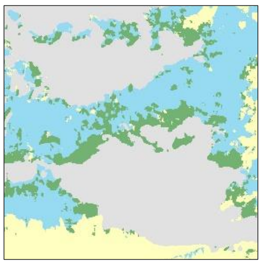
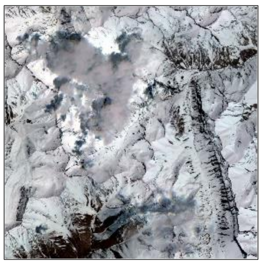
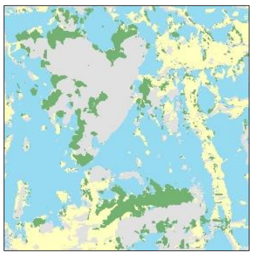
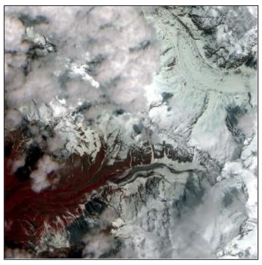
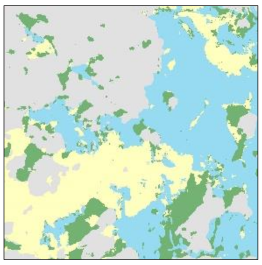
Thus we see that, the supervised classification by our neural network, and CNNs in general, was able to learn enough spatio-contextual information which a traditional, unsupervised classification algorithm could not.

6.3. Area Estimation

Finally, we computed the Snow Covered Area (SCA) and calculated the cloud fraction percentage for every tile. The classification was done on LISS-IV, through CloudSNet, and the area estimates are shown in Table 12. Note that the SCA computed will be for the 'visible' snow, i.e. we do not know yet how much snow actually lies underneath the clouds. In such a manner, the proposed network can be used for calculating the cloud fraction and creating a cloud mask for other satellite sensors as well.

Table 12: Cloud Fraction Percentage and SCA of each tile, computed through CloudSNet classification

Tile No.	LISS-IV FCC	Classified Output	Cloud Fraction (%)	SCA (km^2)
1			46.01	24.45
2			29.89	29.82
3			26.68	31.84
4			62.56	17.85
5			62.67	15.83

6			46.46	27.31
7			25.30	45.12
8			33.90	31.65

⬜ Clouds 🟦 Snow 🟩 Shadows 🟨 Rest

7. CONCLUSION

Our study proposed a novel way of introducing a SWIR band to a high resolution optical sensor. The procedure of resampling and fusion, inherited inside the learning of a convolutional neural network gave high classification accuracies for snow and clouds. Such a multi resolution fusion based model performed significantly higher on most of the metrics, as compared to a standalone FCN on independent bands. It was further found that our deep fully convolutional network was able to learn spectro-contextual information better than the traditional methods, helping in the semantic segmentation of spatial data. Thus, the weights from such a network could be adapted to create cloud masks as a part of Level-2 products from a satellite sensor.

Although our deep network lacked in some measures, such as misclassifying thin clouds, this could be rectified by introducing a cirrus band or by training the network on an extensive amount of images. The model still requires a variety of training data covering diverse topologies, tonal variations, seasons, cloud texture etc, which would improve the model's generalization ability. Also, as the methodology involved creating visually labelled reference data, there lies a great scope of human error. A limitation of using our methodology is that as CNNs can have only integral strides, the resampling of different resolutions can only happen when they are an integral multiple of each other. If this hadn't been the case, we would have directly used the original LISS-IV VNIR bands at 24m resolution with a LISS-III SWIR at 5m resolution. Having said that, we now answer the Research Questions, posed in Section 1.4 as follows:

Q.1. What is the classification accuracy obtained by the network? How can the network be improved to increase the accuracy?

The initially developed baseline architecture achieved classification accuracies of approximately 90%, on an average. We improved the architecture by carrying out experiments on its filter size, downsampling operations and fusion strategies. It was found that a filter of size 5×5 was optimum, while fusing at the higher resolution was effective. Employing transposed convolutions for feature extraction from the SWIR was found to be helpful, but it involved a lot of computation time. We built the optimum architecture using a combination of the above strategies and obtained ~95% class accuracies. The network could be further improved by training it with a variety of images, and assessing the effect of parameter changes. This can be done by varying spatial window sizes, increasing the number of feature maps, etc. Also, expert labellers could be used to make accurate reference maps, which would help in an improved training of the model.

Q.2. Was there any advantage in introducing and fusing a Shortwave Infrared band?

Yes. Introducing SWIR to a fully convolutional VNIR-based model was found to be advantageous. This was because the network could then learn enough spectro-contextual information which would help discriminate clouds and snow. Introducing the SWIR significantly reduced false predictions of snow (by nearly 20%) and increased the true predictions for clouds (by nearly 15%). This could not have been achieved by an FCN-on-SWIR, as it cannot detect most of the snow correctly. The proposed architecture could further be applied and tested on other multi resolution fusion problems as well.

Q.3. Does the proposed network perform better than traditional techniques? Are the extensive computations involved in the proposed network justified?

The proposed network gives significantly higher classification accuracies, compared to traditional techniques used by prominent cloud mask utilities. Also, the fully convolutional approach achieved a realistic semantic segmentation of the image which cannot be observed through traditional techniques (due to their lack of incorporating spatial context). Although the network took more than an hour to learn from four million pixels on a high performance computer; the computations are worth it, as they prepare an efficient classifier robust enough to segregate clouds from snow in multiple satellite images. Moreover, training is a one-time effort; and once a network is fully trained on a particular type of image, it can readily be used to classify similar images.

Recommendations

The proposed methodology could also be adopted for other optical sensors such as the Multispectral Instrument (MSI) available on-board Sentinel-2, and compared with its native Cloud Mask product. As CNNs require an extensive amount of training data, the deep network for Sentinel-2 can be trained and validated by publicly available labelled datasets, such as the one by Hollstein, Segl, Guanter, Brell, & Enesco (2016). Furthermore, the performance of such a trained network could also be tested against the state-of-the-art Fmask algorithm, and with other machine learning classifiers such as Random Forests and Support Vector Machines. An advantage of using the proposed method on Sentinel is that it has a SWIR at 20m and the VNIR bands at 10m. As these resolutions are integral multiples of each other, they wouldn't require a separate interpolation step as the one described in Section 5.3.1.

Another novel way of approaching our problem could be through constructing a Generative Adversarial Network (Choe, Park, & Shim, 2018; Radford, Metz, & Chintala, 2015), where the SWIR band could be used as the Generator and the VNIR bands as the Discriminator. This would be based upon unsupervised learning, resulting in the classification of VNIR bands. As having the actual 'ground' truth is nearly impossible in our case, using such an unsupervised method would be ideal. Furthermore, after detecting clouds, multi-temporal images could be used to mask them out. This would assist in assessing the snow underneath the clouds and help in creating accurate snow cover maps, supporting climate change studies.

.

LIST OF REFERENCES

Ben Driss, S., Soua, M., Kachouri, R., & Akil, M. (2017). A comparison study between MLP and convolutional neural network models for character recognition. In N. Kehtarnavaz & M. F. Carlsohn (Eds.), *SPIE Conference on Real-Time Image and Video Processing* (Vol. 10223, p. 1022306). Anaheim, CA, United States. https://doi.org/10.1117/12.2262589

Bergado, J. R., Persello, C., & Stein, A. (2018). Recurrent Multiresolution Convolutional Networks for VHR Image Classification. *IEEE Transactions on Geoscience and Remote Sensing*, *56*(11), 6361–6374. https://doi.org/10.1109/TGRS.2018.2837357

Birajdar, F., Venkataraman, G., & Samant, H. (2016). Monitoring Snow Cover Area Using Different Algorithms on Indian Remote Sensing Data. In N. J. Raju (Ed.), *Geostatistical and Geospatial Approaches for the Characterization of Natural Resources in the Environment* (pp. 749–753). Cham: Springer International Publishing. https://doi.org/10.1007/978-3-319-18663-4_115

Buhler, Y., Meier, L., & Ginzler, C. (2015). Potential of Operational High Spatial Resolution Near-Infrared Remote Sensing Instruments for Snow Surface Type Mapping. *IEEE Geoscience and Remote Sensing Letters*, *12*(4), 821–825. https://doi.org/10.1109/LGRS.2014.2363237

Choe, J., Park, J. H., & Shim, H. (2018). Generative Adversarial Networks for Unsupervised Object Co-localization. Retrieved from http://arxiv.org/abs/1806.00236

Convolutional Neural Networks - Basics. (2017). Retrieved from https://mlnotebook.github.io/post/CNN1/

Cozzolino, D., Martino, G. Di, Poggi, G., & Verdoliva, L. (2017). A fully convolutional neural network for low-complexity single-stage ship detection in Sentinel-1 SAR images. In *2017 IEEE International Geoscience and Remote Sensing Symposium (IGARSS)* (pp. 886–889). https://doi.org/10.1109/IGARSS.2017.8127094

Dumoulin, V., & Visin, F. (2016). A guide to convolution arithmetic for deep learning. Retrieved from http://arxiv.org/abs/1603.07285

European Space Agency. (2019). Sentinel-2 Cloud Mask Algorithm. Retrieved January 7, 2019, from https://sentinel.esa.int/web/sentinel/technical-guides/sentinel-2-msi/level-2a/algorithm

Gao, B.-C., Han, W., Tsay, S. C., & Larsen, N. F. (1998). Cloud Detection over the Arctic Region Using Airborne Imaging Spectrometer Data during the Daytime. *Journal of Applied Meteorology*, *37*(11), 1421–1429. https://doi.org/10.1175/1520-0450(1998)037<1421:CDOTAR>2.0.CO;2

Geng, J., Wang, H., Fan, J., & Ma, X. (2017). Deep Supervised and Contractive Neural Network for SAR Image Classification. *IEEE Transactions on Geoscience and Remote Sensing*, *55*(4), 2442–2459. https://doi.org/10.1109/TGRS.2016.2645226

Glorot, X., & Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In *JMLR W&CP: Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics (AISTATS 2010)* (Vol. 9, pp. 249–256).

Guirado, E., Tabik, S., Alcaraz-Segura, D., Cabello, J., & Herrera, F. (2017). Deep-learning Versus OBIA for Scattered Shrub Detection with Google Earth Imagery: Ziziphus lotus as Case Study. *Remote Sensing*, *9*(12), 1220. https://doi.org/10.3390/rs9121220

Hollstein, A., Segl, K., Guanter, L., Brell, M., & Enesco, M. (2016). Ready-to-Use Methods for the Detection of Clouds, Cirrus, Snow, Shadow, Water and Clear Sky Pixels in Sentinel-2 MSI Images. *Remote Sensing*, *8*(8), 666. https://doi.org/10.3390/rs8080666

Hughes, M., & Hayes, D. (2014). Automated Detection of Cloud and Cloud Shadow in Single-Date Landsat Imagery Using Neural Networks and Spatial Post-Processing. *Remote Sensing*, *6*(6), 4907–4926. https://doi.org/10.3390/rs6064907

Ioffe, S., & Szegedy, C. (2015). Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. Retrieved from http://arxiv.org/abs/1502.03167

Irish, R. R., Barker, J. L., Goward, S. N., & Arvidson, T. (2006). Characterization of the Landsat-7 ETM+ Automated Cloud-Cover Assessment (ACCA) Algorithm. *Photogrammetric Engineering & Remote Sensing*, *72*(10), 1179–1188. https://doi.org/10.14358/PERS.72.10.1179

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet Classification with Deep Convolutional Neural Networks. In *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1* (pp. 1097–1105). USA: Curran Associates Inc. Retrieved from http://dl.acm.org/citation.cfm?id=2999134.2999257

Kulkarni, A. V., Singh, S. K., Mathur, P., & Mishra, V. D. (2006). Algorithm to monitor snow cover using AWiFS data of RESOURCESAT-1 for the Himalayan region. *International Journal of Remote Sensing*, *27*(12), 2449–2457. https://doi.org/10.1080/01431160500497820

Le Goff, M., Tourneret, J.-Y., Wendt, H., Ortner, M., & Spigai, M. (2017). Deep Learning for Cloud Detection. In *8th International Conference of Pattern Recognition Systems (ICPRS 2017)* (p. 10 (6 .)-10 (6 .)). Institution of Engineering and Technology. https://doi.org/10.1049/cp.2017.0139

Li, Y., Wang, J., Xu, Y., Li, H., Miao, Z., & Zhang, Y. (2017). DeepSAR-Net: Deep convolutional neural networks for SAR target recognition. In *2017 IEEE 2nd International Conference on Big Data Analysis (ICBDA)(* (pp. 740–743). https://doi.org/10.1109/ICBDA.2017.8078734

Lyapustin, A., Wang, Y., & Frey, R. (2008). An automatic cloud mask algorithm based on time series of MODIS measurements. *Journal of Geophysical Research*, *113*(D16), D16207. https://doi.org/10.1029/2007JD009641

Maggiori, E., Tarabalka, Y., Charpiat, G., & Alliez, P. (2017). Convolutional Neural Networks for Large-Scale Remote-Sensing Image Classification. *IEEE Transactions on Geoscience and Remote Sensing*, *55*(2), 645–657. https://doi.org/10.1109/TGRS.2016.2612821

Man, Q. X., Guo, H. D., Liu, G., & Dong, P. L. (2014). Comparison of different methods for monitoring glacier changes observed by Landsat images. *IOP Conference Series: Earth and Environmental Science*, *17*(1), 012127. https://doi.org/10.1088/1755-1315/17/1/012127

Mateo-García, G., Gómez-Chova, L., & Camps-Valls, G. (2017). Convolutional neural networks for multispectral image cloud masking. In *2017 IEEE International Geoscience and Remote Sensing Symposium (IGARSS)* (pp. 2255–2258). https://doi.org/10.1109/IGARSS.2017.8127438

Maurer, E. P., Rhoads, J. D., Dubayah, R. O., & Lettenmaier, D. P. (2003). Evaluation of the snow-covered area data product from MODIS. *Hydrological Processes*, *17*(1), 59–71. https://doi.org/10.1002/hyp.1193

Miller, S. D., Lee, T. F., & Fennimore, R. L. (2005). Satellite-Based Imagery Techniques for Daytime Cloud/Snow Delineation from MODIS. *Journal of Applied Meteorology*, *44*(7), 987–997. https://doi.org/10.1175/JAM2252.1

Mohajerani, S., Krammer, T. A., & Saeedi, P. (2018). Cloud Detection Algorithm for Remote Sensing Images Using Fully Convolutional Neural Networks. Retrieved from http://arxiv.org/abs/1810.05782

Mullissa, A. G., Persello, C., & Tolpekin, V. (2018). Fully Convolutional Networks for Multi-Temporal SAR Image Classification. In *IGARSS 2018 - 2018 IEEE International Geoscience and Remote Sensing Symposium* (pp. 6635–6638). https://doi.org/10.1109/IGARSS.2018.8518780

National Remote Sensing Centre. (2003). *Resourcesat-2 Handbook*. Hyderabad. Retrieved from http://lps16.esa.int/posterfiles/paper1213/%5BRD13%5D_Resourcesat-2_Handbook.pdf

National Remote Sensing Centre. (2017). *User Manual - SWIR Band synthesis Utility for IRS RS2 L4Mx Data.* Hyderabad. Retrieved from http://www.nrsc.gov.in/Satellite_Data_Products_Overview?q=Download_Softwares_1

National Snow and Ice Data Center. (2017). Snow and Climate | National Snow and Ice Data Center. Retrieved June 7, 2018, from https://nsidc.org/cryosphere/snow/climate.html

Nikam, B. R., Garg, V., Gupta, P. K., Thakur, P. K., Senthil Kumar, A., Chouksey, A., … Purohit, S. (2017). Satellite-based mapping and monitoring of heavy snowfall in North Western Himalaya and its hydrologic consequences. *Current Science*, *113*(12), 2328–2334. https://doi.org/10.18520/cs/v113/i12/2328-2334

Perceptron. (2014). Retrieved from http://neuroph.sourceforge.net/tutorials/Perceptron.html

Persello, C., & Stein, A. (2017). Deep Fully Convolutional Networks for the Detection of Informal Settlements in VHR Images. *IEEE Geoscience and Remote Sensing Letters*, *14*(12), 2325–2329. https://doi.org/10.1109/LGRS.2017.2763738

Radford, A., Metz, L., & Chintala, S. (2015). Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. Retrieved from http://arxiv.org/abs/1511.06434

Rango, A. (1993). II. Snow hydrology processes and remote sensing. *Hydrological Processes*, *7*(2), 121–138. https://doi.org/10.1002/hyp.3360070204

Raschka, S. (2015). Single-Layer Neural Networks and Gradient Descent. Retrieved from https://sebastianraschka.com/Articles/2015_singlelayer_neurons.html

Rawat, M. (2018, March 25). Uttarakhand glaciers more sensitive to climate change, says study. *Hindustan Times*. Retrieved from https://www.hindustantimes.com/india-news/uttarakhand-glaciers-more-sensitive-to-climate-change-says-study/story-za80Fw1BwH4LbYjj7DHNjL.html

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, *323*(6088), 533–536. https://doi.org/10.1038/323533a0

Santos, L. A. (2019). Convolutional Neural Networks. Retrieved December 26, 2018, from https://leonardoaraujosantos.gitbooks.io/artificial-inteligence/content/convolutional_neural_networks.html

Savchenkov, A., Davis, A., & Zhao, X. (2017). Generalized Convolutional Neural Networks for Point Cloud Data. Retrieved from http://arxiv.org/abs/1707.06719

Shelhamer, E., Long, J., & Darrell, T. (2017). Fully Convolutional Networks for Semantic Segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *39*(4), 640–651. https://doi.org/10.1109/TPAMI.2016.2572683

Srinivasulu, J., & Kulkarni, A. V. (2004). Estimation of spectral reflectance of snow from IRS-1D LISS-III sensor over the Himalayan terrain. *Journal of Earth System Science*, *113*(1), 117–128. https://doi.org/10.1007/BF02702003

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, *15*, 1929–1958. Retrieved from http://jmlr.org/papers/v15/srivastava14a.html

Tang, B.-H., Shrestha, B., Li, Z.-L., Liu, G., Ouyang, H., Gurung, D. R., … Khun San, A. (2010). Improvement of MODIS snow cover algorithm for the Hindu Kush-Himalayan region. In *2010 IEEE International Geoscience and Remote Sensing Symposium* (pp. 1737–1740). IEEE. https://doi.org/10.1109/IGARSS.2010.5651098

Tekeli, A. E., Sönmez, I., & Erdi, E. (2016). Snow-covered area determination based on satellite-derived probabilistic snow cover maps. *Arabian Journal of Geosciences*, *9*(3), 198. https://doi.org/10.1007/s12517-015-2149-0

Templeton, G. (2015). Artificial neural networks are changing the world. What are they? Retrieved from http://www.extremetech.com/extreme/215170-artificial-neural-networks-are-changing-the-world-what-are-they

Uttarakhand. (2017). Retrieved February 9, 2019, from http://www.nainitaltourism.com/Uttarakhand_Uttaranchal.html

Wilson, A. C., Roelofs, R., Stern, M., Srebro, N., & Recht, B. (2018). The Marginal Value of Adaptive Gradient Methods in Machine Learning. Retrieved from https://arxiv.org/abs/1705.08292v2

Wiratama, W., Lee, J., Park, S.-E., & Sim, D. (2018). Dual-Dense Convolution Network for Change Detection of High-Resolution Panchromatic Imagery. *Applied Sciences*, *8*(10), 1785. https://doi.org/10.3390/app8101785

Yang, Z., Jiang, W., Xu, B., Zhu, Q., Jiang, S., & Huang, W. (2017). A Convolutional Neural Network-Based 3D Semantic Labeling Method for ALS Point Clouds. *Remote Sensing*, *9*(9), 936. https://doi.org/10.3390/rs9090936

Zhan, Y., Wang, J., Shi, J., Cheng, G., Yao, L., & Sun, W. (2017). Distinguishing Cloud and Snow in Satellite Images via Deep Convolutional Network. *IEEE Geoscience and Remote Sensing Letters*, *14*(10), 1785–1789. https://doi.org/10.1109/LGRS.2017.2735801

Zhang, P., Niu, X., Dou, Y., & Xia, F. (2017). Airport Detection on Optical Satellite Images Using Deep Convolutional Neural Networks. *IEEE Geoscience and Remote Sensing Letters*, *14*(8), 1183–1187. https://doi.org/10.1109/LGRS.2017.2673118

Zhu, X., & Helmer, E. H. (2018). An automatic method for screening clouds and cloud shadows in optical satellite image time series in cloudy regions. *Remote Sensing of Environment*, *214*, 135–153. https://doi.org/10.1016/j.rse.2018.05.024

Zhu, X. X., Tuia, D., Mou, L., Xia, G.-S., Zhang, L., Xu, F., & Fraundorfer, F. (2017). Deep Learning in Remote Sensing: A Comprehensive Review and List of Resources. *IEEE Geoscience and Remote Sensing Magazine*, *5*(4), 8–36. https://doi.org/10.1109/MGRS.2017.2762307

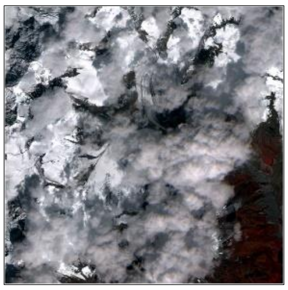
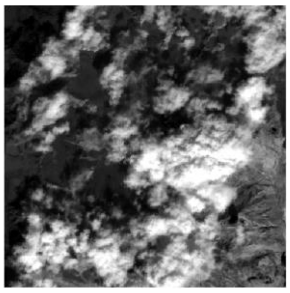
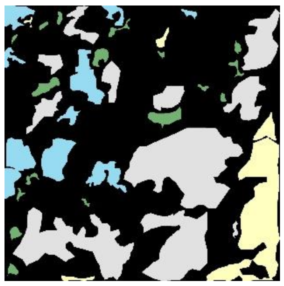
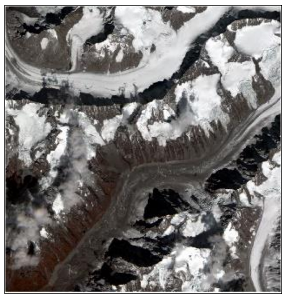
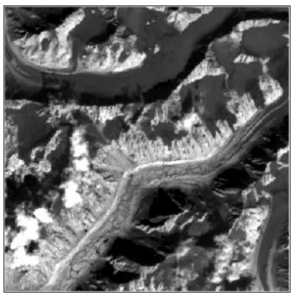
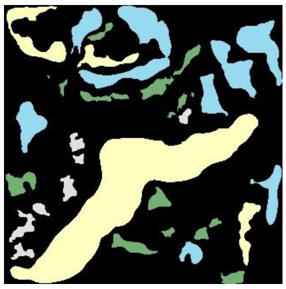
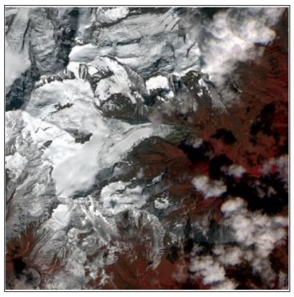
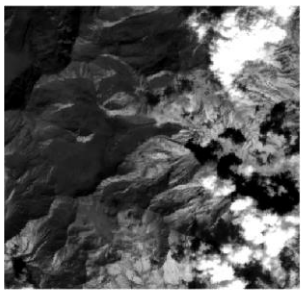
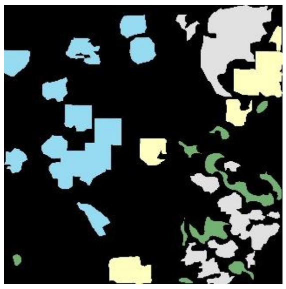
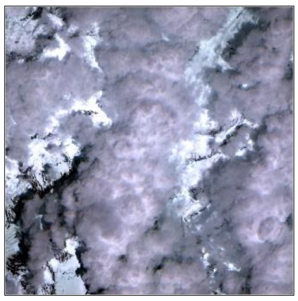
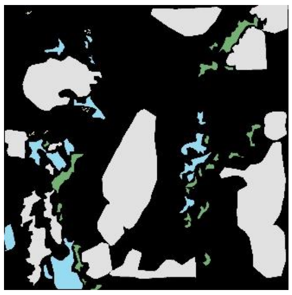
Zhu, Z., Wang, S., & Woodcock, C. E. (2015). Improvement and expansion of the Fmask algorithm: cloud, cloud shadow, and snow detection for Landsats 4–7, 8, and Sentinel 2 images. *Remote Sensing of Environment*, *159*, 269–277. https://doi.org/10.1016/J.RSE.2014.12.014

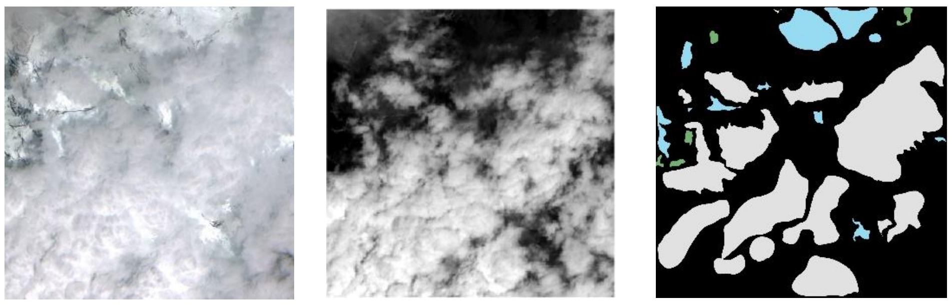
Zhu, Z., & Woodcock, C. E. (2012). Object-based cloud and cloud shadow detection in Landsat imagery. *Remote Sensing of Environment*, *118*, 83–94. https://doi.org/10.1016/j.rse.2011.10.028

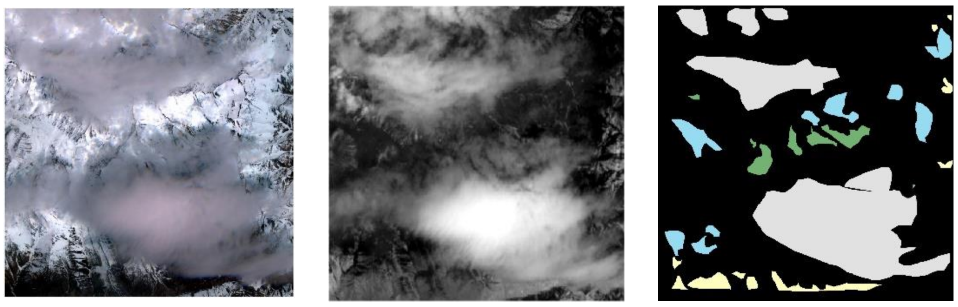
APPENDIX

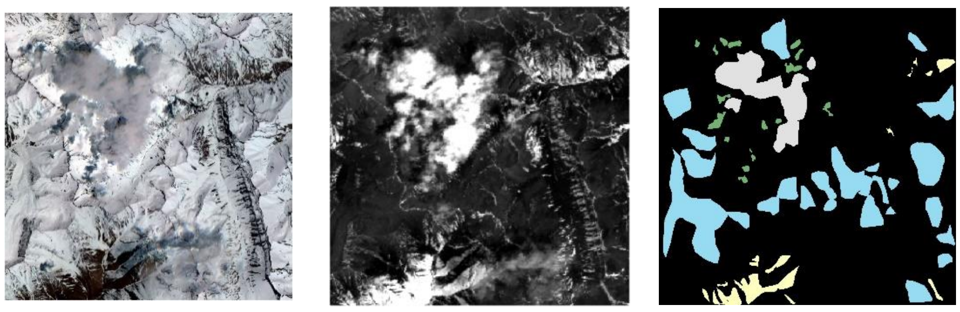
DATASET

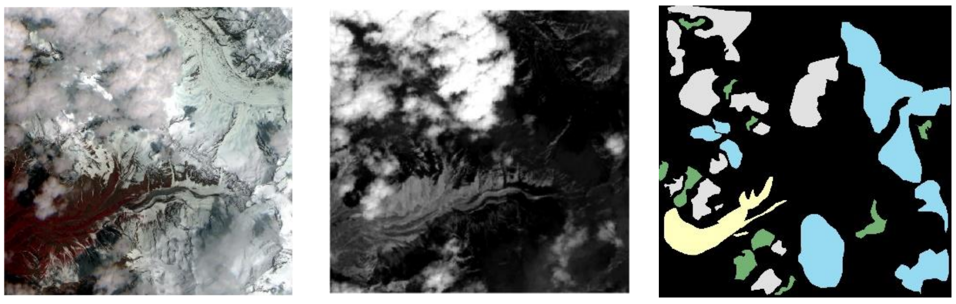
Date	Product Id
9-Oct-2014	1853386251
13-May-2015	1853386181

Tile No.	FCC	SWIR	Reference Map*	No. of Pixels per Class
1				CL 902575 SN 239900 SH 78684 R 261527
2				CL 64277 SN 398185 SH 142396 R 686294
3				CL 352329 SN 364082 SH 108788 R 203871
4				CL 1002073 SN 120689 SH 69046 R 1613

5

CL 1208895
SN 173392
SH 19056
R 476

6

CL 815762
SN 140160
SH 65186
R 74340

7

CL 161960
SN 535707
SH 36329
R 73849

8

CL 362531
SN 539221
SH 104214
R 130619

* in 5m×5m resolution

⬜ Clouds (CL) 🟦 Snow (SN) 🟩 Shadows (SH) 🟨 Rest (R)

INTERMEDIATE NETWORKS

	PA Clouds	PA Snow	UA Clouds	UA Snow	OA	F1 Clouds	F1 Snow	Average F1
Fuse1$_{max}$	96.806	97.014	94.234	97.234	96.910	95.711	96.310	94.871
Fuse1$_{avg}$	97.725	96.556	94.305	97.187	97.137	95.724	96.431	95.007
Fuse1$_{conv}$	97.450	96.900	93.804	96.942	97.174	95.347	96.261	95.135

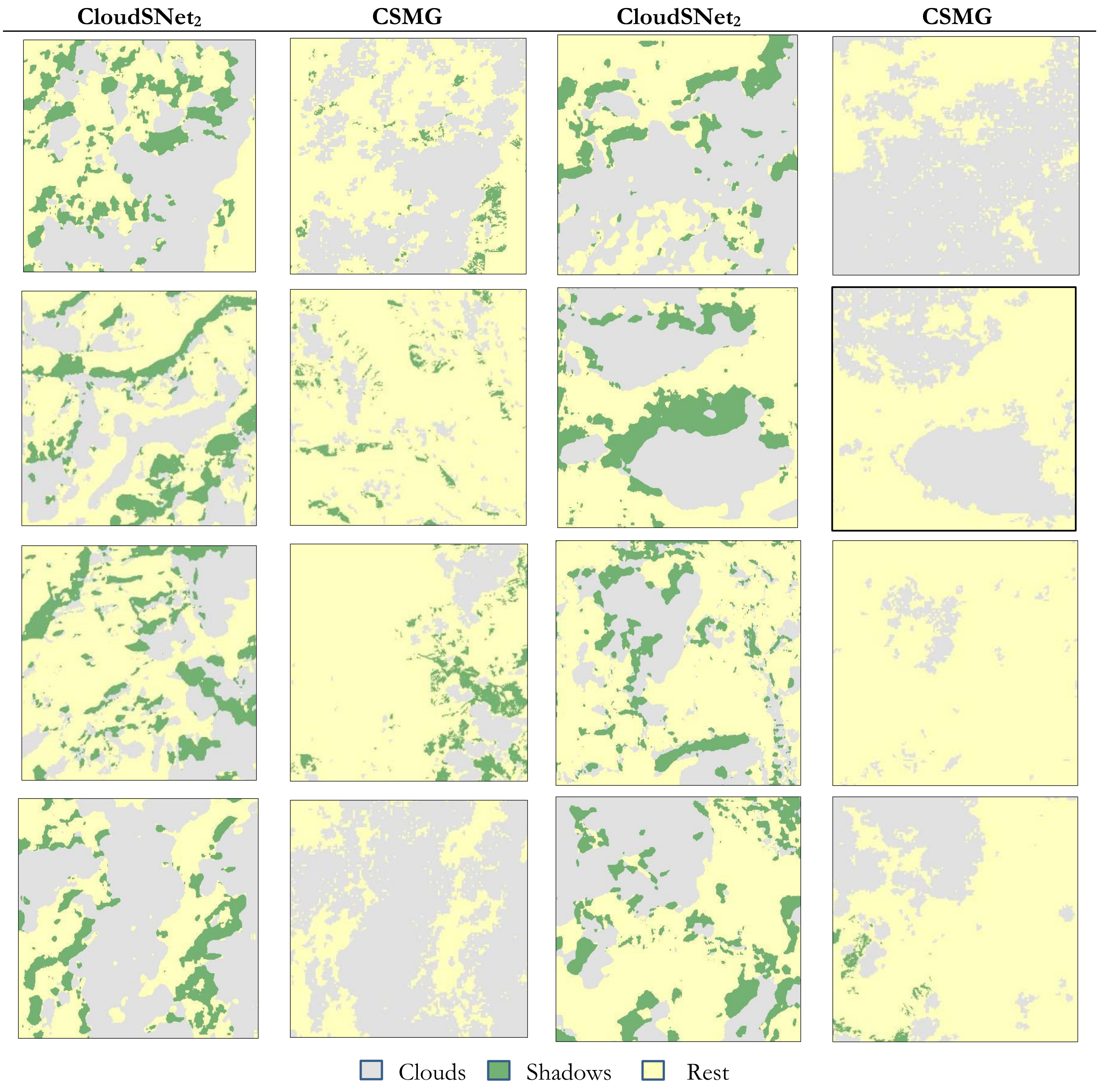
Fuse6	**Fuse7**
SWIR	VNIR
Conv1-1-16	Conv9-1-8
TConv4-2-1-16	maxpool
TConv4-2-1-16	Conv9-1-16
$\oplus$ VNIR	Maxpool
Conv5-1-64	$\oplus$ SWIR
Conv5-2-64	Conv5-1-64
Conv1-1-4	Conv5-2-64
	TConv4-2-1-64
	TConv4-2-1-64
	Conv1-1-4

	PA Clouds	PA Snow	UA Clouds	UA Snow	OA	F1 Clouds	F1 Snow	Average F1
Fuse1	96.806	94.234	97.014	97.234	94.871	96.910	95.711	96.310
Fuse2	95.232	97.176	93.456	94.375	96.194	93.913	95.054	93.734
Fuse6	97.581	95.490	91.472	95.533	91.364	94.428	95.511	94.970
Fuse7	96.747	94.428	97.184	96.773	95.026	96.965	95.587	96.276

	PA Clouds	PA Snow	UA Clouds	UA Snow	OA	F1 Clouds	F1 Snow	Average F1
3x3	96.360	95.455	93.502	97.150	95.905	95.291	95.598	93.281
5×5	97.724	96.556	94.304	97.186	97.136	95.723	96.430	95.006
7×7	97.283	96.519	93.363	96.080	96.899	94.702	95.801	94.726
9×9	96.806	97.014	94.234	97.234	96.910	95.711	96.310	94.871
11×11	97.491	96.848	93.210	96.311	97.168	94.735	95.952	94.946
13×13	97.740	96.754	93.952	96.702	97.244	95.307	96.276	95.095

CLASSIFICATION INTO CLOUDS, SHADOWS, AND REST

	PA Clouds	UA Clouds	F1 Clouds	PA Shadows	UA Shadows	PA Rest	UA Rest	OA
CSMG	56.559	76.037	64.867	4.229	12.995	79.469	54.814	62.311
CloudSNet$_2$	91.800	89.904	90.842	90.910	82.190	84.789	88.816	88.922



CloudSNet2
CSMG
CloudSNet2
CSMG
Clouds
Shadows
Rest