# Applying Rotation-Equivariant Deep Learning to Cloud and Road Segmentation in Satellite and Aerial Imagery

by

Alex Meredith

S.B., Massachusetts Institute of Technology (2021)

Submitted to the Department of Aeronautics and Astronautics
in partial fulfillment of the requirements for the degree of

Master of Science in Aeronautics and Astronautics

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

January 2023

Authored by . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
Alex Meredith
Department of Aeronautics and Astronautics
January 31, 2023

Certified by . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
Kerri L. Cahoy
Associate Professor and Bisplinghoff Faculty Fellow
Thesis Supervisor

Accepted by . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
Jonathan P. How
R. C. Maclaurin Professor of Aeronautics and Astronautics
Chair, Graduate Program Committee

# Applying Rotation-Equivariant Deep Learning to Cloud and Road Segmentation in Satellite and Aerial Imagery

by

Alex Meredith

Submitted to the Department of Aeronautics and Astronautics
on January 31, 2023, in partial fulfillment of the
requirements for the degree of
Master of Science in Aeronautics and Astronautics

## Abstract

Satellite and aerial images have many applications – images of clouds and roads are of particular relevance to this thesis. Satellite images of clouds are frequently used for climate monitoring, weather tracking, satellite instrument calibration, and on-orbit autonomy; satellite and aerial images of roads are frequently used for mapping flooded areas, predicting illegal logging, traffic monitoring, and route planning.

Cloud detection in satellite imagery is key for autonomously taking and downlinking cloud-free images of a target region as well as studying cloud-climate interactions and calibrating microwave radiometers. Many existing state-of-the-art cloud detection algorithms require multispectral inputs and sometimes confuse clouds with snow, ice, or cold water. We propose deep learning models trained on visible-spectrum, long-wave infrared (LWIR), and short-wave infrared (SWIR) imagery for on-orbit cloud detection. Rotation-equivariant deep learning models are *equivariant* to rotations, meaning that when an input to the model is rotated, the model output will be equivalently rotated. We compare rotation-equivariant deep learning models to non-equivariant models, and also present comparisons to rule-based methods for cloud segmentation. Additionally, we compare models trained on visible-spectrum (VIS), LWIR, and SWIR imagery to models trained on only VIS and LWIR, on only VIS and SWIR, and on only VIS imagery and make recommendations for imaging bands to prioritize during instrument selection for resource-constrained missions. We find that augmenting VIS imagery with SWIR imagery is most useful for missions where false positives (non-cloud pixels misidentified as cloud) are extremely costly, and we find that augmenting with LWIR imagery is most useful for missions where false negatives (cloud pixels misidentified as non-cloud) are extremely costly.

A secondary focus of this thesis is evaluating rotation-equivariant deep learning models on the road detection domain. Road detection in satellite and aerial imagery can map safe evacuation routes from areas affected by natural disaster or predict deforestation by identifying roads constructed for the purpose of illegal logging. We present the results of rotation-equivariant and non-equivariant models on road segmentation of aerial imagery, and make recommendations for integrating rotation-equivariance into current state-of-the-art road detection algorithms.

We find that our $C_8$-equivariant dense U-Net, a rotation-equivariant deep learning model, outperforms our other deep learning models on both cloud and road segmentation, and also outperforms rule-based algorithms on cloud segmentation. The $C_8$-equivariant dense U-Net achieves an $F_1$ score of 0.9806 on the cloud segmentation dataset when evaluated with a 2 pixel buffer at the cloud boundaries, and achieves an $F_1$ score of 0.9342 on the road segmentation dataset when evaluated with a 4 pixel buffer at the road boundaries.

Thesis Supervisor: Kerri L. Cahoy
Title: Associate Professor and Bisplinghoff Faculty Fellow

# Acknowledgments

First and foremost, I'd like to thank Prof. Kerri Cahoy for her incredible mentorship, support, wisdom, and unbelievably deep technical knowledge. She is truly the most brilliant, caring, and dedicated advisor I could ask for.

In my time in STAR Lab, I've had the great privilege of working on a variety of interesting projects with incredibly smart and dedicated people. In particular, I'm extremely grateful to Shreeyam Kacker for his friendship, support, knowledge of deep learning, and willingness to help me whenever I need it. I'm also extremely grateful to Patrick McKeen for taking me on as a UROP back in 2020, and for his mentorship and advice.

Thank you to all of my friends and collaborators, including but certainly not limited to: Shreeyam Kacker and Georges Labrèche for their work on on-orbit machine learning, Patrick McKeen, Joey Murphy, Prof. Paula do Vale Pereira, Joe Kusters, and Mason Black for their work on BeaverCube; Joe Kusters, Hannah Tomio, Shreeyam Kacker, and Violet Felt for their work on BeaverCube-2; and Lucy Halperin, Amelia Gagnon, Juliana Chew, Endrit Shehaj, Dr. Riley Fitzgerald, and Dr. Stephen Leroy for their work on GPS radio occultation and gravity waves. All of you have been so patient, kind, and welcoming to me. Thank you also to my officemates, past and present: Mary Dahl, Sophia Vlahakis, Juliana Chew, Ilaria Petracca, and Sammy Hasler, for keeping me company and keeping the Halloween decorations up year-round.

Last but certainly not least, I'm incredibly thankful to my family for their love and support, as well as my mom's powerful advocacy for women in STEM, my dad's excellent music taste, and Luke and Isaac's creative and hilarious sibling hijinks. Most of all, I'm grateful to Kimmy for her endless love and support, excellent hugs, and her ability to make me smile even on the worst days. Kimmy – the best, luckiest, and most important part of my time at MIT has been falling in love with you.

This work was supported by the National Science Foundation under grant GEO-1850089 and the NSF Graduate Research Fellowship Program under Grant No. 2141064.

Any opinions, findings, conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

# Contents

THIS PAGE INTENTIONALLY LEFT BLANK

# List of Figures

# List of Tables

THIS PAGE INTENTIONALLY LEFT BLANK

# Chapter 1

# Introduction

## 1.1 Motivation

Small satellites are popular tools for Earth observation. Satellite imagery can provide valuable insights about the Earth; for example, satellite imagery can be used to track deforestation, which is an application case of interest in this work [47]. Satellite images can be rotated arbitrarily about the camera boresight or telescope boresight. Rotation-equivariant image processing algorithms can leverage this effect to improve object detection or image segmentation performance. Rotation-equivariant deep learning models have already been used with some success for object detection [19] and image segmentation [37] on satellite imagery. This thesis focuses on two particular classification tasks: cloud detection and road detection, and evaluates several image processing algorithms, including rotation-equivariant deep learning algorithms, for each task.

On-orbit cloud detection is a critical enabler of satellite autonomy and high-throughput imaging pipelines. Identifying cloudy images on-orbit allows satellites to downlink only cloud-free images, optimizing the useful data throughput of a satellite [28]. Many satellite instruments require cloud masks for data analysis: microwave radiometers require cloud-free data for instrument calibration [16], and NASA's Clouds and the Earth's Radiant Energy System (CERES) instrument requires cloudy data to study cloud-climate interactions [36]. Achieving on-orbit cloud detection with a cohosted instrument allows satellites to downlink data with a concurrent and co-located cloud mask, improving data throughput and thus speeding up instrument calibration and data analysis.

Route planning is crucial for autonomous driving, and road extraction from satellite imagery can be used to quickly map areas affected by flooding and other natural disasters, as well as to map evacuation routes [14]. Roads have a distinct topology where segments are generally connected to form a larger network, often in the form of a geometrically regular grid, making road detection a good candidate domain on which the benefits of rotation-equivariant deep learning can be demonstrated. Road geometry has previously been successfully leveraged to learn the orientation of roads in order to predict road networks and refine masks with missing road segments [7] [5]. Improving road detection in satellite imagery with rotation-equivariant deep learning networks can help in applications such as mapping flooded areas in time-critical situations in order to aid evacuation efforts and tracking road construction to predict illegal deforestation.

## 1.2 Background

### 1.2.1 Deep Learning

Deep learning is a type of machine learning that uses multiple hidden layers in an artificial neural network to train a model relating an input to an output. Deep learning models have shown promising gains in a variety of domains, including image segmentation, and especially segmentation of biomedical imagery. Both biomedical images and satellite images can be arbitrarily rotated about the camera vector, unlike many other types of imagery such as from vehicular or marine platforms. This similar geometry means that many models designed for segmenting biomedical imagery in theory can easily be adapted to segment satellite imagery.

U-Nets and dense U-Nets are two deep learning architectures which have demonstrated strong performance on segmenting biomedical imagery [42] [18]. U-Nets are convolutional neural network (CNN)-based deep learning models originally designed for segmenting biomedical imagery [42]. U-Nets downsample an input to create a high-resolution encoding, then upsample to create an output mask, and include feed-forward connections between the downsampling and upsampling path to better localize features [42]. Dense models replace convolutions in CNNs with "dense blocks", which contain multiple convolutions connected by feed-forward operations, improving gradient and feature propagation throughout a network [23]. Dense U-Nets replace convolution layers in U-Nets with dense blocks, and have successfully outperformed U-Nets on removing artifacts from biomedical images constructed with 2D sparse photoacoustic tomography [18].

### Transformation-Equivariant Deep Learning

Equivariance is a symmetric form of invariance: when an input to a shift-equivariant deep learning model is shifted, the output of the model will be equivalently shifted. Although most operations in CNNs are equivariant to translational shifts, most layers in traditional CNNs are not rotationally equivariant. Commonly used strided operations, including max pooling, average pooling, and strided convolutions, are subject to aliasing and lose translational equivariance [53]. As a result, most CNNs and CNN-based deep learning models, including basic U-Nets and dense U-Nets, are not equivariant to translational or rotational shifts in input.

Translational equivariance can be achieved by replacing strided operations with densely evaluated operations followed by a strided blur, which effectively low-pass filters a signal before sampling, reducing aliasing and improving translational equivariance [53]. Using blurs for anti-aliasing improves performance on object detection and can be easily integrated into almost any deep learning model [53]. A translationally equivariant version of AlexNet, a CNN designed for object detection, showed an 0.39% increase in accuracy and a 5.13% increase in consistency over a non-equivariant version of AlexNet when evaluated on the ImageNet dataset [53]. Translationally equivariant U-Nets have also previously demonstrated strong performance on segmenting clouds in satellite imagery [28].

Partial rotational equivariance can be achieved using group equivariant CNNs, which exploit symmetry by generalizing convolution layers to "group convolutions". Group convolutions are equivariant to a specific group of transformations, often $C_8$ (the group of rotations by integer multiples of 45°) or other cyclic groups. Group equivariance generally improves performance and reduces the parameter space of deep learning models. By directly encoding symmetry in the model, the model only has to learn to detect features of interest in the training phase and does not have to learn symmetry, speeding up training [12]. $C_8$-equivariant and other partially rotationally equivariant models are particularly well-suited to domains where images may be arbitrarily rotated about the camera vector, such as biomedical and satellite imagery [17].

$C_{16}$-equivariant CNNs with anti-aliased strided operations for translational equivariance have demonstrated state-of-the-art performance on identifying handwritten digits from the popular MNIST dataset [51]. Similarly, $C_8$-equivariant dense U-Nets have shown state-of-the-art performance on segmenting biomedical imagery, despite the fact that the models considered implemented strided operations without anti-aliasing and thus were not fully translationally equivariant [17]. Other models equivariant to transformations under $C_8$ but not to translations have been successfully applied to object detection and segmentation for deforestation detection in satellite imagery, outperforming non-equivariant models [19] [37].

Weiler *et al.* experimented with $C_n$-equivariant CNNs for $n$ varying from 1 to 24, allowing the number of model parameters to increase with the number of orientation channels [51]. Weiler *et al.* found that performance generally improves as angular resolution increases, but that adding more than 12 to 16 orientation channels yields no extra benefit [51]. Graham *et al.* experimented with $C_4$-, $C_8$-, and $C_{12}$-equivariant CNNs, with the total number of model parameters held constant, and found that $C_8$-equivariant models perform best for most applications [17].

### 1.2.2 Cloud Detection

Cloud detection is a critical capability for weather and climate satellite missions and has historically been performed on the ground using downlinked satellite multispectral data and physics-driven rule-based methods. Examples of physics-driven rule based methods still evaluated on the ground include Fmask, which detects clouds, clear land, clear water, and cloud shadow in Landsat and Sentinel-2 data [55], the continuity MODIS-VIIRS cloud mask, which detects clouds in MODIS and VIIRS data [15], the MODIS cloud mask retrieved for the CERES mission, which detects clouds in MODIS data [44], and the GOES cloud mask, which detects clouds and probable clouds in the GOES advanced baseline imager (ABI) data [22]. All of these methods use physics-derived reflectance and brightness temperature thresholds in different bands of a multispectral data input to determine whether a pixel is cloudy or clear. These methods often rely on determining the surface terrain type based on spectral properties of the inputted data, and have different cloud thresholds depending on the underlying terrain.

Most satellite instruments that require cloud masks for data processing or instrument calibration use cloud masks derived from physics-driven rule-based methods and multispectral input. For example, the TROPICS mission requires cloud masks prior to calibration in order to identify clear-sky pixels used to calibrate radiances of its microwave instrument, and uses cloud masks from GOES-16 in this pre-calibration process [16]. Similarly, validation of TEMPEST-D calibration requires extracting clear-sky data over ocean [8]. Suomi-NPP requires cloud masks to geolocate data from its ATMS instrument and uses VIIRS cloud masks [54]. MicroMAS-2A, like TROPICS, required cloud masks for radiance calibration and used cloud masks from VIIRS onboard Suomi-NPP and VIIR onboard Fengyun-3C for calibration [13].

Other methods for cloud detection include rule-based methods driven by statistics rather than physics, like random tree and random forest detection. The `s2cloudless` algorithm is a gradient-boosted tree-based algorithm for Sentinel-2 imagery that improves upon Fmask by almost 10% – likely in part because Fmask is designed to work for Landsat or Sentinel-2, while `s2cloudless` is optimized for Sentinel-2 [56]. Some statistics-driven rule-based methods have been tested on-orbit. On EO-1, Bayesian thresholding (BT) and random forest algorithms were tested for cloud detection, and a rule-based novelty detection system was also tested on-orbit on EO-1 [48]. The random forest algorithm on EO-1 achieved 94.5% accuracy using a 5x5 kernel around each pixel and three visible-spectrum bands [48]. Rule-based methods have also been used onboard OPS-SAT for cloud detection [27]. OPS-SAT tested luminosity thresholding, a rule-based method similar to the Bayesian thresholding algorithm used aboard EO-1, and k-means segmentation onboard, and additionally tested a random forest algorithm on the OPS-SAT engineering model on the ground [27]. OPS-SAT has no multispectral sensors, so all methods tested used visible-spectrum imagery only.

Deep learning on visible-spectrum or multispectral data can also be used for cloud detection. OPS-SAT evaluated a U-Net-based deep learning algorithm on-orbit for cloud detection trained on visible-spectrum Landsat imagery, in addition to three different rule-based methods [27]. The U-Net outperformed all three rule-based methods, achieving a balanced accuracy of 77-89% over three test images; the random forest algorithm was the next-best performing method and achieved a balanced accuracy of 66-80% [27]. A similar model achieved 91.7% overall accuracy when trained and tested on Landsat images, using three visible-spectrum Landsat bands (B2-B4) and a long-wave infrared (LWIR) band derived from averaging Landsat B10 and B11 [28]. Improved results can be achieved when multispectral input data is considered – Spatial Procedures for Automated Removal of Cloud and Shadow (SPARCS), a convolutional neural network (CNN) trained on all Landsat 8 bands except the panchromatic band (B8), can discriminate between clear-sky, cloud, cloud shadow, water, and snow/ice pixels with 97.1% accuracy [25]. However, SPARCS only categorizes the central region of an image, requiring a 28 px buffer around the classifiable region, and SPARCS' performance metrics allow for a 2 px accuracy buffer for cloud and cloud shadow boundaries [25]. The use of a 2 px accuracy buffer means that SPARCS' performance metrics consider pixels that fall within 2 px of a cloud or cloud shadow boundary to be classified correctly if SPARCS classifies these pixels as either of the boundary classes [25].

### 1.2.3 Road Detection

Road detection is key to autonomous driving and route planning and has been studied extensively [14]. Current state-of-the-art approaches to road detection use deep learning and generally rely on CNN-inspired architectures, sometimes with post-processing to leverage road network geometry to improve performance.

Many road segmentation models use deep learning architectures without post-processing. Volodymyr Mnih created the Massachusetts Roads Dataset and applied a CNN without post-processing to road segmentation, inspiring a follow-on movement of research into applying deep learning to road segmentation in aerial imagery [38]. A related segmentation model without post-processing is LinkNet, a U-Net-inspired architecture which extensively uses feed-forward operations, concatenating the input of each encoder block in the model to the input of each decoder block in order to retain spatial information for semantic segmentation [11]. LinkNet slightly outperforms vanilla U-Nets on road segmentation when evaluated on the Massachusetts Roads Dataset, achieving an $F_1$ score of 0.8393 when a 4 px buffer is taken into account at road boundaries; in contrast, the U-Net achieves a lower $F_1$ score of 0.8339 [5]. Another approach is using a general adversarial network (GAN), which uses a "discriminator" network to optimize the training of a "generator" network, which outputs segmentation masks. Abdollahi *et al.* have demonstrated very strong performance on road segmentation on a limited subset of the Massachusetts Roads Dataset using a GAN, achieving an $F_1$ score of 0.9220 [2].

Some state-of-the-art architectures for road segmentation use post-processing to improve performance. One such approach annotates ground truth masks with road orientation labels and uses multi-task learning to jointly learn road orientation and perform pixel-level road segmentation, then uses a post-processing CNN to refine the road network [7]. The post-processing CNN is trained on corrupted versions of ground truth masks to recover missing segments of the road network and to delete singleton road pixels [7]. This "orientation learning" approach leverages the fact that roads can be represented as vectors and that road networks are typically fully connected to improve performance. Bandara *et al.* developed an architecture that integrates "orientation learning" and graph learning in both spatial and interaction space into a ResNet-inspired deep learning architecture in order to learn road network topologies and further improve the connectivity and accuracy of road masks [5].

## 1.3 Summary of Literature

Translation-equivariant [28] and $C_8$-equivariant [37] deep learning models have separately been successfully applied to segmenting satellite imagery. Deep learning models that are both $C_8$-equivariant and translationally equivariant have successfully demonstrated identifying handwritten digits in images, outperforming non-equivariant models [51]. Many non-equivariant deep learning models exist for both road detection in aerial imagery and cloud detection in satellite imagery; SPARCS [25] and SPIN Road Mapper [5] are two such state-of-the-art models for cloud and road detection, respectively.

However, despite the excellent work done on deep learning for satellite and aerial image segmentation, no $C_8$- and translation-equivariant models have yet been developed for or applied to segmenting satellite and aerial imagery, and to the author's knowledge, even purely $C_8$-equivariant models have never been applied to either cloud detection or road detection.

## 1.4 Thesis Outline

The core contribution of this thesis is the development of rotationally and translationally equivariant deep learning models for image segmentation onboard CubeSats. This thesis primarily focuses on developing deep learning models for cloud detection in satellite imagery, but also includes models developed for road detection, and draws comparisons between the cloud detection and road detection problem domains and the performance of different algorithms across both domains. This thesis is organized as follows:

- Chapter 1 of this thesis introduces the problems of cloud detection and road detection in satellite and aerial imagery, provides an overview of the field, and describes current state-of-the-art cloud and road detection algorithms.
- Chapter 2 describes the datasets used for road and cloud detection, outlines the models applied to cloud and road segmentation, and explains the metrics used to evaluate the performance and resource consumption of each algorithm.
- Chapter 3 presents the results of the cloud detection experiments and discusses the segmentation performance and resource consumption of each algorithm.
- Chapter 4 presents the results of the road detection experiments and discusses the segmentation performance and resource consumption of each algorithm.
- Chapter 5 analyzes the results of the experiments presented in chapters 3 and 4 and makes comparisons to the literature.
- Chapter 6 summarizes our conclusions and recommendations for future work.

THIS PAGE INTENTIONALLY LEFT BLANK

# Chapter 2

# Approach

This chapter details the cloud and road segmentation case studies and datasets, describes the two rule-based algorithms applied to cloud segmentation and the four deep learning algorithms applied to both road and cloud segmentation, and explains the metrics used to evaluate the segmentation performance and resource consumption of each algorithm.

## 2.1 Case Studies and Datasets

### 2.1.1 Cloud Segmentation

Clouds have varying morphologies; cumulonimbiform, cumuliform, stratiform, stratocumuliform, and cirriform clouds have been visible in Landsat imagery ever since Landsat 1 and Landsat 2 [6]. The exact shape of clouds in each of these five major cloud families can vary significantly; each of these cloud families contains many subtypes [43]. Unlike other categories on which $C_8$-equivariant deep learning models have been demonstrated in the past, clouds as a broad category vary significantly in morphology. Additionally, with "strong orientation" defined as the property that human interpreters are able to identify and agree on the orientation of randomly rotated images of a target segmentation class, many cloud types, especially types in the cumuliform family, do not have a fixed shape or strong orientation [43]. The cloud

Table 2.1: Summary of multispectral data available from Landsat 8; bands we use (2-4, 6, 10) are bolded. Adapted from [3].

| **Band** | **Name** | **Wavelength ($\mu m$)** | **Resolution (m)** |
|---|---|---|---|
| 1 | Coastal Aerosol | 0.433-0.453 | 30 |
| **2** | **Red** | **0.450-0.515** | **30** |
| **3** | **Green** | **0.525-0.600** | **30** |
| **4** | **Blue** | **0.630-0.680** | **30** |
| 5 | Near infrared (NIR) | 0.845-0.885 | 30 |
| **6** | **Short-wave infrared (SWIR) 1** | **1.560-1.660** | **30** |
| 7 | SWIR 2 | 2.100-2.300 | 30 |
| 8 | Panchromatic | 0.500-0.680 | 15 |
| 9 | Cirrus | 1.360-1.390 | 30 |
| **10** | **Long-wave infrared (LWIR) 1** | **10.30-11.30** | **100** |
| 11 | LWIR 2 | 11.50-12.50 | 100 |

detection case study in this thesis is intended to evaluate the performance of $C_8$-equivariant deep learning models on a classical image segmentation problem without strong feature orientation.

### Modified SPARCS Dataset

We evaluate the six different image segmentation models described in §2.2 on a modified version of the Spatial Procedures for Automated Removal of Cloud and Shadow (SPARCS) dataset of 80 Landsat 8 images representing 16 different biomes [25]. The SPARCS dataset contains high-accuracy segmentation masks created by two human operators, classifying each pixel as cloud, cloud shadow, cloud shadow over water, water, ice/snow, land, or flooded for each scene [25]. Because this thesis focuses on cloud detection, we post-process the masks in the SPARCS dataset to re-classify all non-cloud pixels as "background", creating binary masks where each pixel is classified as cloud or background.

Landsat 8 has eleven spectral bands (see Table 2.1), but the SPARCS dataset includes data from Landsat bands 1-10 only. We chose to focus on visible-spectrum, long-wave infrared (LWIR), and short-wave infrared (SWIR) imagery, and so we only use data from Landsat bands 2-4 (blue, green, and red), band 6 (SWIR 1), and band 10 (LWIR 1).

### 2.1.2 Road Segmentation

Unlike clouds, roads, especially straight roads in gridlike road networks, have strong orientation. State-of-the-art road detection models like SPINRoadMapper and Orientation Learning leverage this orientation to better identify road networks, but none of these models are $C_8$-equivariant [5] [7]. The road detection case study in this thesis is intended to evaluate the performance of $C_8$-equivariant deep learning models on a satellite image segmentation problem with strong feature orientation.

#### Modified Massachusetts Roads Dataset

We evaluate the image segmentation models described in §2.2 on a modified version of the Massachusetts Roads Dataset [38]. The Massachusetts Roads Dataset contains 1171 aerial images of roads in Massachusetts, and the truth masks were generated by hand-correcting rasterized road centerlines from OpenStreetMap labels [38]. Unfortunately, not all images in the Massachusetts Roads Dataset are complete – some include considerable portions of blank white space. As in [5], we cropped all images in the Massachusetts Roads Dataset from $1500 \times 1500$ pixels to multiple $512 \times 512$ pixel patches, and discarded the resulting crops that contained white space, keeping only the "complete" crops.

## 2.2 Models

This thesis focuses primarily on evaluating the performance of the $C_8$-equivariant dense U-Net presented in §2.2.6 on segmentation of satellite and aerial imagery. The luminosity thresholding algorithm, random forest algorithm, and U-Net model described in §2.2.1-2.2.3 are included as baseline methods for comparison, and the dense

U-Net and $C_8$-equivariant U-Net models described in §2.2.4-2.2.5 are included to help distinguish the effects of dense blocks and the effects of $C_8$-equivariance.

### 2.2.1 Luminosity Thresholding

Luminosity thresholding is a simple cloud detection algorithm which classifies each pixel individually based on its luminosity in each band of an image. For a multi-band image, one luminosity threshold is found for each band. Each pixel is classified based on its luminosity in each band, resulting in a probabilistic cloud mask with the contribution from each band weighted equally. For an image with four bands, a pixel that is classified as the target class (e.g. cloud) based on its luminosity in three of the four bands, but classified as background based on its luminosity in the fourth band, would receive an 0.75 probability of being the target class in the final probability mask.

For each band, given $n$ pixels in the training set, the luminosity thresholding algorithm implemented in this thesis solves for the threshold $t$ and directional parameter $d$ satisfying:

$$\underset{t}{\operatorname{argmax}} \Sigma_{i=1}^{n} f(t, d, \ell_i, c_i) \tag{2.1}$$

where the function $f(t, d, \ell_i, c_i)$ represents the binary classification accuracy of a pixel $i$ with luminosity $\ell_i$ and true class $c_i$ given a threshold $t$ and direction $d$:

$$f(t, d=1, \ell_i, c_i=1) = \begin{cases} 1 & \ell_i > t \\ 0 & \ell_i \leq t \end{cases} \qquad f(t, d=1, \ell_i, c_i=0) = \begin{cases} 1 & \ell_i \leq t \\ 0 & \ell_i > t \end{cases} \tag{2.2}$$

$$f(t, d=0, \ell_i, c_i=1) = \begin{cases} 1 & \ell_i \leq t \\ 0 & \ell_i > t \end{cases} \qquad f(t, d=0, \ell_i, c_i=0) = \begin{cases} 1 & \ell_i > t \\ 0 & \ell_i \leq t \end{cases} \tag{2.3}$$

The directional parameter $d$ essentially represents whether or not luminosity in a certain band is positively correlated with the target class ($d = 1$) or negatively correlated with the target class ($d = 0$). The threshold $t$ is used to classify a pixel based on its luminosity; for example, if the luminosity is positively correlated with the target class ($d = 1$) and the pixel luminosity is $\geq t$, then the pixel is classified as the target class. The pixel true class $c_i$ represents whether a pixel belongs to the target class ($c_i = 1$) or not ($c_i = 0$).

Luminosity thresholding can be considered to be a special case of the random forest algorithm presented in subsection 2.2.2, where for an image with $b$ bands, $b$ depth-2 trees vote on whether each pixel should be classified as the target class or as background based on its luminosity in each of the $b$ bands. Figure 2-1 shows this architecture for a luminosity thresholding architecture which classifies pixels based on luminosity in the red, green, blue, LWIR, and SWIR bands of an image.

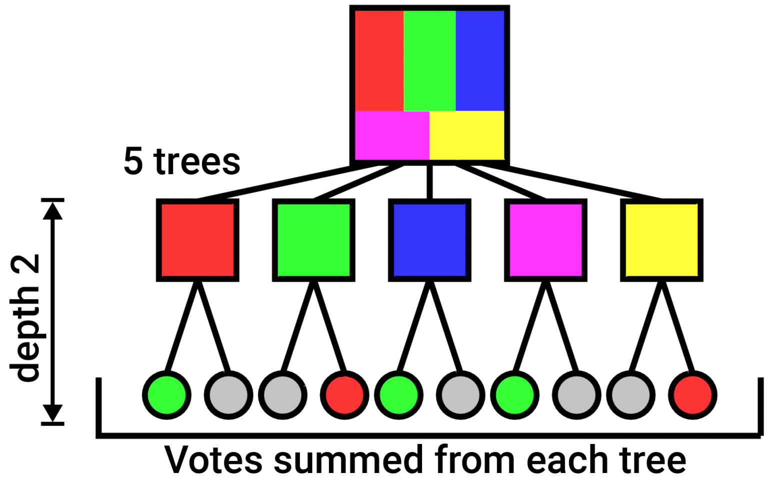


Figure 2-1: Luminosity thresholding architecture diagram showing trees representing red, green, blue, LWIR, and SWIR bands voting based on thresholds in each band in a 5-tree random forest. Figure credit: Alex Meredith and Shreeyam Kacker, MIT.

Unlike the kernel-based random forest algorithm presented in section 2.2.2 and the deep learning algorithms presented in sections 2.2.3-2.2.6, luminosity thresholding classifies each pixel individually without considering the context of any surrounding pixels. The luminosity thresholding algorithm is therefore unable to learn or leverage the morphology of the target class (e.g. clouds, road networks) when classifying images.

### 2.2.2 Random Forest

Random forest classification is an ensemble learning method where $n$ trees of depth $d$ are trained individually on separate random samples of the training data. Each depth-$d$ tree makes $d$ decisions to classify a pixel and its associated features, and then the $n$ final classification decisions made by each of the $n$ trees are aggregated into a collective decision made by the forest, which typically outperforms any individual tree as a classifier [41].

During training, each tree solves for the splits that will maximize the decrease in "impurity", which is roughly equivalent to the likelihood of eventual misclassification of a data point, at each of its $d$ levels [34]. In this thesis, we use Gini impurity, which is the probability of incorrectly classifying a data point if it were randomly labeled based on the class distribution of the dataset [34]. For a split $t$ and classification between $n$ classes $c_1...c_n$, where $p(c_i|t)$ represents the probability that a data point belongs to class $c_i$ conditional on the split $t$, Gini impurity is given by [34]:

$$i_g(t) = \Sigma_{i=1}^{n} \, p(c_i|t)(1 - p(c_i|t)) \tag{2.4}$$

As in [48], [28], and [27], we use a kernel-based random forest classifier, which considers pixel luminosities in a $k \times k$ kernel centered on the pixel of interest when classifying a pixel. As shown in Figure 2-2, we use a $3 \times 3$ kernel around each pixel, and as such, the random forest we trained uses 9 features per imaging band considered for each pixel; we use 5 bands, so our random forest uses a total of $3 \times 3 \times 5 = 45$ features per pixel. In contrast, the luminosity thresholding algorithm presented in §2.2.1 only considers 1 feature per band for each pixel (namely, the pixel luminosity), for a total of 5 features per pixel. The deep learning algorithms explored in §2.2.3-2.2.6 classify each image as a whole, and thus consider tens of thousands of features per band during classification. The random forest algorithm therefore represents a middle ground between the luminosity thresholding algorithm and the deep learning algorithms explored in §2.2.3-2.2.6 in terms of number of features considered, amount of spatial context considered, and computational complexity.

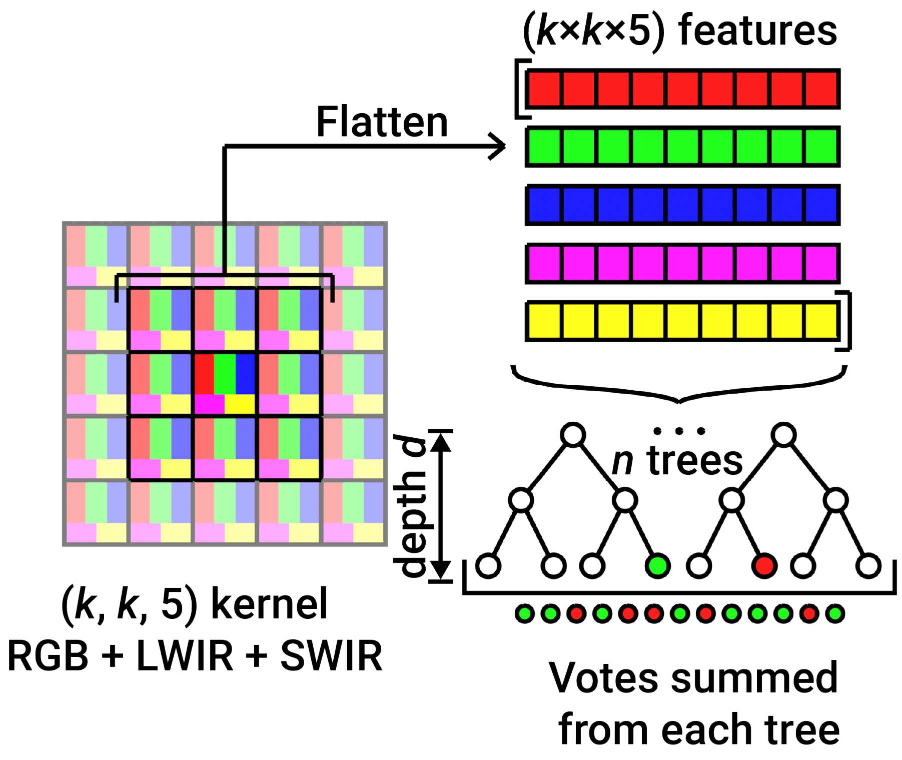


Figure 2-2: Random forest architecture diagram showing classification of a single pixel based on luminosities in the red, green, blue, LWIR, and SWIR bands of pixels in a $k \times k$ window surrounding the pixel of interest, using a forest with $n$ trees of depth $d$. Figure credit: Alex Meredith and Shreeyam Kacker, MIT.

### 2.2.3 U-Net

Ronneberger *et al.* created the U-Net architecture for biomedical image segmentation [42]. U-Nets are "U-shaped" and consist of a contracting path, where an input is repeatedly spatially downscaled to capture context, and an expanding path, where an input is repeatedly spatially upscaled to precisely localize the features in the final output map [42]. U-Nets also feature concatenations between the contracting path and the expanding path; feed-forward connections like these smooth the "loss landscape" of deep learning models, improving model convergence [30].

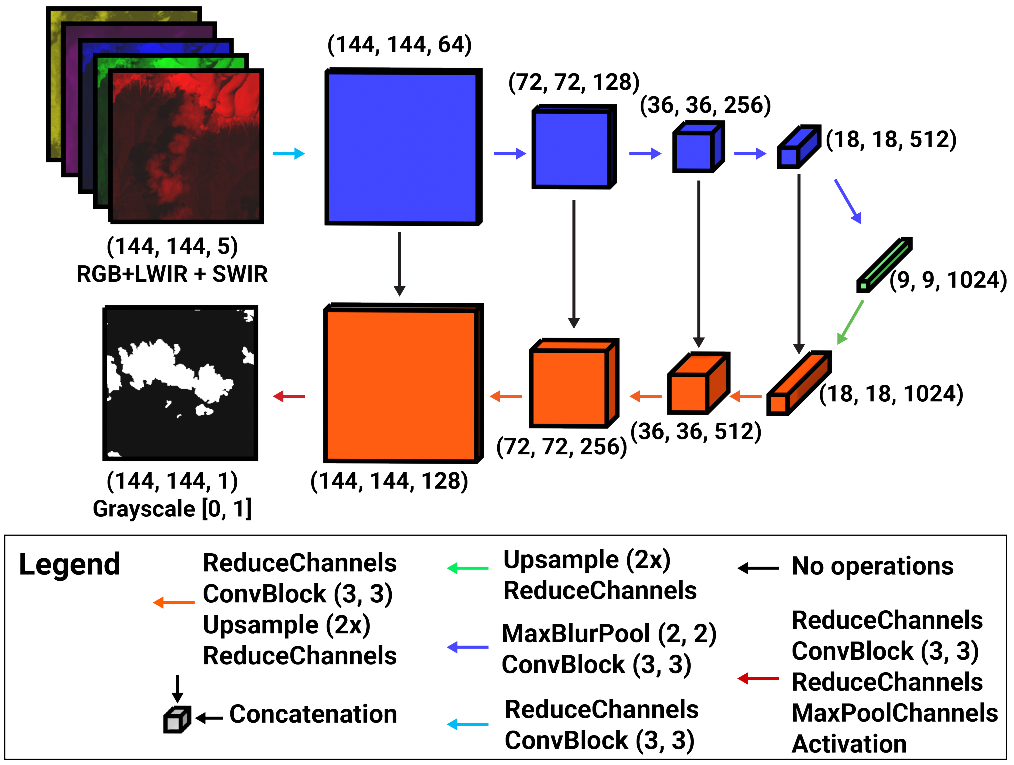


Figure 2-3: U-Net model architecture showing classification of an entire five-band (red, green, blue, LWIR, SWIR) image. See Figure 2-4 for the definitions of "ReduceChannels", "ConvBlock", and "Activation". Image credit: Alex Meredith and Shreeyam Kacker, MIT.

As shown in Figure 2-3, we adapted the original U-Net architecture [42] to work for a $144 \times 144$ pixel image rather than a $512 \times 512$ pixel image. We kept the same number of channels in each layer as in the original U-Net, using 64 channels in the input to our first spatial downsampling operation [42]. We replaced max-pooling operations in the

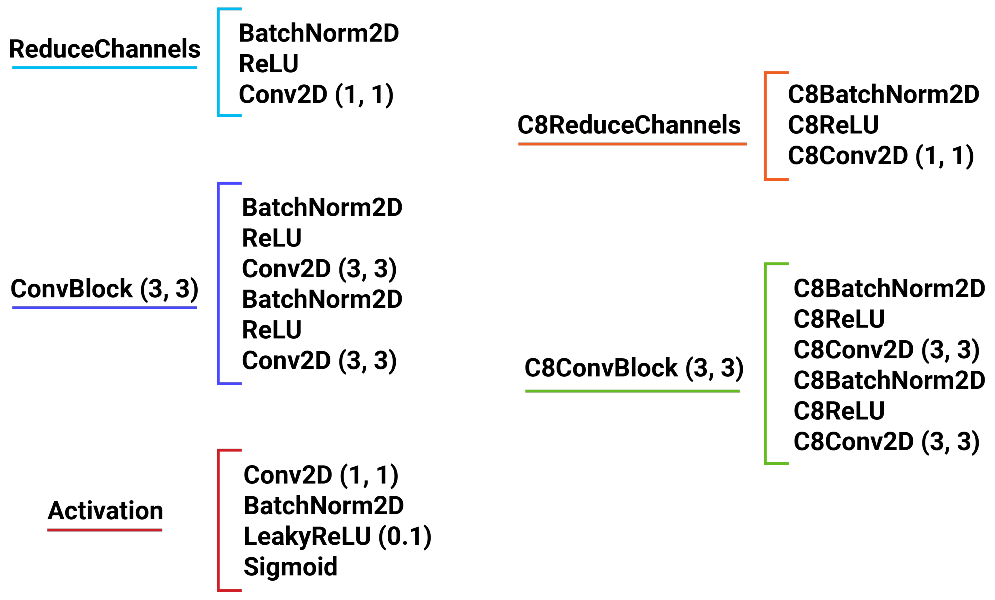


Figure 2-4: Details of operations used in the deep learning models described in §2.2.3-2.2.6. Image credit: Alex Meredith, MIT.

original U-Net with max-blur-pooling, which greatly reduces the aliasing caused by max-pooling and thus improves translational equivariance [53]. We also added batch normalization prior to each rectified linear unit (ReLU) to reduce internal covariate shift and speed up training [26].

As defined by Ioffe & Szegedy, internal covariate shift is “the change in the distribution of network activations due to the change in network parameters during training”, and can cause nonlinear activation functions to “saturate”, meaning that their derivatives trend towards zero, slowing down the convergence of a network [26]. Batch normalization integrates normalization into a model architecture to prevent activation functions from saturating, and uses mini-batch statistics for normalization for reasons of computational efficiency and gradient propagation [26].

### 2.2.4 Dense U-Net

Dense deep learning models replace convolution layers with “dense” blocks, which are composed of multiple convolution layers with feed-forward operations that connect the inputs of each of the convolutions [23]. These feed-forward connections, like the feed-forward connections used to connect the contracting and expanding paths in U-Nets, improve gradient propagation and smooth the "loss landscape" of a network, improving model convergence [30]. Dense U-Nets combine dense blocks with U-Nets, replacing the convolution layers in a U-Net with dense blocks.

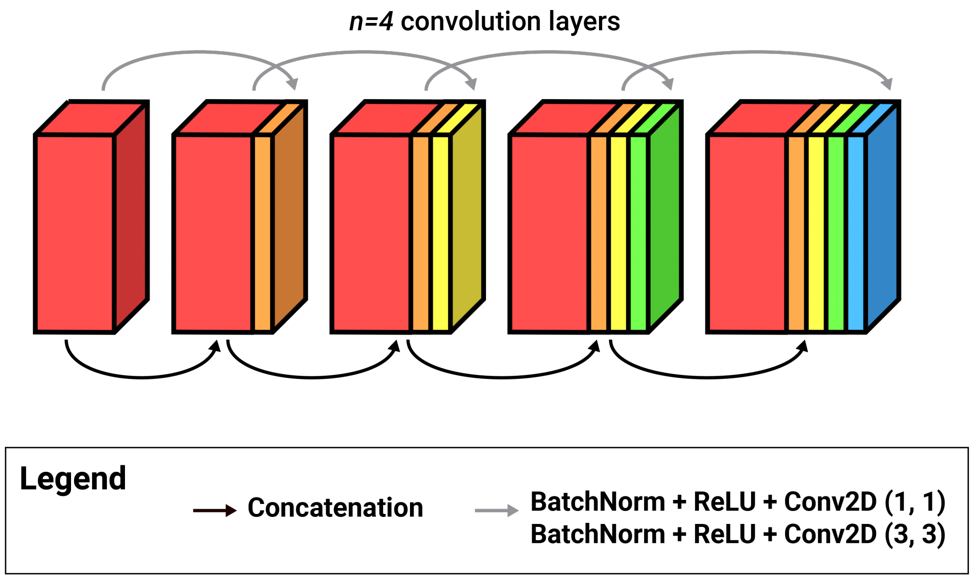

*n=4* convolution layers
Legend
Concatenation
BatchNorm + ReLU + Conv2D (1, 1)
BatchNorm + ReLU + Conv2D (3, 3)

Figure 2-5: Architecture of a "dense block", which is the primary building block of the dense U-Net, with $n = 4$ convolution layers and $c_{out} = c_{in}/4$ output channels for the convolutional layers. Image credit: Alex Meredith, MIT.

The four primary design parameters for dense blocks are $n$, the number of convolution layers per dense block, $k_1$, the size of the kernel used in the first convolution in each convolution layer, $k_2$, the size of the kernel used in the second convolution in each convolution layer, and $c_{out}$, the number of output channels of each convolution layer. The input of each convolution layer is concatenated to its output before undergoing the next convolution; as a result, the output of a dense block with $c_{in}$ input channels has $c_{in} + nc_{out}$ channels. A typical dense block from our dense U-Net is shown in Figure 2-5, with $n = 4$ convolution layers, $k_1 = 1$, $k_2 = 3$, $c_{out} = c_{in}/4$, where $c_{in}$ is the number of channels in the input to the dense block, and $2c_{in}$ channels in the final dense block output.

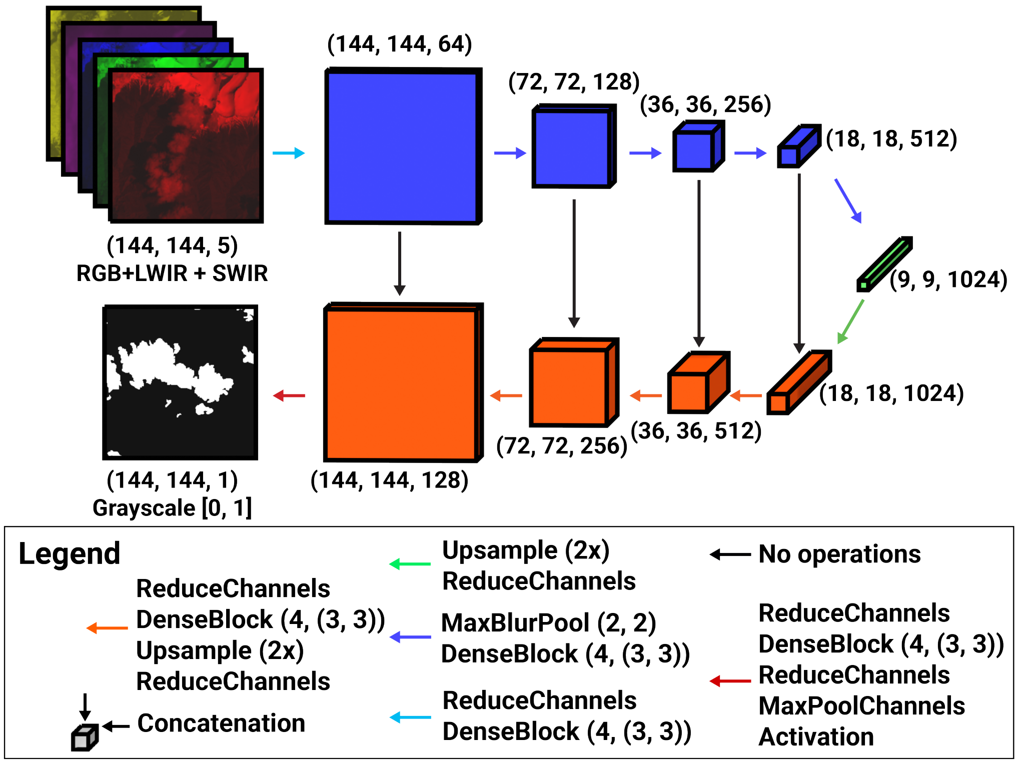


Figure 2-6: Dense U-Net-based model architecture showing classification of an entire five-band (red, green, blue, LWIR, SWIR) image. See Figure 2-4 for the definitions of "ReduceChannels" and "Activation", and see Figure 2-5 for the definition of "DenseBlock". Image credit: Alex Meredith and Shreeyam Kacker, MIT.

In this thesis, all dense blocks use $n = 4$ convolution layers, fewer than the 6 to 24 convolution layers used in each dense block in the original DenseNet [23]. Other dense deep learning models sometimes vary the number of convolution layers used in each dense block; Graham *et al.* use 3 to 6 convolution layers per dense block in their DSF-CNN for classification and 2 to 4 convolution layers per dense block for segmentation [17]. However, we exclusively use $n = 4$ convolution layers per dense block, which has previously been shown by Cai *et al.* to outperform a U-Net on biomedical image segmentation, improving $F_1$ score from 0.9002 to 0.9335, when integrated into a U-Net-based architecture [9].

As in the original DenseNet [23], we use a $1 \times 1$ convolution kernel in the first convolution and a $3 \times 3$ convolution kernel in the second convolution of each convolution layer. The $1 \times 1$ convolution is a "bottleneck" layer used to reduce the number of channels of the input into the $3 \times 3$ convolution, speeding up training [23]. The $3 \times 3$ convolution extracts spatial context and is computationally lighter than convolutions with larger kernels. Convolutions with larger kernels can be decomposed into successive $3 \times 3$ convolutions, making $3 \times 3$ convolutions useful for computationally efficient deep learning networks [21].

### 2.2.5 $C_8$-Equivariant U-Net

The $C_8$-equivariant U-Net adapts the architecture of the U-Net described in §2.2.3 to be $C_8$-equivariant, resulting in a $C_8$-equivariant steerable filter CNN (SFCNN)[50]. $C_8$ is the group of 45° rotations, so $C_8$-equivariance means that any rotation of an integer multiple of 45° of the model input yields an equivalent rotation of the outputted feature map. $C_8$-equivariance proves especially useful for segmenting satellite and aerial imagery because satellite and aerial images can be rotated arbitrarily about the camera boresight axis. The rotation equivariance property of $C_8$-equivariant deep learning models does not hold for rotations other than integer multiples of 45°, but $C_8$-equivariant models have shown good equivariance to other rotations in practice, especially when the training dataset is augmented with arbitrary rotations [51].

Because the regular representation of $C_8$ supports pointwise nonlinearities, many operations, including batch normalization and rectified linear activation unit (ReLU) operations, can simply be applied to each orientation representation in $C_8$-equivariant deep learning models while preserving $C_8$-equivariance [50]. $C_n$-equivariant convolutions (where $C_n$ is the group of $\frac{360°}{n}$ rotations), however, differ fairly significantly from regular convolutions. $C_n$-equivariant convolutions operate on functions on $\mathbb{R}^2 \rtimes C_n$ and use g-filters, which are composed of $n$ different filters [50]. Each g-filter has $n$ different representations; the different filters are rotated and reordered in each representation [50]. An input feature map on $\mathbb{R}^2 \rtimes C_n$ is convolved with each of the $n$ representations of the g-filter, and a linear combination over all $n$ orientations in the output of each convolution is taken to form a single orientation representation in the output feature map on $\mathbb{R}^2 \rtimes C_n$ [50].

For clarity, a $C_4$-equivariant convolution is shown in Figure 2-7, along with an "input convolution" that transforms an input image on $\mathbb{R}^2$ to a function on $\mathbb{R}^2 \rtimes C_4$ and an "orientation pooling" layer that takes the orientation-wise maximum to generate a final output map on $\mathbb{R}^2$. See Appendix A for further detail on $C_n$-equivariant convolutions; Figure 2-7 is also reproduced there.

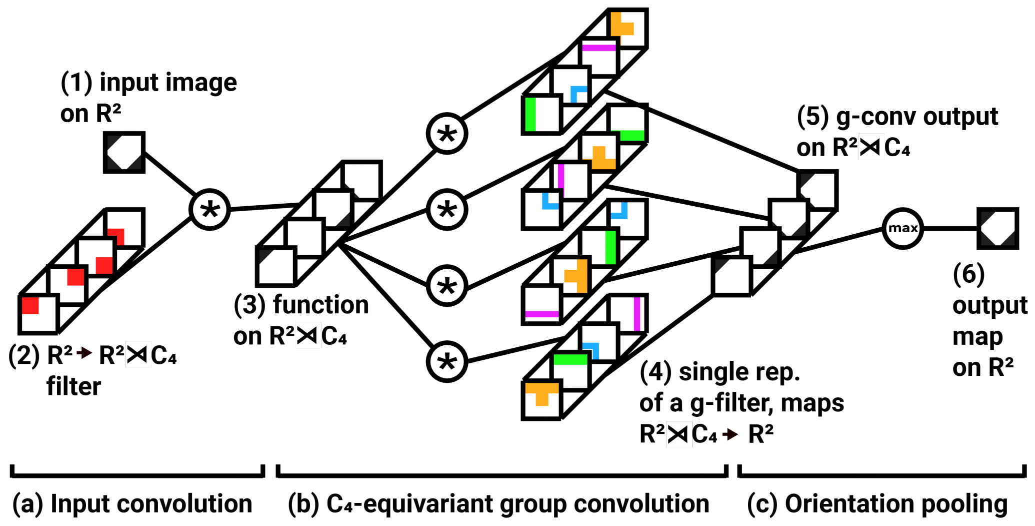


Figure 2-7: (a) An input convolution that lifts an input image to $\mathbb{R}^2 \rtimes C_4$, (b) A $C_4$-equivariant group convolution, (c) Orientation pooling to produce an output map on $\mathbb{R}^2$. Image credit: Alex Meredith, MIT.

The $C_8$-equivariant U-Net is very similar to the U-Net presented in §2.2.3, but with nearly all operations replaced with $C_8$-equivariant equivalents, such as $C_8$-equivariant convolutions and equivariance-preserving pointwise nonlinearities. The only exceptions occur in the first "input" convolution step (shown with the light blue arrow in Figure 2-8) and the final "output" convolution step (shown with the red arrow in Figure 2-8).

For the $C_8$-equivariant U-Net, the input convolution step involves a non-equivariant batch normalization and rectified linear activation unit (ReLU) followed by a convolution that "lifts" the input from an image on $\mathbb{R}^2$ to a feature map on $\mathbb{R}^2 \rtimes C_8$. As shown in Figure 2-7(a), a $C_n$-equivariant "lifting" convolution involves convolving the input with a single filter rotated into $n$ different orientations. In contrast, the U-Net performs a normal convolution on $\mathbb{R}^2$ instead of a "lifting" convolution in its input block, as shown in Figure 2-3, because the U-Net operates on feature maps on $\mathbb{R}^2$

rather than $\mathbb{R}^2 \rtimes C_8$ at each step.

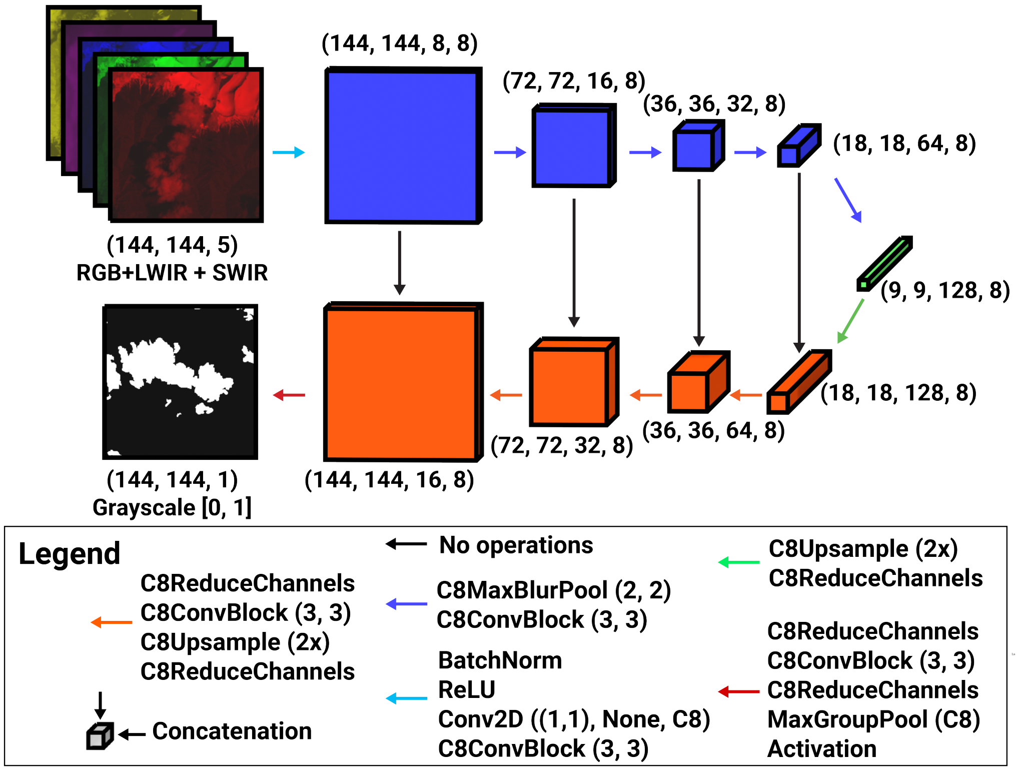


Figure 2-8: $C_8$-equivariant U-Net-based model architecture showing classification of an entire five-band (red, green, blue, LWIR, SWIR) image. At each layer, the (x, y, c, n) tuple represents the image dimensions $(x \times y)$, number of channels per orientation $c$, and number of orientations $n$, with $n = 8$ representing $C_8$-equivariance. See Figure 2-4 for the definitions of "C8ConvBlock", "C8ReduceChannels", and "Activation". Image credit: Alex Meredith and Shreeyam Kacker, MIT.

The output convolution step involves "group pooling", also called "orientation pooling", which projects a feature map on $\mathbb{R}^2 \rtimes C_8$ back to $\mathbb{R}^2$, followed by the same activation block used in the U-Net. Orientation pooling, as shown in Figure 2-7(c), involves taking the pixel-wise maximum or average across the orientation dimension. This is the same operation as the channel-wise max pooling performed by the U-Net in its output block, except over orientations instead of channels.

### 2.2.6 $C_8$-Equivariant Dense U-Net

The $C_8$-equivariant dense U-Net is the most computationally complex model explored in this thesis. It replaces the $C_8$-equivariant convolution blocks described in §2.2.5 with $C_8$-equivariant versions of the dense blocks described in §2.2.4. The $C_8$-equivariant dense U-Net is able to both leverage the natural rotational symmetry of satellite and aerial imagery and benefit from the improved gradient propagation of dense deep learning networks. Like the dense U-Net detailed in §2.2.4, the

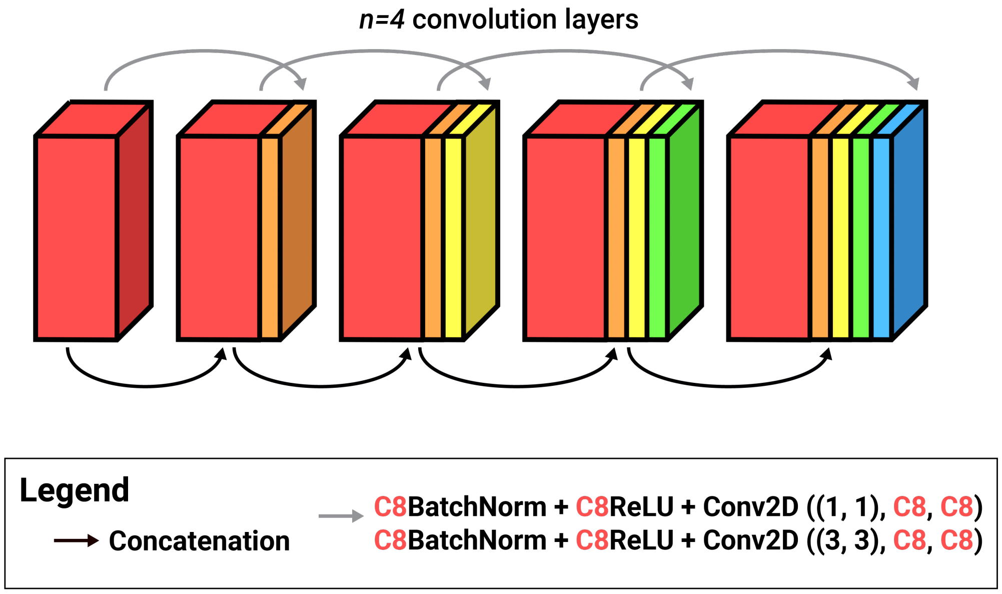

*n=4* convolution layers
Legend
Concatenation
C8BatchNorm + C8ReLU + Conv2D ((1, 1), C8, C8)
C8BatchNorm + C8ReLU + Conv2D ((3, 3), C8, C8)

Figure 2-9: Architecture of a $C_8$-equivariant "dense block", which is the primary building block of the $C_8$-equivariant dense U-Net. The $C_8$-equivariant "dense block" differs from the "dense block" shown in Figure 2-9 by the use of $C_8$-equivariant operations, emphasized here in red. Image credit: Alex Meredith, MIT.

$C_8$-equivariant dense U-Net uses $k = 4$ convolution layers in each dense block, has $c_{out} = c_{in}/4$ output channels for each convolution layer, and performs a $1 \times 1$ convolution followed by a $3 \times 3$ convolution in each convolution layer, as shown in Figure 2-9. However, unlike the convolutions in the dense U-Net, the convolutions in the dense block used in the $C_8$-equivariant dense U-Net are, naturally, $C_8$-equivariant.

As shown in Figure 2-10, the $C_8$-equivariant dense U-Net has essentially the same architecture as the $C_8$-equivariant U-Net (see Figure 2-8), except for the use of $C_8$-equivariant dense blocks in place of $C_8$-equivariant convolution blocks.

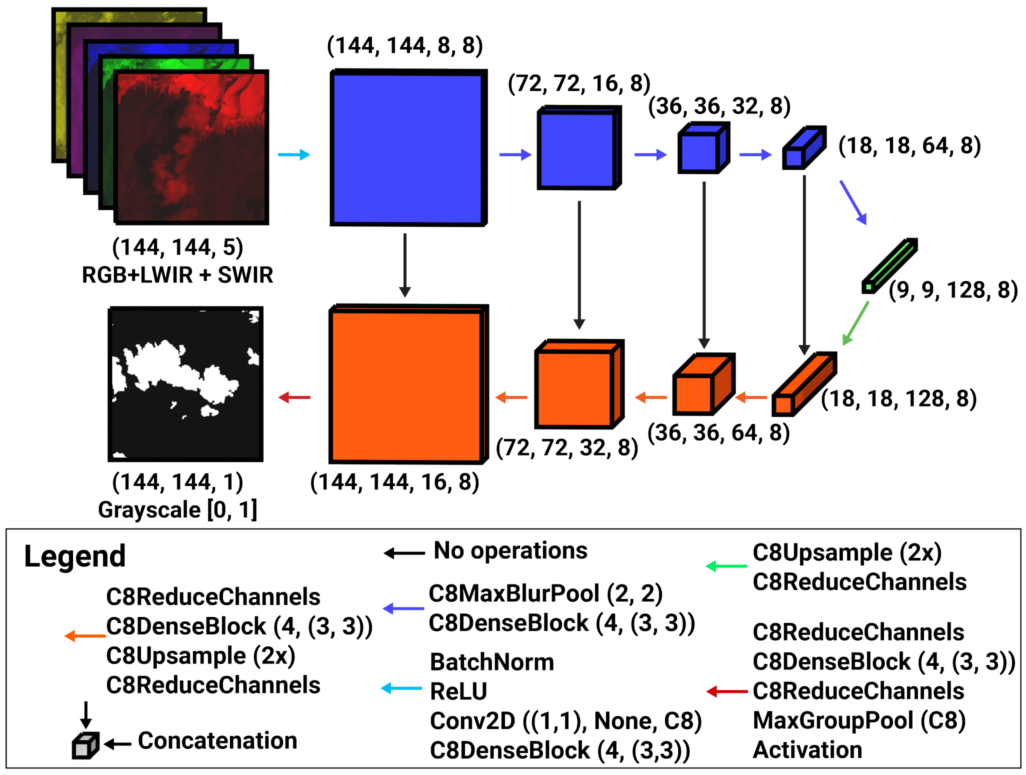


Figure 2-10: $C_8$-equivariant dense U-Net-based model architecture showing classification of an entire five-band (red, green, blue, LWIR, SWIR) image. At each layer, the (x, y, c, n) tuple represents the image dimensions ($x \times y$), the number of channels per orientation $c$, and the number of orientations $n$. See Figure 2-4 for the definitions of "C8ReduceChannels" and "Activation", and see Figure 2-9 for the definition of "C8DenseBlock". Image credit: Alex Meredith and Shreeyam Kacker, MIT.

## 2.3 Metrics

In this section, we present the quantitative metrics used to evaluate the classification performance and resource consumption of each algorithm used in our cloud detection and road detection experiments.

### 2.3.1 Classification Performance

Each model trained on the road dataset generates a road mask for each image in the test set, and each model trained on the cloud dataset generates a cloud mask for each image in the test set. The resulting masks show the probability of a cloud or road at each pixel, ranging from 0 (no cloud or no road, respectively) to 1 (certain cloud or certain road, respectively). We convert each probability mask to a binary mask with a threshold of 0.5 – all pixels with probability $\geq 0.5$ of a road or cloud are considered to be a road or cloud prediction, respectively, and all pixels with probability $< 0.5$ of a road or cloud are considered to be a non-cloud or non-road prediction, respectively.

After generating binary masks for each image, we find true positives (model-generated mask correctly makes a positive prediction for each pixel), true negatives (model-generated mask correctly makes a negative prediction for each pixel), false positives (model-generated mask incorrectly makes a positive prediction), and false negatives (model-generated mask incorrectly makes a negative prediction at each pixel).

We use these pixel-level classifications to calculate accuracy, sensitivity, specificity, balanced accuracy, precision, recall, $F_1$ score, and intersection over union (IoU) for each model. For a perfect model, each of these metrics would be 1.0, reflecting perfect classification at the pixel level that results in zero false positives and zero false negatives. These metrics are given by the following equations, with $TP$ representing the number of true positives, $TN$ representing the number of true negatives, $FP$ representing the number of false positives, and $FN$ representing the number of false negatives.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.5}$$

$$\text{Sensitivity} = \frac{TP}{TP + FN} \tag{2.6}$$

$$\text{Specificity} = \frac{TN}{TN + FP} \tag{2.7}$$

$$\text{Balanced Accuracy} = \frac{1}{2}(\text{Sensitivity} + \text{Specificity}) \tag{2.8}$$

$$\text{Precision} = \frac{TP}{TP + FP} \tag{2.9}$$

$$\text{Recall} = \text{Sensitivity} \tag{2.10}$$

$$F_1 \text{ Score} = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{2.11}$$

$$\text{IoU} = \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall} - \text{Precision} \cdot \text{Recall}} \tag{2.12}$$

In addition to the binary classification metrics in Equations 2.5-2.12, we plot the receiver operating characteristic (ROC) curve for each model. The ROC curve captures the probabilistic nature of the model-generated cloud and road masks, plotting true positive rate on the y-axis against false positive rate on the x-axis for different cloud or road prediction thresholds from 0 to 1. We additionally calculate AUC (area under the curve), a metric which represents the area under the ROC curve. AUC ranges from 0 to 1, where 1 represents a perfect classifier and 0.5 represents random guessing. We approximate AUC using midpoint Riemann sums.

### Buffered Evaluation

"Buffered evaluation" is a common practice for evaluating segmentation performance when object boundaries are fuzzy or truth masks are inconsistent in quality. When buffered evaluation is used, any pixels within a fixed distance (or buffer) of a boundary between two classes can be classified as either of the classes on the boundary and still be considered correct [39]. Hughes & Kennedy used a 2 pixel buffer on cloud and cloud shadow boundaries when evaluating the original SPARCS CNN to capture the inherent fuzziness of cloud boundaries [25]. Bandara *et al.* used a 4 pixel buffer on road boundaries because the truth masks in the Massachusetts Roads Dataset are generated using rasterized road centerlines and have uniform-width roads [5] [38].

In this thesis, we evaluate the results of each of our cloud detection experiments with and without a 2 pixel buffer on cloud boundaries. We evaluate the results of each of our road detection experiments with and without a 4 pixel buffer on road boundaries. In our buffered evaluation, any pixel within 2 pixels of a cloud boundary (or within 4 pixels of a road boundary) is considered correctly classified regardless of its model-generated class label. We then compute and report each classification metric both without considering a buffer and for our buffered evaluation.

### False Positive Ratios

The SPARCS dataset contains labels for cloud shadows, cloud shadows over water, water, ice/snow, land, and flooded terrain classes in addition to cloud labels [25]. We can gain insight into the relative performance of each model on distinguishing clouds from different types of terrain by analyzing the frequency of each terrain class among pixels falsely identified as clouds by each model. For each of our cloud experiments, we present a table of ratios of frequency among false positives to overall frequency among non-cloud pixels in the SPARCS dataset for each terrain class. These ratios express whether or not each terrain class is overrepresented or underrepresented in the set of pixels falsely identified as cloud by the model; a ratio exceeding 1.0 indicates that a terrain class is overrepresented amongst false positives, and a ratio below 1.0 indicates that a terrain class is underrepresented amongst pixels falsely identified as cloud by the model.

Importantly, these ratios have certain limitations. First, the ratios only consider pixels which are falsely identified as cloud by the model, and do not consider pixels which are falsely identified as non-cloud by the model. For example, if the model misclassifies many cloud pixels over snow as non-cloud, the ratios do not capture this trend. Another key limitation is that models vary in specificity, so if one model has a higher ratio for a specific terrain class when compared to another model, this increase in ratio does not necessarily indicate a higher number of false positives belonging to that terrain class. These ratios can only be used to compare *relative* performance on different terrain classes. For example, a model with a ratio of 2.0 for snow/ice terrain can be said to have a relative disadvantage at distinguishing clouds from ice and snow versus other terrain types when compared to a model with a ratio of 0.5 for snow/ice terrain, but nothing can be said about which model actually performs better at distinguishing snow/ice terrain from clouds unless additional information is considered.

The Massachusetts Roads Dataset only contains road labels; as such, we do not produce false positive ratio tables for our road detection experiments.

### 2.3.2 Resource Consumption and Model Complexity

We evaluate the complexity of each model based on the **number of trainable model parameters**, the **total number of model parameters**, and the **size of the fully trained and saved model**. In order to evaluate resource consumption, we also consider the **inference time to classify a single image** and the **peak memory allocated when classifying a single image**, using both a CPU backend and GPU backend available using Google Colab. The CPU backend uses an Intel Xeon processor, and the GPU backend uses an Nvidia K80 GPU. We average the inference time over 1000 classifications and take the maximum of the peak allocated memory over 100 classifications.

The luminosity thresholding and random forest algorithms do not support a GPU backend, so we only report results for the CPU backend for those algorithms. For the CPU backend, virtual memory usage is measured using the Python `tracemalloc` library with Python 3.8.16, and for the GPU backend, memory usage is measured using PyTorch 1.13.0 – we use `torch.cuda.max_memory_allocated()` to measure the peak memory allocated to tensors on the GPU. Disk usage is not tracked for either backend.

## 2.4 Optimizers, Hyperparameters, and Losses

This section explains the optimizers and loss functions used to train the deep learning models described in §2.2.3-2.2.6. This section also details the hyperparameters we use to train our deep learning models, describing batch size, learning rate, and number of epochs among other hyperparameters.

### 2.4.1 Optimizers and Hyperparameters

#### Cloud Detection Experiments

We train our cloud detection models using the Adam optimizer with the default parameters [29]. We train each model with a learning rate of 0.002 and a *total* batch size of 40 – all models use distributed training across 4 GPUs, and use a batch size of 10 images per GPU for a total batch size of 40 images per epoch. We train the dense U-Net and dense $C_8$-equivariant U-Net for 500 epochs. In machine learning, one epoch refers to all of the training data being passed through a model once, so 500 epochs is equivalent to 500 passes over the training data. The U-Net shows an uncharacteristic spike in test error at 500 epochs, so we train the U-Net and $C_8$-equivariant U-Net for 505 epochs in order to make fair performance comparisons between models.

#### Road Detection Experiments

We train our road detection models with the same optimizer and hyperparameters as Bandara *et al.*, except for the fact that we trained our road detection models for only 90 epochs to prevent overtraining [5]. Accordingly, we use stochastic gradient descent (SGD) as our optimizer with a momentum of 0.9 and weight decay of 0.0005. We train each model for 90 epochs, and use a step-learning rate scheduler with a learning rate of 0.01 for epochs 1-50, and a learning rate of 0.001 for epochs 51-90. We use a *total* batch size of 32 for all models – some models use distributed training across 4 GPUs, and use a batch size of 8 images per GPU for a total batch size of 32 images per epoch [5].

### 2.4.2 Loss Functions

Loss functions describe an objective to minimize when training a deep learning model – during training, optimizers seek to drive loss on the training dataset to zero, with the hope that a model that achieves low loss on the training dataset will generalize well to the test dataset. In this thesis, we train our cloud detection models using focal loss and train our road detection models using dice loss (detailed in §2.4.2 and §2.4.2 respectively), but also experiment with softIOU loss for road segmentation (see §2.4.2).

#### Focal Loss

We use $\alpha$-weighted focal loss for our cloud detection experiments. Focal loss is a loss function based off of cross-entropy loss, which "down-weights" the loss for examples that are common in the training dataset, weighting the loss from "hard" classification examples more heavily instead [32]. A class-weighted variation on focal loss, $\alpha$-weighted focal loss, weights examples from sparse classes more heavily in order to overcome class imbalance in the training dataset [32].

For $\alpha$-weighted focal loss, there are two parameters: $\gamma$ and $\alpha$. As $\gamma$ increases, "hard" examples, or examples classified incorrectly by the deep learning model with a high confidence, are weighted more strongly [32]. The parameter $\alpha$ is a vector of weights for each class, and as $\alpha_c$ increases for a particular class, examples from that class are weighted more heavily [32]. We chose $\gamma = 2$ and $\alpha_{\text{cloud}} = 0.8$ for our cloud detection experiments.

For a pixel of true class $y$, assuming a probability vector $p$ containing probabilities $p_0, ..., p_n$, with $p_c$ representing the model's outputted probability of the pixel belonging to class $c$, unweighted focal loss is [32]:

$$\text{FL}(p, y) = -(1 - p_y)^{\gamma} \log(p_y) \tag{2.13}$$

For a pixel of true class $y$, assuming a probability vector $p$ and a weights vector $\alpha$ containing weights $\alpha_0, ..., \alpha_n$, with $\alpha_c$ representing the weight given to class $c$, $\alpha$-

weighted focal loss is [32]:

$$\mathrm{FL}_\alpha(\alpha, p, y) = -\alpha_y(1-p_y)^\gamma \log(p_y) \tag{2.14}$$

### SoftIoU Loss

SoftIoU loss, or soft intersection-over-union loss, is a modified version of the intersection-over-union metric often used in image segmentation [24]. We initially used softIoU loss for our road detection experiments in order to match Bandara *et al.* However, our models failed to converge with softIoU loss (the IoU metric is known to have unfavorable gradient properties [45]), so we switched to using dice loss (see §2.4.2), a similar loss function with better gradient properties, for our road detection experiments. We present the results of our road experiments with softIoU loss in Appendix B.

Intersection-over-union, or IoU, for two sets X and Y is:

$$\mathrm{IoU}(X, Y) = \frac{|X \cap Y|}{|X \cup Y|} \tag{2.15}$$

For a binary classification problem, given a vector of true classes $y$ and predicted probabilities $p$, where for pixel $i$, $y_i$ represents the true class of pixel $i$ (e.g. 0 or 1) and $p_i$ represents the model-predicted probability that pixel $i$ belongs to class 1, IoU can be approximated as [24]:

$$\mathrm{IoU}(p, y) = \frac{\Sigma_i y_i p_i}{\Sigma_i y_i + p_i - y_i p_i} \tag{2.16}$$

SoftIoU loss applies the "softmax" function to the vector of predicted probabilities $p$ prior to computing the IoU [24]. When applied to a vector $p$, where $p_i$ represents the model-predicted probability that pixel $i$ belongs to class 1 in a binary classification problem, the "softmax" function is given by:

$$\sigma(p)_i = \frac{\exp(p_i)}{\exp(p_i) + \exp(1-p_i)} \tag{2.17}$$

Finally, softIoU loss is given by:

$$\mathrm{SL}(p, y) = -\mathrm{IoU}(\sigma(p), y) \tag{2.18}$$

**Dice Loss**

For our road detection experiments, we ultimately use dice loss due to its similarity to softIoU loss and its more favorable gradient properties. Dice loss is a loss function based on of the Sørensen-Dice coefficient [31]. The Sørensen-Dice coefficient measures the "similarity" between two sets X and Y [31]:

$$\mathrm{DSC}(X, Y) = \frac{2|X \cap Y|}{|X| + |Y|} \tag{2.19}$$

For a binary classification problem, given a vector of true classes $y$ and predicted probabilities $p$, where for pixel $i$, $y_i$ represents the true class of pixel $i$ (e.g. 0 or 1) and $p_i$ represents the model-predicted probability that pixel $i$ belongs to class 1, dice loss is [31]:

$$\mathrm{DL}(p, y) = 1 - \frac{\Sigma 2 y_i p_i}{\Sigma y_i + p_i} \tag{2.20}$$

## 2.5 Training Facilities and Software Implementation

We use freely available resources for implementing and training our models, including the MIT Engaging cluster for training and several different open-source libraries for implementation.

### 2.5.1 Training Facilities

We train the deep learning models described in §2.2.3-2.2.6 on the MIT Engaging cluster [1], a distributed computing cluster shared by MIT researchers and affiliates. For our road detection experiments, we train the U-Net on a single node with 8 CPUs and 1 GPU, and train all other deep learning models on four nodes with 8 CPUs and 1 GPU each (for a total of 32 CPUs and 4 GPUs for each model). For our cloud detection experiments, we train all deep learning models on four nodes with 8 CPUs and 1 GPU each (for a total of 32 CPUs and 4 GPUs for each model). All GPUs used were Nvidia Tesla K20m GPUs.

We train the luminosity thresholding model described in §2.2.1 and the random forest model described in §2.2.2 on a 2020 Macbook Pro with an M1 processor and 16 GB of RAM.

### 2.5.2 Software Implementation

We implement our random forest model (see §2.2.2) using `scikit-learn` [34]. We implement our deep learning models (see §2.2.3-2.2.6) in Python using the PyTorch machine learning library. We use PyTorch DistributedDataParallel for multi-GPU training. We use a modified version of the `e2cnn` library for the $C_8$-equivariant layers in our $C_8$-equivariant U-Net and our $C_8$-equivariant dense U-Net [10]. We use the `segmentation_tools_pytorch` implementation of dice loss for our road detection experiments; for softIoU loss, we use Bandara *et al.*'s implementation, and we implement focal loss ourselves.

All of our code is freely available at `https://github.com/alexmeredith8299/masters-thesis`.

THIS PAGE INTENTIONALLY LEFT BLANK

# Chapter 3

# Cloud Segmentation Results

## 3.1 Evaluating Model Performance

We analyze the performance of the six algorithms described in §2.2.1-2.2.6 on our modified version of the SPARCS dataset, with each model trained on Landsat 8 data from the red, green, blue, SWIR 1, and LWIR 1 bands. We qualitatively evaluate each model by visually comparing model-generated masks to "truth" masks and multispectral model input for two different example images. The first example image, which we refer to as the "easy" image sample, is part of the image with a root filename of `‘‘LC82210662014229LGN00_18’’` in the SPARCS test set [25]. The second example image, which we refer to as the "hard" image sample, is also from the SPARCS test set and is part of the image with root filename `‘‘LC81480352013195LGN00_32’’` [25]. We quantitatively evaluate each model by calculating the metrics described in §2.3.1 and the ratios of the frequency of each terrain class among false positives to the frequency of each terrain class in the dataset as a whole, as described in §2.3.1. Finally, we measure the number of parameters, saved model size, peak memory usage, and inference time of each algorithm in order to evaluate model complexity and resource usage.

### 3.1.1 Luminosity Thresholding

In Figure 3-1, the luminosity thresholding algorithm is applied to an "easy" image sample – that is, an image without snow, water, or bright coastlines or rooftops. Figure 3-1(a) shows the visible-spectrum image input, Figure 3-1(b) shows the long-wave infrared (LWIR) image input, Figure 3-1(c) shows the short-wave infrared (SWIR) image input, Figure 3-1(f) shows the cloud mask generated by the luminosity thresholding algorithm, Figure 3-1(g) shows the "true" cloud mask, Figure 3-1(e) shows the difference between the luminosity-generated mask and the "truth" mask without buffering, and Figure 3-1(d) shows the difference between the luminosity-generated mask and the "truth" mask allowing for a 2 px buffer at the cloud edges. The luminosity-generated mask (Figure 3-1(f)) looks qualitatively similar to the "truth" mask, if more conservative at the cloud edges – this conservatism is also demonstrated by the differences at the cloud edges shown in Figure 3-1(e). The luminosity-generated mask also misses the optically thin cloud patch boxed in green.

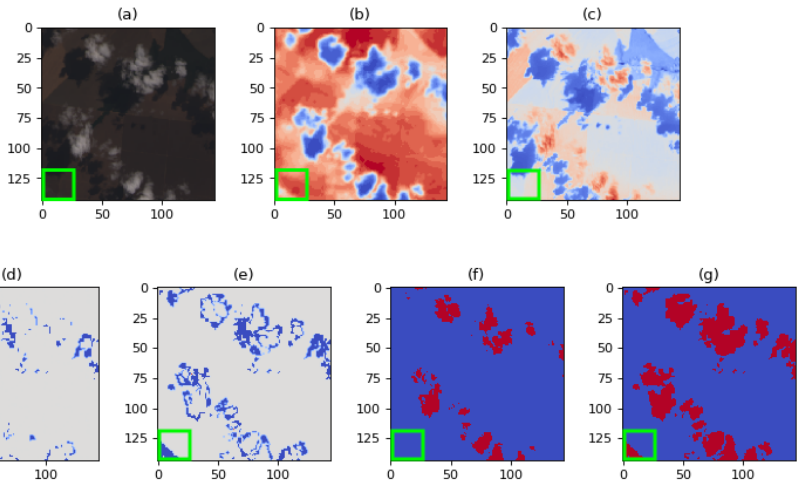


Figure 3-1: Luminosity thresholding algorithm evaluated on an "easy" image segmentation example, with an optically thin cloud patch boxed in green. (a) Visible-spectrum image, (b) LWIR image, (c) SWIR image, (d) Difference between "truth" mask and luminosity-generated mask, excluding pixels within 2 px of a cloud boundary, (e) Difference between "truth" mask and luminosity-generated mask, (f) Luminosity-generated mask, (g) "Truth" mask.

In Figure 3-2, the luminosity thresholding algorithm is applied to a "difficult" image sample – an image with overlapping clouds and snow. This "difficult" image sample also includes an optically thin cloud patch (boxed in green) which is visible in the Landsat panchromatic and cirrus bands (B8 and B9, respectively), but which is not visible any of the bands (Landsat B2-4, B6, and B10) used by the luminosity thresholding algorithm. As shown in Figure 3-2(f), the luminosity-generated cloud mask misidentifies nearly all the snow pixels in this image as cloud, demonstrating qualitatively that the luminosity thresholding algorithm struggles to distinguish between clouds and snow or ice. Also, the luminosity-generated cloud mask misses the optically thin cloud patch boxed in green.

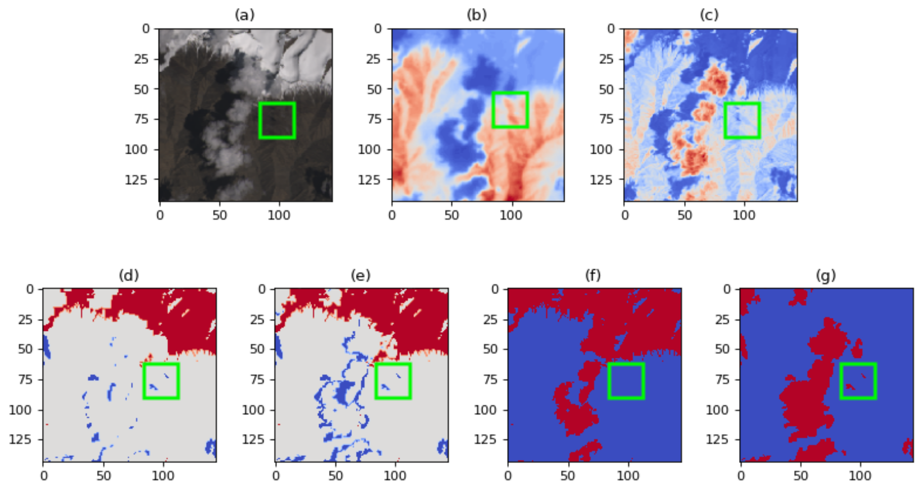


Figure 3-2: Luminosity thresholding algorithm evaluated on a "hard" image segmentation example, with an optically thin cloud patch boxed in green. (a) Visible-spectrum image, (b) LWIR image, (c) SWIR image, (d) Difference between "truth" mask and luminosity-generated mask, excluding pixels within 2 px of a cloud boundary, (e) Difference between "truth" mask and luminosity-generated mask, (f) Luminosity-generated mask, (g) "Truth" mask.

The performance of the luminosity thresholding across all metrics is shown in Table 3.1. Although the specificity of the luminosity thresholding algorithm is relatively high (96.71% with no buffer and 97.24% with a 2 px buffer), the sensitivity of the luminosity thresholding algorithm is very low (31.45% with no buffer and 45.03% with a 2 px buffer). This makes sense, given the qualitative conservatism of the luminosity thresholding algorithm shown in Figures 3-1 and 3-2 – although the luminosity-generated cloud masks have a similar "shape" as the "truth" masks (except for the misidentification of snow pixels as clouds), there are many cloud pixels, especially at the cloud edges, that are classified as non-cloud by the luminosity thresholding algorithm in both luminosity-generated cloud masks.

Table 3.1: Performance metrics detailed in §2.3.1 for the luminosity thresholding algorithm on the cloud dataset, evaluated with no buffer and with a 2 px buffer at cloud boundaries.

| **Buffer** | **Acc.** | **Bal. Acc.** | **Sens.** | **Spec.** | **Prec.** | **Recall** | **F$_1$** | **IoU** |
|---|---|---|---|---|---|---|---|---|
| 0 px | 89.19% | 64.08% | 31.45% | 96.71% | 55.40% | 31.45% | 0.4013 | 0.2510 |
| **2 px** | **92.58%** | **71.13%** | **45.03%** | **97.24%** | **61.48%** | **45.03%** | **0.5199** | **0.3512** |

In addition to “cloud”, the SPARCS dataset has six other terrain labels: cloud shadow over land, cloud shadow over water, water, ice/snow, land, and flooded. We determined the true terrain class of all non-cloud pixels misidentified by the luminosity thresholding algorithm as cloud pixels, and for each terrain class, we divided the frequency of that class amongst false positives by the frequency of that class in the dataset as a whole. The resulting ratios are shown in Table 3.2. Most notably, pixels belonging to the ice/snow terrain class are over 14 times more common amongst pixels misidentified as clouds by the luminosity thresholding algorithm than they are in the dataset as a whole. This is a quantitative indicator that the luminosity thresholding algorithm struggles to disambiguate clouds and ice/snow, likely because the luminosity thresholding algorithm does not take any spatial context into into account when classifying pixels, and clouds and snow have different spatial textures but similar spectral characteristics in the available bands.

### 3.1.2 Random Forest

In Figure 3-3, the random forest algorithm is applied to an “easy” image sample with no snow, water, or bright non-cloud features. The cloud mask generated by the random forest algorithm, shown in Figure 3-3(f), looks qualitatively very similar to the “truth” mask shown in Figure 3-3(g), except for the fact that the random forest algorithm misses an optically thin patch of cloud in the bottom left of the image (boxed in green).

Table 3.2: Ratios representing the frequency of each terrain class (cloud shadow over land, cloud shadow over water, water, ice/snow, land, and flooded) among pixels falsely identified as clouds by the luminosity thresholding algorithm to the frequency of each terrain class among all non-cloud pixels in the SPARCS dataset, as explained in §2.3.1, evaluated with no buffer and with a 2 px buffer at cloud boundaries.

| **Buffer** | **Cl. Shad. (Land)** | **Cl. Shad. (Water)** | **Water** | **Ice/Snow** | **Land** | **Flood.** |
|---|---|---|---|---|---|---|
| 0 px | 0.22 | 0.00 | 0.00 | 14.07 | 0.03 | 0.00 |
| 2 px | 0.10 | 0.00 | 0.00 | 14.24 | 0.03 | 0.00 |

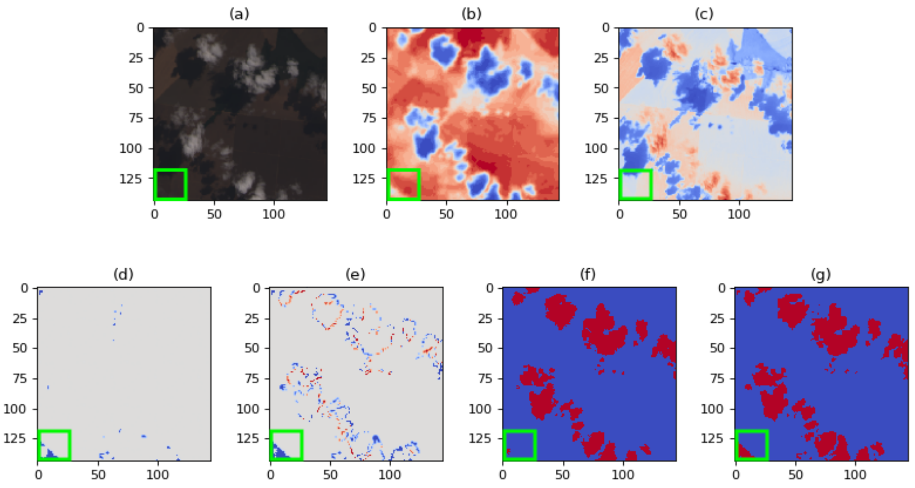


Figure 3-3: Random forest algorithm evaluated on an "easy" image segmentation example, with an optically thin cloud patch boxed in green. (a) Visible-spectrum image, (b) LWIR image, (c) SWIR image, (d) Difference between "truth" mask and mask generated by the random forest algorithm, excluding pixels within 2 px of a cloud boundary, (e) Difference between "truth" mask and mask generated by the random forest algorithm, (f) Mask generated by the random forest algorithm, (g) "Truth" mask.

In Figure 3-4, the random forest algorithm is applied to a “hard” image sample with overlapping snow and cloud pixels. The cloud mask generated by the random forest algorithm, shown in Figure 3-4(f), looks qualitatively very different from the “truth” mask shown in Figure 3-4(g). Interestingly, it seems as though the random forest algorithm is able to successfully classify the snow at the top of the image as non-cloud, but the random forest algorithm struggles to differentiate some pixels in the mountain valleys from cloud, perhaps because the random forest algorithm only examines a $3 \times 3$ window surrounding a pixel during classification and lacks broader spatial context. The random forest algorithm also misses the optically thin cloud patch boxed in green.

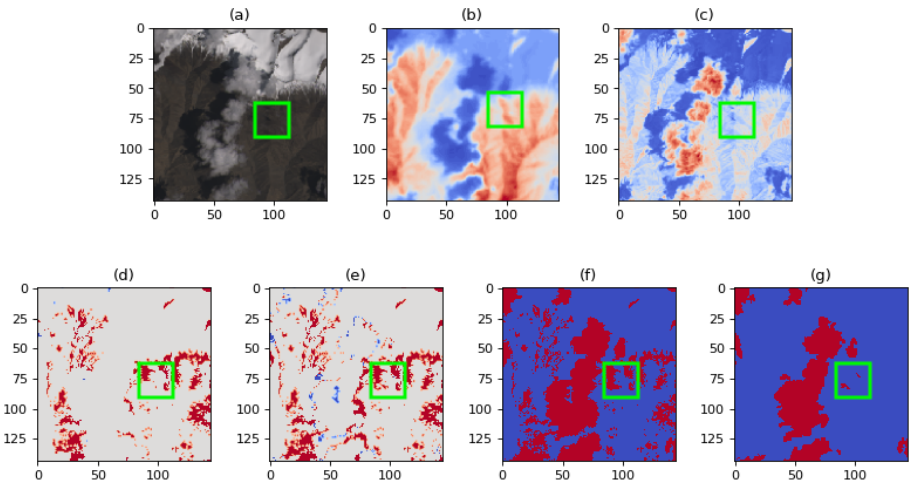


Figure 3-4: Random forest algorithm evaluated on a “hard” image segmentation example, with an optically thin cloud patch boxed in green. (a) Visible-spectrum image, (b) LWIR image, (c) SWIR image, (d) Difference between “truth” mask and mask generated by the random forest algorithm, excluding pixels within 2 px of a cloud boundary, (e) Difference between “truth” mask and mask generated by the random forest algorithm, (f) Mask generated by the random forest algorithm, (g) “Truth” mask.

Table 3.3 shows the performance of the random forest algorithm across all metrics. The random forest algorithm improves significantly on the performance of the luminosity thresholding algorithm across all metrics, and balances sensitivity and specificity much better than the luminosity thresholding algorithm.

Table 3.4 gives the ratio of the frequency of each terrain class amongst pixels

Table 3.3: Performance metrics detailed in §2.3.1 for the random forest algorithm on the cloud dataset, evaluated with no buffer and with a 2 px buffer at cloud boundaries.

| **Buffer** | **Acc.** | **Bal. Acc.** | **Sens.** | **Spec.** | **Prec.** | **Recall** | **F$_1$** | **IoU** |
|---|---|---|---|---|---|---|---|---|
| 0 px | 95.60% | 90.46% | 83.80% | 97.13% | 79.16% | 83.80% | 0.8141 | 0.6865 |
| **2 px** | **98.68%** | **97.91%** | **96.91%** | **98.91%** | **92.11%** | **96.91%** | **0.9445** | **0.8948** |

falsely identified as cloud by the random forest method to the frequency of that class amongst all pixels in the SPARCS dataset. The "land" terrain class is the most overrepresented amongst random forest false positives for a 2 px buffered evaluation. This quantitatively demonstrates the qualitative result from Figure 3-4 – that the random forest algorithm sometimes misidentifies land pixels, likely due to lack of spatial context, but can distinguish between clouds and snow fairly well.

Table 3.4: Ratios representing the frequency of each terrain class (cloud shadow over land, cloud shadow over water, water, ice/snow, land, and flooded) among pixels falsely identified as clouds by the random forest algorithm to the frequency of each terrain class among all non-cloud pixels in the SPARCS dataset, as explained in §2.3.1, evaluated with no buffer and with a 2 px buffer at cloud boundaries.

| **Buffer** | **Cl. Shad. (Land)** | **Cl. Shad. (Water)** | **Water** | **Ice/Snow** | **Land** | **Flood.** |
|---|---|---|---|---|---|---|
| 0 px | 1.63 | 0.78 | 0.14 | 1.03 | 1.05 | 0.00 |
| 2 px | 0.58 | 0.18 | 0.04 | 0.74 | 1.20 | 0.00 |

### 3.1.3 U-Net

Figure 3-5 shows the application of the U-Net to an "easy" image sample with no snow. The U-Net-generated cloud mask shown in Figure 3-5(f) looks qualitatively very similar to the "truth" mask shown in Figure 3-5(g). The similarity is further borne out by the 2 px buffer difference map shown in Figure 3-5(d), which shows almost no differences between the two masks. Unlike the luminosity thresholding algorithm (see Figure 3-1) and the random forest algorithm (see Figure 3-3), the

U-Net correctly identifies the optically thin cloud patch (boxed in green) as cloud, although it overestimates the size of the optically thin cloud patch.

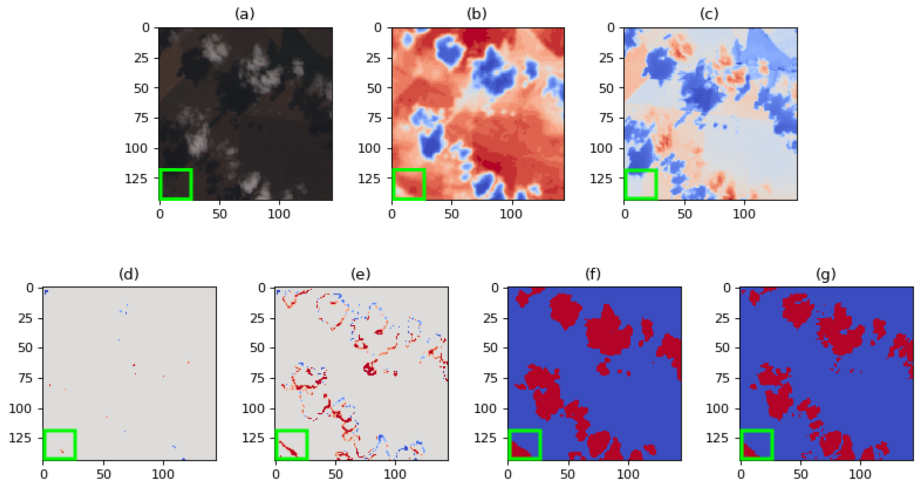


Figure 3-5: U-Net performance on an "easy" image segmentation example, with an optically thin cloud patch boxed in green. (a) Visible-spectrum image, (b) LWIR image, (c) SWIR image, (d) Difference between "truth" mask and U-Net-generated mask, excluding pixels within 2 px of a cloud boundary, (e) Difference between "truth" mask and U-Net-generated mask, (f) U-Net-generated mask, (g) "Truth" mask.

Figure 3-6 shows the application of the U-Net to a "hard" image sample with overlapping clouds and snow. The U-Net-generated cloud mask shown in Figure 3-6(f) looks qualitatively similar to the "truth" mask shown in Figure 3-6(g), except for the fact that the U-Net misses the optically thin cloud patch (boxed in green) and except for another small difference in the bottom right of the image. In the 2 px buffer difference map shown in Figure 3-6(d), only this difference and a few other scattered very small differences are visible between the two masks. The U-Net mask shown in 3-6(f) looks much closer to the truth mask than either the luminosity thresholding mask shown in 3-2(f) or the random forest mask shown in 3-4(f).

The U-Net metrics are given in Table 3.5. Although the U-Net has slightly lower sensitivity and recall than the random forest algorithm, it beats the random forest algorithm on every other metric.

The ratios shown in Table 3.6 show that with a 2 px buffer, the most overrepresented class amongst false positives (by far) is cloud shadow over land. This implies that most differences between U-Net-generated cloud masks and "truth" masks oc-

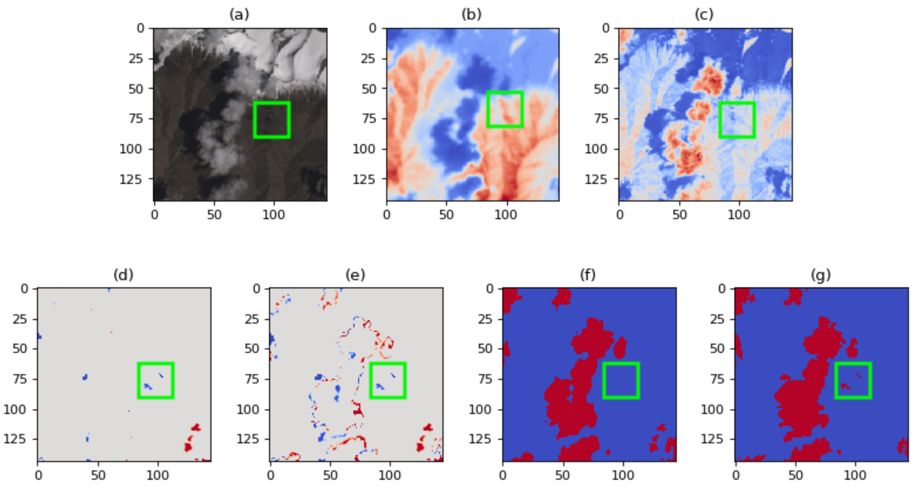


Figure 3-6: U-Net performance on a “hard” image segmentation example, with an optically thin cloud patch boxed in green. (a) Visible-spectrum image, (b) LWIR image, (c) SWIR image, (d) Difference between “truth” mask and U-Net-generated mask, excluding pixels within 2 px of a cloud boundary, (e) Difference between “truth” mask and U-Net-generated mask, (f) U-Net-generated mask, (g) “Truth” mask.

Table 3.5: Performance metrics detailed in §2.3.1 for the U-Net evaluated on the cloud dataset, evaluated with no buffer and with a 2 px buffer at cloud boundaries.

| **Buffer** | **Acc.** | **Bal. Acc.** | **Sens.** | **Spec.** | **Prec.** | **Recall** | **$F_1$** | **IoU** |
|---|---|---|---|---|---|---|---|---|
| 0 px | 96.59% | 91.82% | 85.62% | 98.02% | 84.88% | 85.62% | 0.8525 | 0.7429 |
| **2 px** | **99.41%** | **98.17%** | **96.54%** | **99.80%** | **98.46%** | **96.54%** | **0.9749** | **0.9511** |

cur near cloud boundaries, which fits the narrative told by the strong qualitative performance of the U-Net-generated cloud masks in Figures 3-5 and 3-6.

Table 3.6: Ratios representing the frequency of each terrain class (cloud shadow over land, cloud shadow over water, water, ice/snow, land, and flooded) among pixels falsely identified as clouds by the U-Net to the frequency of each terrain class among all non-cloud pixels in the SPARCS dataset, as explained in §2.3.1, evaluated with no buffer and with a 2 px buffer at cloud boundaries.

| **Buffer** | **Cl. Shad. (Land)** | **Cl. Shad. (Water)** | **Water** | **Ice/Snow** | **Land** | **Flood.** |
|---|---|---|---|---|---|---|
| 0 px | 3.28 | 0.59 | 0.15 | 1.21 | 0.85 | 0.00 |
| 2 px | 3.30 | 0.00 | 0.11 | 0.21 | 1.08 | 0.00 |

### 3.1.4 Dense U-Net

Figure 3-7 evaluates the dense U-Net on an "easy" image sample with no snow. The cloud mask generated by the dense U-Net (shown in Figure 3-7(f)) looks qualitatively very similar to the "truth" mask shown in Figure 3-7(g). This similarity is also demonstrated by the difference map shown in Figure 3-7(e) and the 2 px buffer difference map shown in Figure 3-7(d) – it is clear that most differences between the cloud mask generated by the dense U-Net and the "truth" mask occur within 2 px of a cloud edge and thus disappear in the buffered difference map.

Figure 3-8 evaluates the dense U-Net on a "hard" image sample with overlapping clouds and snow. As in Figure 3-7, the cloud mask generated by the dense U-Net looks qualitatively very similar to the "truth mask", including in the bottom right of the mask (where the U-Net-generated mask differed from the "truth" mask). However, the dense U-Net, like the U-Net, random forest algorithm, and luminosity thresholding algorithm, misidentifies the optically thin cloud patch boxed in green.

All performance metrics for the dense U-Net are given in Table 3.7. The dense U-Net shows a modest improvement in all metrics except specificity when compared to the U-Net (but the dense U-Net specificity is only 0.01% lower than the U-Net specificity).

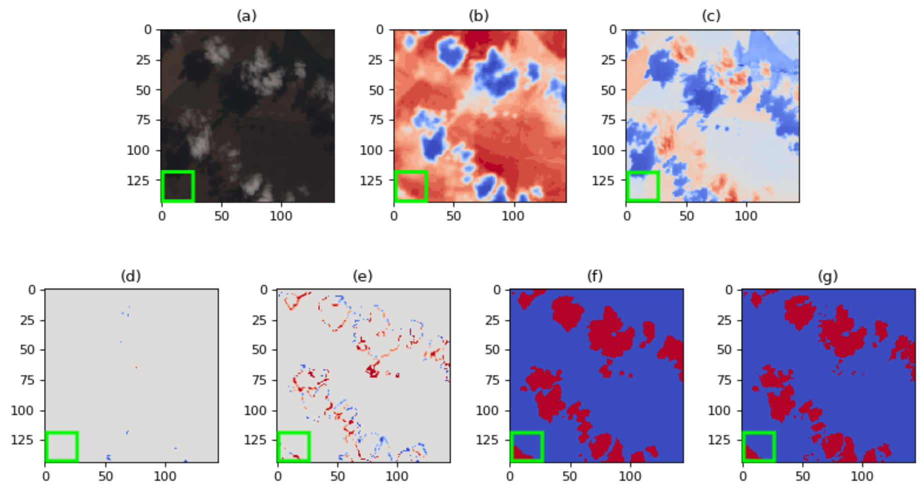


Figure 3-7: Dense U-Net evaluated on an “easy” image segmentation example, with an optically thin cloud patch boxed in green. (a) Visible-spectrum image, (b) LWIR image, (c) SWIR image, (d) Difference between “truth” mask and mask generated by the dense U-Net, excluding pixels within 2 px of a cloud boundary, (e) Difference between “truth” mask and mask generated by the dense U-Net, (f) Mask generated by the dense U-Net, (g) “Truth” mask.

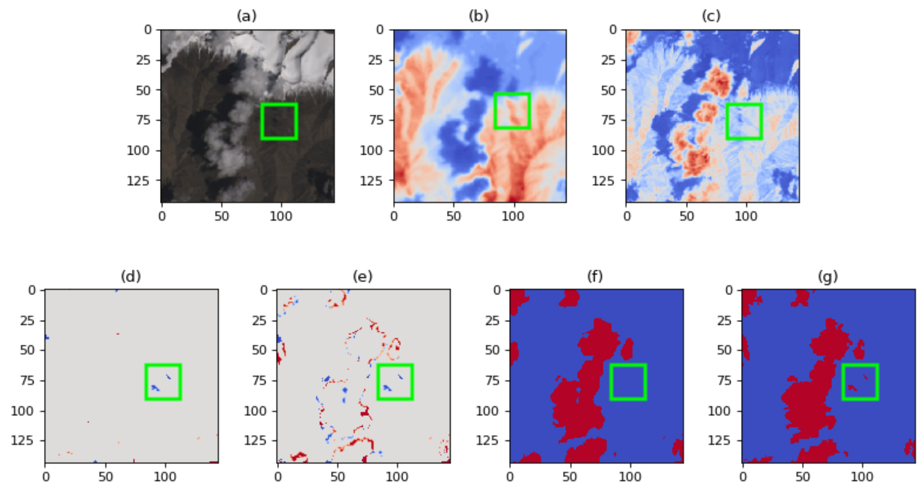


Figure 3-8: Dense U-Net evaluated on a “hard” image segmentation example, with an optically thin cloud patch boxed in green. (a) Visible-spectrum image, (b) LWIR image, (c) SWIR image, (d) Difference between “truth” mask and mask generated by the dense U-Net, excluding pixels within 2 px of a cloud boundary, (e) Difference between “truth” mask and mask generated by the dense U-Net, (f) Mask generated by the dense U-Net, (g) “Truth” mask.

Table 3.7: Performance metrics detailed in §2.3.1 for the dense U-Net evaluated on the cloud dataset, evaluated with no buffer and with a 2 px buffer at cloud boundaries.

| **Buffer** | **Acc.** | **Bal. Acc.** | **Sens.** | **Spec.** | **Prec.** | **Recall** | $\mathbf{F_1}$ | **IoU** |
|---|---|---|---|---|---|---|---|---|
| 0 px | 96.77% | 93.11% | 88.36% | 97.86% | 84.31% | 88.36% | 0.8629 | 0.7588 |
| **2 px** | **99.43%** | **98.34%** | **96.89%** | **99.79%** | **98.47%** | **96.89%** | **0.9768** | **0.9546** |

The terrain class ratios for pixels falsely identified as clouds by the dense U-Net are given in Table 3.8. Cloud shadows over land are heavily overrepresented among false positives identified by the dense U-Net, implying that most false positives occur at cloud edges or near clouds.

Table 3.8: Ratios representing the frequency of each terrain class (cloud shadow over land, cloud shadow over water, water, ice/snow, land, and flooded) among pixels falsely identified as clouds by the dense U-Net to the frequency of each terrain class among all non-cloud pixels in the SPARCS dataset, as explained in §2.3.1, evaluated with no buffer and with a 2 px buffer at cloud boundaries.

| **Buffer** | **Cl. Shad. (Land)** | **Cl. Shad. (Water)** | **Water** | **Ice/Snow** | **Land** | **Flood.** |
|---|---|---|---|---|---|---|
| 0 px | 3.21 | 0.62 | 0.14 | 1.33 | 0.85 | 0.01 |
| 2 px | 3.61 | 0.00 | 0.12 | 0.45 | 1.04 | 0.05 |

Overall, the U-Net and dense U-Net perform very well on cloud segmentation, especially when a 2 px buffer is taken into account at cloud boundaries. Nevertheless, the $C_8$-equivariant models outperform the U-Net and dense U-Net on cloud segmentation, especially within 2 px of the cloud boundaries, and so can provide important value for missions with very high segmentation accuracy requirements.

### 3.1.5 $C_8$-Equivariant U-Net

Figure 3-9 shows the performance of the $C_8$-equivariant U-Net on an "easy" image sample. The cloud mask generated by the $C_8$-equivariant U-Net looks qualitatively very similar to the "truth" mask, and not many differences can be seen in the 2 px buffer difference map.

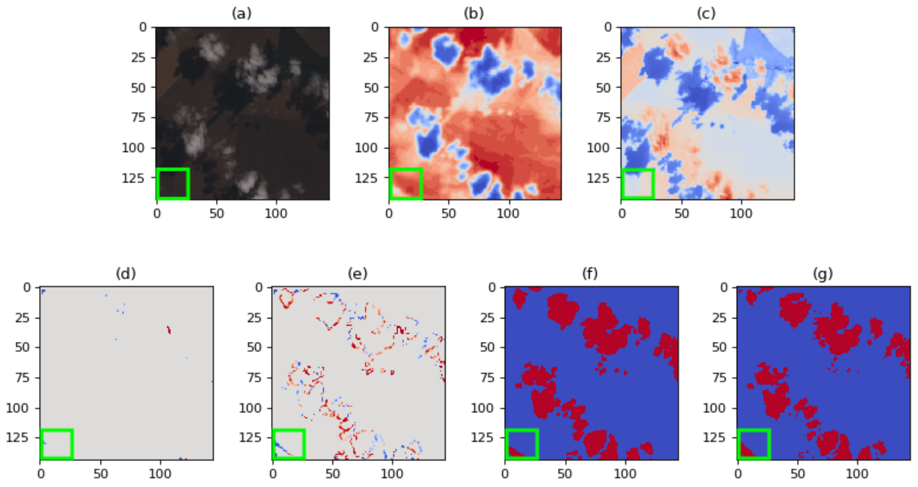


Figure 3-9: $C_8$-equivariant U-Net evaluated on an "easy" image segmentation example, with an optically thin cloud patch boxed in green. (a) Visible-spectrum image, (b) LWIR image, (c) SWIR image, (d) Difference between "truth" mask and mask generated by the $C_8$-equivariant U-Net, excluding pixels within 2 px of a cloud boundary, (e) Difference between "truth" mask and mask generated by the $C_8$-equivariant U-Net, (f) Mask generated by the $C_8$-equivariant U-Net, (g) "Truth" mask.

Figure 3-10 shows the performance of the $C_8$-equivariant U-net on a "hard" image sample with clouds overlapping snow. The cloud mask generated by the $C_8$-equivariant U-Net misidentifies a few cloud patches as non-cloud, especially near the cloud boundaries. These false negative patches can be seen in the 2 px buffer difference map (Figure 3-10(d)). These misidentified patches include the optically thin cloud patch boxed in green.

The metrics for the $C_8$-equivariant U-Net are given in Table 3.9. The $C_8$-equivariant U-Net improves over the U-Net on all metrics, and also notably improves over the dense U-Net on all metrics, showing that $C_8$-equivariance leads to bigger performance gains over adopting a dense architecture. Although the dense U-Net and the $C_8$-equivariant U-Net perform similarly when evaluated with no buffer, the $C_8$-equivariant U-Net more than doubles the gains of the dense U-Net over the U-Net when a 2 px buffer is taken into account, achieving an $F_1$ score of 0.9797. The dense U-Net achieves an $F_1$ score of 0.9768 when a 2 px buffer is taken into account, and the U-Net achieves an $F_1$ score of 0.9749 when a 2 px buffer is taken into account.

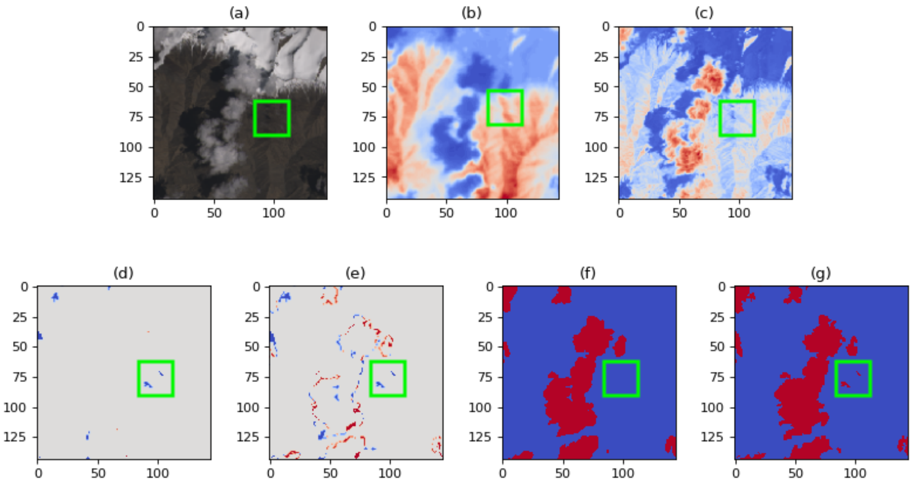


Figure 3-10: $C_8$-equivariant U-Net evaluated on a "hard" image segmentation example, with an optically thin cloud patch boxed in green. (a) Visible-spectrum image, (b) LWIR image, (c) SWIR image, (d) Difference between "truth" mask and mask generated by the $C_8$-equivariant U-Net, excluding pixels within 2 px of a cloud boundary, (e) Difference between "truth" mask and mask generated by the $C_8$-equivariant U-Net, (f) Mask generated by the $C_8$-equivariant U-Net, (g) "Truth" mask.

Table 3.9: Performance metrics detailed in §2.3.1 for the $C_8$-equivariant U-Net evaluated on the cloud dataset, evaluated with no buffer and with a 2 px buffer at cloud boundaries.

| **Buffer** | **Acc.** | **Bal. Acc.** | **Sens.** | **Spec.** | **Prec.** | **Recall** | **F$_1$** | **IoU** |
|---|---|---|---|---|---|---|---|---|
| 0 px | 96.87% | 91.99% | 85.64% | 98.34% | 87.01% | 85.64% | 0.8632 | 0.7593 |
| **2 px** | **99.54%** | **98.51%** | **97.18%** | **99.84%** | **98.78%** | **97.18%** | **0.9797** | **0.9602** |

The false positive terrain class ratios for the $C_8$-equivariant U-Net are given in Table 3.10. Cloud shadows over land are clearly overrepresented amongst false positives, like the U-Net and dense U-Net.

Table 3.10: Ratios representing the frequency of each terrain class (cloud shadow over land, cloud shadow over water, water, ice/snow, land, and flooded) among pixels falsely identified as clouds by the $C_8$-equivariant U-Net to the frequency of each terrain class among all non-cloud pixels in the SPARCS dataset, as explained in §2.3.1, evaluated with no buffer and with a 2 px buffer at cloud boundaries.

| **Buffer** | **Cl. Shad. (Land)** | **Cl. Shad. (Water)** | **Water** | **Ice/Snow** | **Land** | **Flood.** |
|---|---|---|---|---|---|---|
| 0 px | 3.05 | 0.68 | 0.17 | 1.52 | 0.84 | 0.00 |
| 2 px | 2.45 | 0.00 | 0.10 | 0.91 | 1.07 | 0.00 |

## 3.1.6 $C_8$-Equivariant Dense U-Net

Figure 3-11 evaluates the $C_8$-equivariant dense U-Net on an "easy" image sample, on which it performs extremely well. False positives at the cloud edges can be seen in the difference map in Figure 3-11(e), but these disappear in the 2 px buffer difference map (Figure 3-11(d)), indicating that these false positives are only occur within 1-2 px of cloud boundaries.

Figure 3-12 evaluates the $C_8$-equivariant dense U-Net on a "hard" image sample. False positives at the cloud edges can again be seen in the difference map in Figure 3-12(e). Although these false positives mostly disappear in the 2 px buffer difference map in Figure 3-12(d), a few pockets of false positives and false negatives remain, including an optically thin cloud patch (boxed in green) which is misidentified as non-cloud.

The performance metrics for the $C_8$-equivariant dense U-Net are given in Table 3.11. The $C_8$-equivariant dense U-Net improves over the next-best model, the $C_8$-equivariant U-Net, on all metrics except for accuracy on specificity, where it ties the performance of the $C_8$-equivariant U-Net.

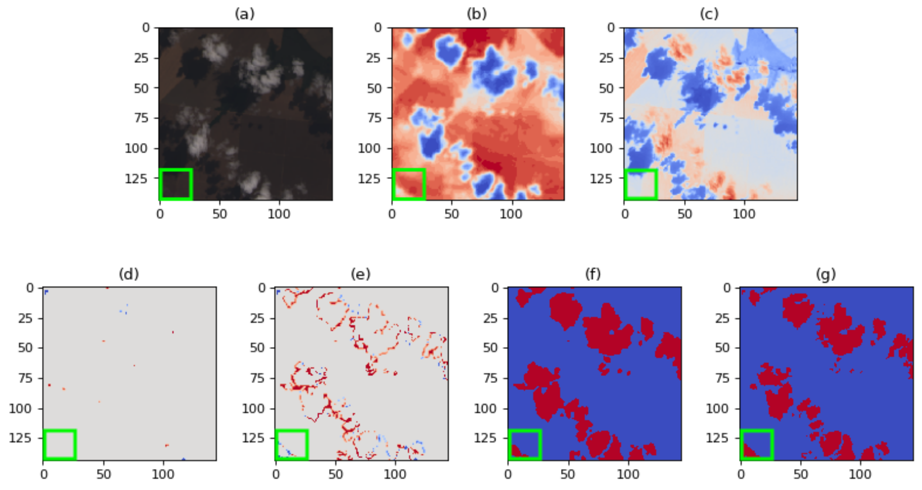


Figure 3-11: $C_8$-equivariant dense U-Net evaluated on an "easy" image segmentation example, with an optically thin cloud patch boxed in green. (a) Visible-spectrum image, (b) LWIR image, (c) SWIR image, (d) Difference between "truth" mask and mask generated by the $C_8$-equivariant dense U-Net, excluding pixels within 2 px of a cloud boundary, (e) Difference between "truth" mask and mask generated by the $C_8$-equivariant dense U-Net, (f) Mask generated by the $C_8$-equivariant dense U-Net, (g) "Truth" mask.

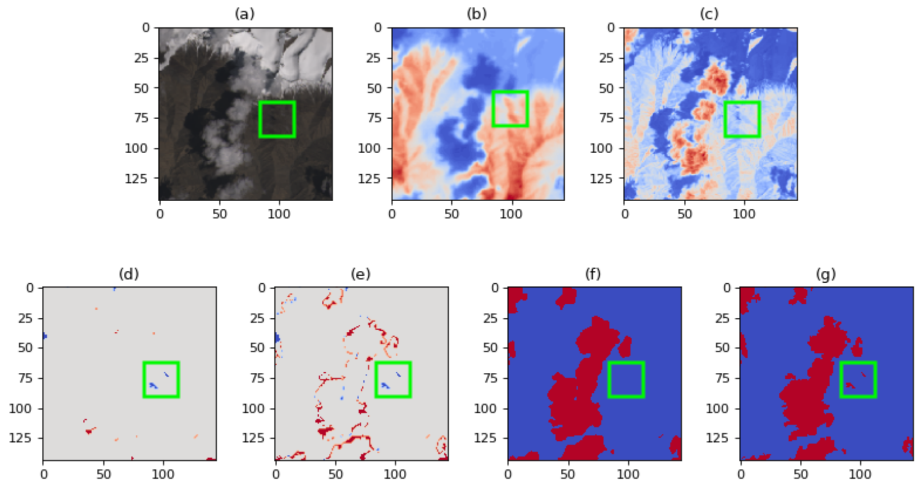


Figure 3-12: $C_8$-equivariant dense U-Net evaluated on a "hard" image segmentation example, with an optically thin cloud patch boxed in green. (a) Visible-spectrum image, (b) LWIR image, (c) SWIR image, (d) Difference between "truth" mask and mask generated by the $C_8$-equivariant dense U-Net, excluding pixels within 2 px of a cloud boundary, (e) Difference between "truth" mask and mask generated by the $C_8$-equivariant dense U-Net, (f) Mask generated by the $C_8$-equivariant dense U-Net, (g) "Truth" mask.

Table 3.11: Performance metrics detailed in §2.3.1 for the $C_8$-equivariant dense U-Net evaluated on the cloud dataset, evaluated with no buffer and with a 2 px buffer at cloud boundaries.

| **Buffer** | **Acc.** | **Bal. Acc.** | **Sens.** | **Spec.** | **Prec.** | **Recall** | **F$_1$** | **IoU** |
|---|---|---|---|---|---|---|---|---|
| 0 px | 97.01% | 93.16% | 88.17% | 98.16% | 86.15% | 88.17% | 0.8715 | 0.7722 |
| **2 px** | **99.54%** | **98.58%** | **97.32%** | **99.84%** | **98.82%** | **97.32%** | **0.9806** | **0.9620** |

The terrain ratios for the $C_8$-equivariant dense U-Net are given in Table 3.12. Interestingly, pixels in the ice/snow terrain class are overrepresented among false positives identified by the $C_8$-equivariant dense U-Net, and pixels in the land terrain class are underrepresented. As expected, cloud shadows and ice/snow pixels are both overrepresented, as these terrain classes should be the hardest to distinguish from clouds, but it is unclear why the $C_8$-equivariant dense U-Net has a relative advantage on land pixels versus snow/ice pixels when compared to the other deep learning methods. It is possible that this result reflects improved performance distinguishing optically thin clouds from land when compared to the other deep learning models.

Table 3.12: Ratios representing the frequency of each terrain class (cloud shadow over land, cloud shadow over water, water, ice/snow, land, and flooded) among pixels falsely identified as clouds by the $C_8$-equivariant dense U-Net to the frequency of each terrain class among all non-cloud pixels in the SPARCS dataset, as explained in §2.3.1, evaluated with no buffer and with a 2 px buffer at cloud boundaries.

| **Buffer** | **Cl. Shad. (Land)** | **Cl. Shad. (Water)** | **Water** | **Ice/Snow** | **Land** | **Flood.** |
|---|---|---|---|---|---|---|
| 0 px | 3.23 | 0.76 | 0.16 | 1.60 | 0.82 | 0.00 |
| 2 px | 3.30 | 0.00 | 0.15 | 1.36 | 0.97 | 0.00 |

### 3.1.7 Summary

#### Segmentation Performance

We first qualitatively compare the performance of the models presented in §3.1.1-3.1.6 by applying each model to an "easy" cloud segmentation example with no snow, ice, cold water, or bright land pixels and visually comparing the masks generated by each model, which are shown in Figure 3-13. The luminosity thresholding algorithm (mask shown in Figure 3-13(e)) and random forest algorithm (mask shown in Figure 3-13(f)) perform much better on this input than on the "hard" cloud segmentation example shown in Figure 3-14, highlighting the strengths and limitations of rule-based image segmentation algorithms that take only limited spatial context into account.

Nevertheless, both the luminosity thresholding and random forest algorithms miss the optically thin cloud patch boxed in green. This cloud patch is wispy and barely perceptible in the false-color images used to generate the cloud masks for the SPARCS dataset, which remap Landsat B6 (SWIR 1) to red, Landsat B5 (near infrared) to green, and Landsat B4 (red) to blue [25]. In the downsampled images we use for training, which contain data from Landsat B2-4, B6, and B10, this cloud patch is difficult to see in B6 (SWIR 1) and completely invisible in B2-4 and B10. In contrast, all four deep learning algorithms (masks shown in Figure 3-13(g-j)) successfully capture this optically thin cloud patch to some degree. The $C_8$-equivariant U-Net (mask shown in Figure 3-13(i)) and the $C_8$-equivariant dense U-Net (mask shown in Figure 3-13(j)) qualitatively classify the optically thin cloud boxed in green most accurately; the U-Net (mask shown in Figure 3-13(g)) and dense U-Net (mask shown in Figure 3-13(h)) overestimate the size of this cloud patch.

Figure 3-14 shows the six models presented in §3.1.1-3.1.6 applied to a "hard" cloud segmentation example with overlapping snow and cloud. The four cloud masks generated by the U-Net (Figure 3-14(g)), the dense U-Net (Figure 3-14(h)), the $C_8$-equivariant U-Net (Figure 3-14(i)), and the $C_8$-equivariant dense U-Net (Figure 3-14(j)) all look qualitatively very similar to the "truth" mask (Figure 3-14(d)), except for the missing patch of cloud boxed in green. This patch does not appear to be bright

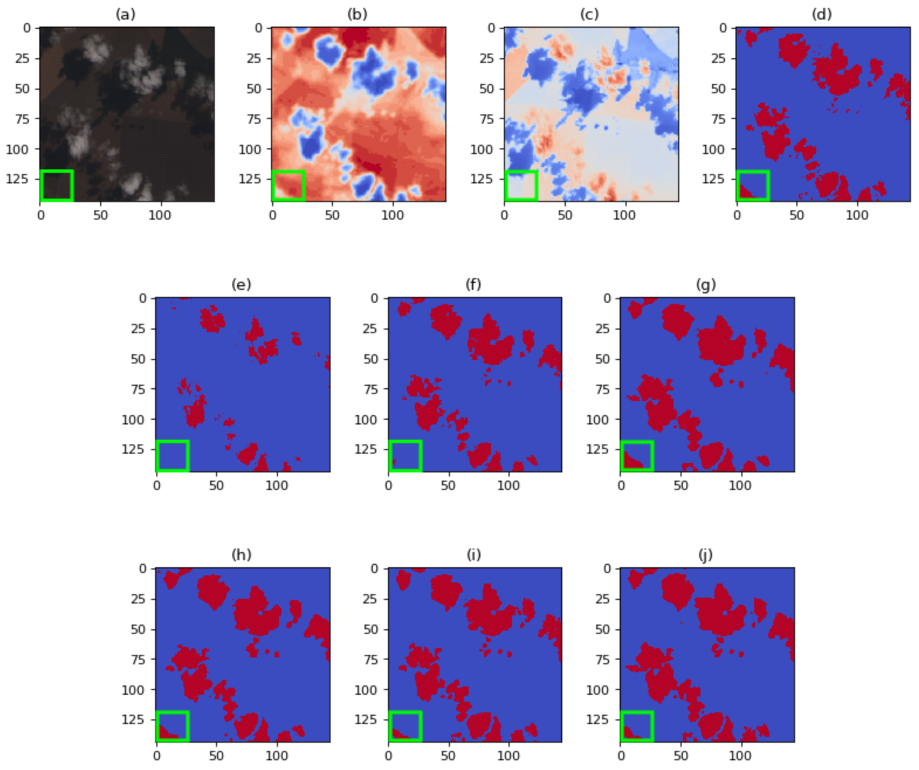


Figure 3-13: Luminosity thresholding algorithm, random forest algorithm, U-Net, dense U-Net, $C_8$-equivariant U-Net and $C_8$-equivariant dense U-Net evaluated on an “easy” image segmentation sample, with an optically thin cloud patch boxed in green. (a) Visible-spectrum image input, (b) LWIR input, (c) SWIR input, (d) “Truth” mask, (e) Mask generated by the luminosity thresholding algorithm, (f) Mask generated by the random forest algorithm, (g) Mask generated by the U-Net, (h) Mask generated by the dense U-Net, (i) Mask generated by the $C_8$-equivariant U-Net, (j) Mask generated by the $C_8$-equivariant dense U-Net.

in either the VIS image input (Figure 3-14(a)) or the SWIR image input (Figure 3-14(c)), nor does it appear to be cold in the LWIR image input (Figure 3-14(b)). As such, this cloud patch is essentially invisible to the cloud segmentation models. Close inspection of Landsat data from all bands reveals that this cloud patch is barely perceptible in the cirrus band (B9) and the high-resolution panchromatic band (B8), and invisible in all other bands.

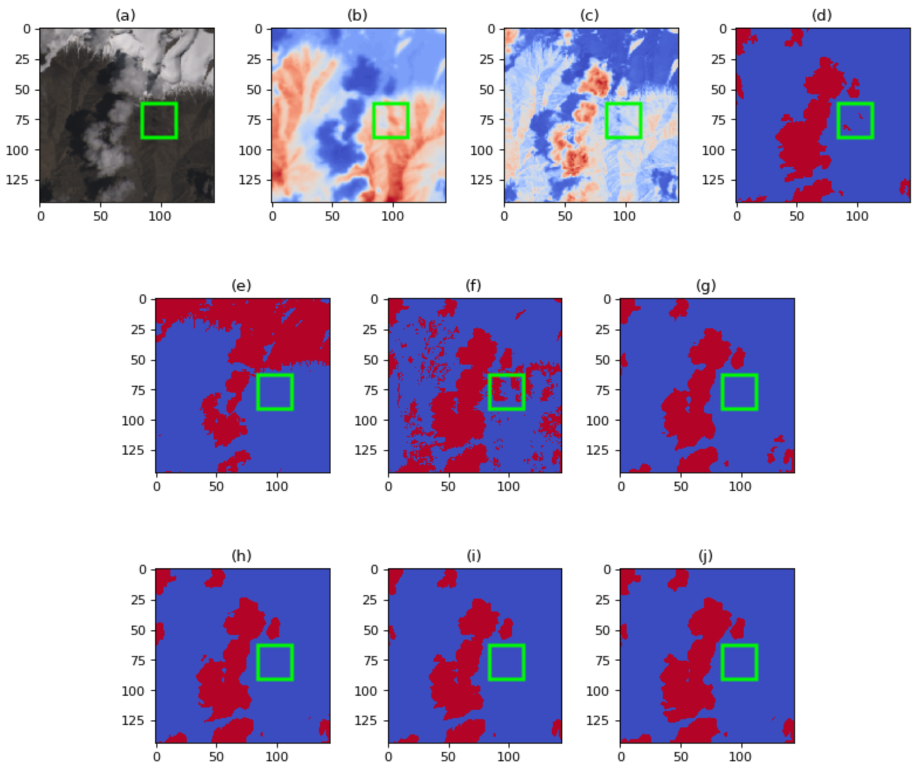


Figure 3-14: Luminosity thresholding algorithm, random forest algorithm, U-Net, dense U-Net, $C_8$-equivariant U-Net and $C_8$-equivariant dense U-Net evaluated on a "hard" image segmentation sample, with an optically thin cloud patch boxed in green. (a) Visible-spectrum image input, (b) LWIR input, (c) SWIR input, (d) "Truth" mask, (e) Mask generated by the luminosity thresholding algorithm, (f) Mask generated by the random forest algorithm, (g) Mask generated by the U-Net, (h) Mask generated by the dense U-Net, (i) Mask generated by the $C_8$-equivariant U-Net, (j) Mask generated by the $C_8$-equivariant dense U-Net.

The performance metrics for the models presented in §3.1.1-3.1.6, evaluated on the SPARCS dataset without a buffer, are given in Table 3.13, with the best performance on each metric bolded. The dense U-Net has the best sensitivity and recall, and the $C_8$-equivariant U-Net has the best specificity and precision, but the $C_8$-equivariant dense U-Net is nearly as good on each of these metrics, and has the best accuracy, balanced accuracy, $F_1$ score, and intersection-over-union. This demonstrates the superior performance of the $C_8$-equivariant dense U-Net, the $C_8$-equivariant U-Net's bias toward specificity over sensitivity, and the dense U-Net's bias toward sensitivity over specificity when the models are evaluated without a buffer.

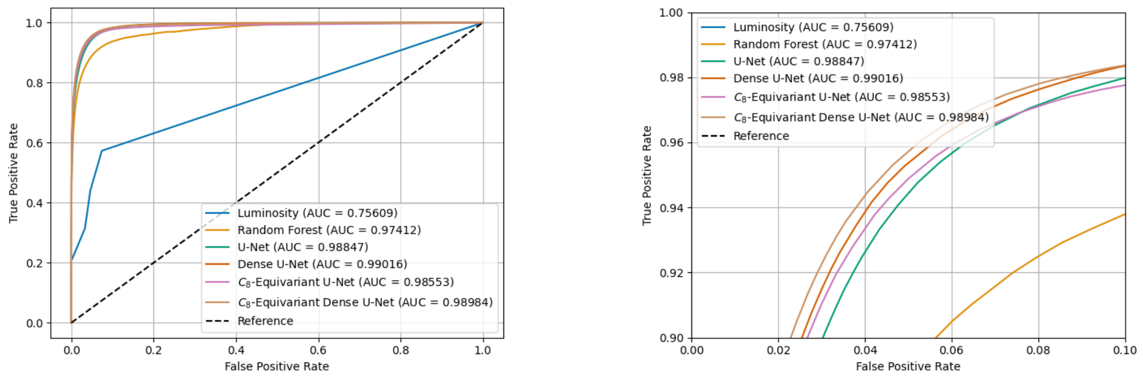
Table 3.13: Performance metrics detailed in §2.3.1 for the luminosity thresholding algorithm, random forest algorithm, U-Net, dense U-Net, $C_8$-equivariant U-Net, and $C_8$-equivariant dense U-Net evaluated on the modified SPARCS dataset, evaluated without using a buffer at cloud boundaries.

| **Model** | **Acc.** | **Bal. Acc.** | **Sens.** | **Spec.** | **Prec.** | **Recall** | **F$_1$** | **IoU** |
|---|---|---|---|---|---|---|---|---|
| Luminosity Thresholding | 89.19% | 64.08% | 31.45% | 96.71% | 55.40% | 31.45% | 0.4013 | 0.2510 |
| Random Forest | 95.60% | 90.46% | 83.80% | 97.13% | 79.16% | 83.80% | 0.8141 | 0.6865 |
| U-Net | 96.59% | 91.82% | 85.62% | 98.02% | 84.88% | 85.62% | 0.8525 | 0.7429 |
| Dense U-Net | 96.77% | 93.11% | **88.36%** | 97.86% | 84.31% | **88.36%** | 0.8629 | 0.7588 |
| $C_8$-Equivariant U-Net | 96.87% | 91.99% | 85.64% | **98.34%** | **87.01%** | 85.64% | 0.8632 | 0.7593 |
| $C_8$-Equivariant Dense U-Net | **97.01%** | **93.16%** | 88.17% | 98.16% | 86.15% | 88.17% | **0.8715** | **0.7722** |

Figure 3-15 shows the receiver-operating characteristic (ROC) curve for each of the models presented in §3.1.1-3.1.6, evaluated without a buffer. The curve representing the $C_8$-equivariant dense U-Net is consistently closest to the $(0, 1)$ point representing perfect classification, demonstrating its superior performance. The next closest curve represents the dense U-Net, then the $C_8$-equivariant U-Net, then the U-Net, then the random forest algorithm, and finally the curve furthest from $(0, 1)$ represents the luminosity thresholding algorithm. This order is consistent with the order of the $F_1$

scores presented in Table 3.13, except for the fact that the ROC curve representing the dense U-Net is closer to $(0, 1)$ than the curve representing the $C_8$-equivariant U-Net. This means that with a cloud threshold other than 0.5, the dense U-Net might outperform the $C_8$-equivariant U-Net.

Also, the gap between the ROC curves representing the deep learning models presented in §3.1.3-3.1.6 and the curve representing the random forest algorithm is very large, and the gap between the curve representing the random forest algorithm and the curve representing the luminosity thresholding algorithm is even larger. This reflects the large performance gains made by increasing model complexity and taking larger amounts of spatial context into account during classification, and fits with the results presented in Tables 3.13 and 3.14.



(a) Original view.

(b) Zoomed-in view.

Figure 3-15: Receiver-operating characteristic (ROC) curves for the luminosity thresholding algorithm, random forest algorithm, U-Net, dense U-Net, $C_8$-equivariant U-Net, and $C_8$-equivariant dense U-Net, evaluated on the modified SPARCS dataset without using a buffer at cloud boundaries.

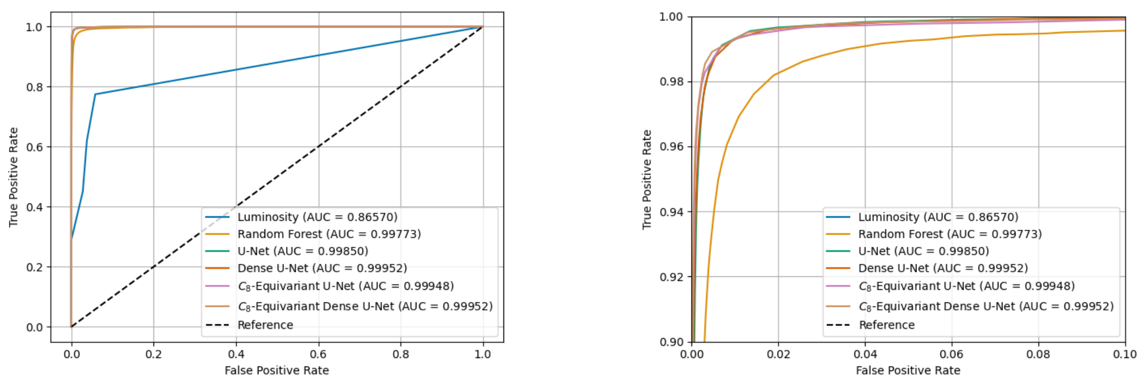
The performance metrics for all models on the cloud dataset (with a 2 px buffer) are given in Table 3.14, with the best performance on each metric bolded. The $F_1$ scores for the four deep learning algorithms are notably closer when a 2 px buffer is taken into account than when no buffer is used (see Table 3.13). This indicates that some of the performance gains made by the $C_8$-equivariant dense U-Net can be attributed to better discrimination between cloud and non-cloud pixels at cloud boundaries. These cloud discrimination gains disappear if a 2 px buffer at cloud boundaries is used when evaluating performance. Nevertheless, it is clear that the $C_8$-equivariant dense U-Net demonstrates the best cloud segmentation performance even when a 2 px buffer is used, outperforming the U-Net, dense U-Net, and $C_8$-equivariant U-Net by 0.0057, 0.0038, and 0.0009, respectively, in terms of $F_1$ score. In fact, the $C_8$-equivariant dense U-Net performs the best on every quantitative metric evaluated when a 2 px buffer is used, as shown in Table 3.14.

Table 3.14: Performance metrics detailed in §2.3.1 for the luminosity thresholding algorithm, random forest algorithm, U-Net, dense U-Net, $C_8$-equivariant U-Net, and $C_8$-equivariant dense U-Net evaluated on the modified SPARCS dataset, evaluated using a 2 px buffer at cloud boundaries.

| **Model** | **Acc.** | **Bal. Acc.** | **Sens.** | **Spec.** | **Prec.** | **Recall** | **$F_1$** | **IoU** |
|---|---|---|---|---|---|---|---|---|
| Luminosity Thresholding | 92.22% | 70.16% | 43.18% | 97.14% | 60.20% | 43.18% | 0.5029 | 0.3359 |
| Random Forest | 98.68% | 97.91% | 96.91% | 98.91% | 92.11% | 96.91% | 0.9445 | 0.8948 |
| U-Net | 99.41% | 98.17% | 96.54% | 99.80% | 98.46% | 96.54% | 0.9749 | 0.9511 |
| Dense U-Net | 99.43% | 98.34% | 96.89% | 99.79% | 98.47% | 96.89% | 0.9768 | 0.9546 |
| Rotationally Equivariant U-Net | **99.54%** | 98.51% | 97.18% | **99.84%** | 98.78% | 97.18% | 0.9797 | 0.9602 |
| Rotationally Equivariant Dense U-Net | **99.54%** | **98.58%** | **97.32%** | **99.84%** | **98.82%** | **97.32%** | **0.9806** | **0.9620** |

Figure 3-16 shows the ROC curve for each of the models presented in §3.1.1-3.1.6 when evaluated with a 2 px buffer at the cloud boundaries. The curves representing the deep learning models presented in §3.1.3-3.1.6 are clustered much more closely than when no buffer is considered (see Figure 3-15). This reflects the fact that some of the performance gains made by the $C_8$-equivariant and dense models come from better classification at the cloud boundaries, and these gains are not reflected when a 2 px buffer is considered at the cloud boundaries. Nevertheless, the $C_8$-equivariant dense U-Net and the dense U-Net have the highest area under the ROC curve (AUC), followed by the $C_8$-equivariant U-Net, reflecting the order of the curves visible in Figure 3-15.

Although the $C_8$-equivariant dense U-Net and the dense U-Net are tied for the highest AUC, the $C_8$-equivariant dense U-Net outperforms the dense U-Net except for a small region where sensitivity is high and specificity is low, and the $C_8$-equivariant dense U-Net gets visibly closer to $(0, 1)$ than any other classifier, reflecting its superior performance. Finally, the curves representing the deep learning models presented in §3.1.3-3.1.6 continue to significantly outperform the curve representing the random forest classifier, which in turn significantly outperforms the curve representing the luminosity thresholding classifier, regardless of whether a 2 px buffer at the cloud boundaries is used.



(a) Original view.

(b) Zoomed-in view.

Figure 3-16: Receiver-operating characteristic (ROC) curves for the luminosity thresholding algorithm, random forest algorithm, U-Net, dense U-Net, $C_8$-equivariant U-Net, and $C_8$-equivariant dense U-Net, evaluated on the modified SPARCS dataset using a 2 px buffer at cloud boundaries.

## Resource Utilization

Table 3.15 shows the resource utilization of all models when classifying a single image using both a CPU backend and GPU backend. Our luminosity thresholding and random forest algorithms do not support a GPU backend, so only results for a CPU backend are reported. As expected, the luminosity thresholding algorithm has the fastest inference time with a CPU backend due to its simplicity, and the random forest algorithm has the second fastest inference time with a CPU backend. The U-Net and $C_8$-equivariant U-Net require less memory and have faster inference time than the dense U-Net and $C_8$-equivariant dense U-Net, respectively, when using a GPU backend. This is likely because the increased number of concatenations in the dense models reduces the number of opportunities to combine tensor operations when using a GPU backend.

However, the dense U-Net and $C_8$-equivariant dense U-Net classify images more quickly than the U-Net and $C_8$-equivariant U-Net, respectively, when using a CPU backend, likely because the dense models use fewer parameters overall, as shown in Table 3.16. Finally, the $C_8$-equivariant models classify images more slowly than their non-equivariant equivalents regardless of backend, despite using less GPU memory and having fewer parameters overall. The $C_8$-equivariant models are 2 to 3 times slower than their non-equivariant equivalents with a GPU backend, but are only 1.1-1.3 times slower with a CPU backend. It seems likely that this is because of overhead added by the `e2cnn` library related to checking group representations, and it is possible that exporting the fully trained $C_8$-equivariant models to pure PyTorch equivalents would speed up inference.

Table 3.16 shows the size of the saved model for all algorithms, and also shows the number of parameters and the number of trainable parameters for the deep learning models. Notably, the random forest algorithm requires the most storage by far – this is one typical and major drawback of data-driven methods. Also notable is the fact that the $C_8$-equivariant dense U-Net has fewer parameters and a smaller total model size than all the other deep learning models. This demonstrates that the $C_8$-equivariant dense U-Net is able to outperform other models despite its lower model complexity and fewer total parameters by encoding rotation equivariance into its layers and by using dense blocks to improve gradient propagation and trainability.

Table 3.15: Peak memory usage over 100 single-image classifications and average inference time over 1000 single-image classifications using both a CPU (Intel Xeon) and GPU (Nvidia K80) backend for the luminosity thresholding algorithm, random forest algorithm, U-Net, dense U-Net, $C_8$-equivariant U-Net, and $C_8$-equivariant dense U-Net on the modified SPARCS dataset.

| **Model** | **GPU Mem. (MiB)** | **CPU Mem. (KiB)** | **GPU Inf. Time (s)** | **CPU Inf. Time (s)** |
|---|---|---|---|---|
| Luminosity Thresholding | – | 344.9 | – | 0.0003 |
| Random Forest | – | 1445.8 | – | 0.1136 |
| U-Net | 464.9 | 127.0 | 0.0092 | 0.3698 |
| Dense U-Net | 517.4 | 103.6 | 0.0150 | 0.1801 |
| $C_8$-Equivariant U-Net | 448.3 | 101.2 | 0.0185 | 0.3926 |
| $C_8$-Equivariant Dense U-Net | 503.5 | 144.8 | 0.0484 | 0.2281 |

Table 3.16: Saved model size, total number of parameters, and number of trainable parameters for the luminosity thresholding algorithm, random forest algorithm, U-Net, dense U-Net, $C_8$-equivariant U-Net and $C_8$-equivariant dense U-Net trained on the modified SPARCS dataset.

| **Model** | **Model Size** | **Total Parameters** | **Trainable Parameters** |
|---|---|---|---|
| Luminosity Thresholding | 4 KB | – | – |
| Random Forest | 1.3 GB | – | – |
| U-Net | 295.9 MB | $2.46 \times 10^7$ | $2.46 \times 10^7$ |
| Dense U-Net | 32.2 MB | $2.64 \times 10^6$ | $2.64 \times 10^6$ |
| $C_8$-Equivariant U-Net | 123.9 MB | $2.10 \times 10^6$ | $2.10 \times 10^6$ |
| $C_8$-Equivariant Dense U-Net | 14.6 MB | $2.93 \times 10^5$ | $2.90 \times 10^5$ |

## 3.2 Evaluating Different Combinations of Spectral Bands

On our modified version of the SPARCS dataset, we analyze the performance of the $C_8$-equivariant dense U-Net described in §2.2.6 trained on visible-spectrum Landsat 8 data only, visible-spectrum data augmented with data from the LWIR 1 band, visible-spectrum data augmented with data from the SWIR 1 band, and visible-spectrum data augmented with data from the LWIR 1 and SWIR 1 bands. We make comparisons between the models trained on these four training sets and make recommendations for bands to prioritize when designing or selecting instruments for resource-constrained missions.

Qualitatively, we evaluate each model by visually comparing model-generated masks to "truth" masks and multispectral model input for two different example images. The first example image, which we refer to as the "easy" image sample, is part of the image with a root filename of `‘‘LC82210662014229LGN00_18’’` in the SPARCS test set [25]. The second example image, which we refer to as the "hard" image sample, is also from the SPARCS test set and is part of the image with root filename `‘‘LC81480352013195LGN00_32’’` [25]. We quantitatively evaluate each model by calculating the metrics described in §2.3.1 and the ratios of the frequency of each terrain class among false positives to the frequency of each terrain class in the dataset as a whole, as described in §2.3.1. Finally, we measure the number of parameters, saved model size, peak memory usage, and inference time of each algorithm in order to evaluate model complexity and resource usage.

### 3.2.1 Visible-Spectrum

In Figure 3-17, we evaluate the $C_8$-equivariant U-Net trained on visible-spectrum data only on an "easy" image segmentation example. The model-generated mask shown in Figure 3-17(f) looks similar to the "truth" mask shown in Figure 3-17(g), except for the fact that it is missing a cloud patch in the bottom left (boxed in green). This cloud patch appears cold in the LWIR band (as shown in Figure 3-17(b), but does not appear visually distinct from the other land pixels in the visible-spectrum band (shown in Figure 3-17(a)). The model appears to miss this optically thin cloud patch.

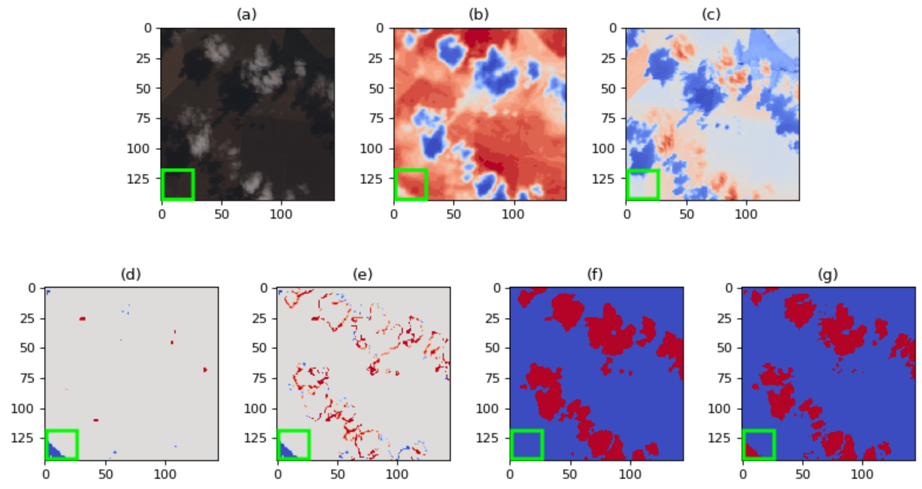


Figure 3-17: $C_8$-equivariant dense U-Net trained on visible-spectrum input only evaluated on an "easy" image segmentation example, with an optically thin cloud patch boxed in green. (a) Visible-spectrum image, (b) LWIR image, (c) SWIR image, (d) Difference between "truth" mask and mask generated by the $C_8$-equivariant dense U-Net trained on visible-spectrum data, excluding pixels within 2 px of a cloud boundary, (e) Difference between "truth" mask and mask generated by the $C_8$-equivariant dense U-Net trained on visible-spectrum data, (f) Mask generated by the $C_8$-equivariant dense U-Net trained on visible-spectrum data, (g) "Truth" mask.

Figure 3-18 evaluates the $C_8$-equivariant U-Net trained on visible-spectrum data only on a "hard" image segmentation example, with a cloud patch over snow boxed in green and an optically thin cloud patch boxed in pink. The model-generated mask shown in Figure 3-17(f) clearly has some extra cloud patches, one of which (on top) appears to really be snow, and one of which (on left) appears to really be land. This is indicative of the $C_8$-equivariant U-Net's difficulty distinguishing between cloud and snow pixels when trained only on visible-spectrum data.

Table 3.17 presents the overall performance of the $C_8$-equivariant U-Net trained on visible-spectrum data only. Notably, this model performs worse than all of the deep learning models presented in §3.1.3-3.1.6, but outperforms the random forest algorithm trained on visible-spectrum, LWIR, and SWIR data in terms of $F_1$ score.

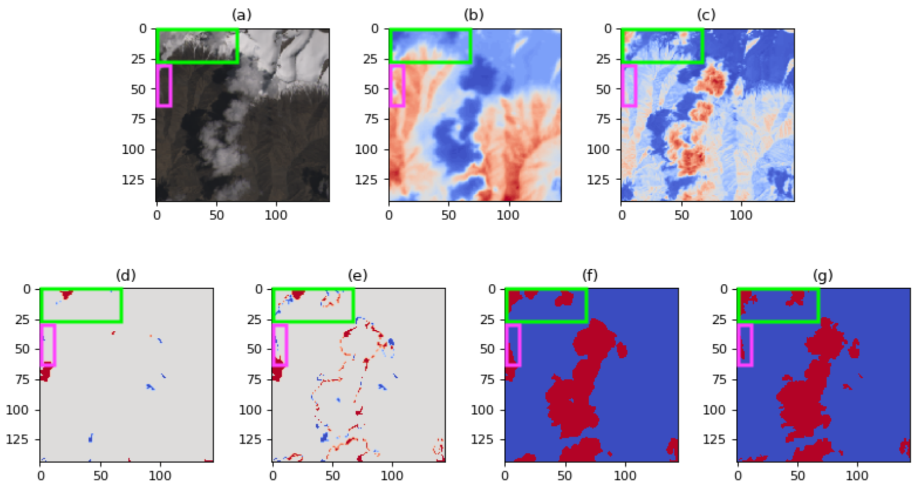


Figure 3-18: $C_8$-equivariant dense U-Net trained on visible-spectrum input only evaluated on a "hard" image segmentation example, with a cloud patch over snow boxed in green and an optically thin cloud patch boxed in pink. (a) Visible-spectrum image, (b) LWIR image, (c) SWIR image, (d) Difference between "truth" mask and mask generated by the $C_8$-equivariant dense U-Net trained on visible-spectrum data, excluding pixels within 2 px of a cloud boundary, (e) Difference between "truth" mask and mask generated by the $C_8$-equivariant dense U-Net trained on visible-spectrum data, (f) Mask generated by the $C_8$-equivariant dense U-Net trained on visible-spectrum data, (g) "Truth" mask.

Table 3.17: Performance metrics detailed in §2.3.1 for the $C_8$-equivariant dense U-Net trained on visible-spectrum data only evaluated on the cloud dataset, evaluated with no buffer and with a 2 px buffer at cloud boundaries.

| **Buffer** | **Acc.** | **Bal. Acc.** | **Sens.** | **Spec.** | **Prec.** | **Recall** | **F$_1$** | **IoU** |
|---|---|---|---|---|---|---|---|---|
| 0 px | 96.53% | 90.34% | 82.29% | 98.38% | 86.86% | 82.29% | 0.8451 | 0.7318 |
| **2 px** | **99.11%** | **96.69%** | **93.54%** | **99.84%** | **98.70%** | **93.54%** | **0.9605** | **0.9240** |

Table 3.18 presents the false positive terrain class ratios for the $C_8$-equivariant U-Net trained on visible-spectrum data only. Ice and snow pixels are clearly over-represented amongst false positives found by this model, and interestingly make up a larger share of the false positives when the cloud edges are excluded using a 2 px buffer. This indicates that this model has real trouble distinguishing clouds from ice and snow, and that many of the ice and snow pixels misidentified as clouds are not simply misidentified at cloud boundaries, but rather represent patches of snow and ice misidentified as clouds despite not being anywhere near clouds in the image.

Table 3.18: Ratios representing the frequency of each terrain class (cloud shadow over land, cloud shadow over water, water, ice/snow, land, and flooded) among pixels falsely identified as clouds by the $C_8$-equivariant dense U-Net trained on visible-spectrum data only to the frequency of each terrain class among all non-cloud pixels in the SPARCS dataset, as explained in §2.3.1, evaluated with no buffer and with a 2 px buffer at cloud boundaries.

| **Buffer** | **Cl. Shad. (Land)** | **Cl. Shad. (Water)** | **Water** | **Ice/Snow** | **Land** | **Flood.** |
|---|---|---|---|---|---|---|
| 0 px | 2.68 | 0.54 | 0.26 | 1.29 | 0.89 | 0.07 |
| 2 px | 1.51 | 0.00 | 0.28 | 3.15 | 0.89 | 0.55 |

### 3.2.2 Visible-Spectrum + LWIR

Figure 3-19 evaluates the $C_8$-equivariant dense U-Net trained on visible-spectrum and LWIR data on an "easy" cloud segmentation example. Like the $C_8$-equivariant dense U-Net trained only on visible-spectrum data, it appears to mostly miss the optically thin cloud in the bottom left of the image (boxed in green) – however, it identifies some pixels in the bottom left of the image as cloud, whereas the model trained on visible-spectrum data misses that cloud patch altogether.

Figure 3-20 evaluates the $C_8$-equivariant dense U-Net trained on visible-spectrum and LWIR data on a "hard" cloud segmentation example with overlapping clouds and snow. Like the $C_8$-equivariant dense U-Net trained only on visible-spectrum data, it identifies two extra cloud patches, one of which (top) appears to be snow, and one of

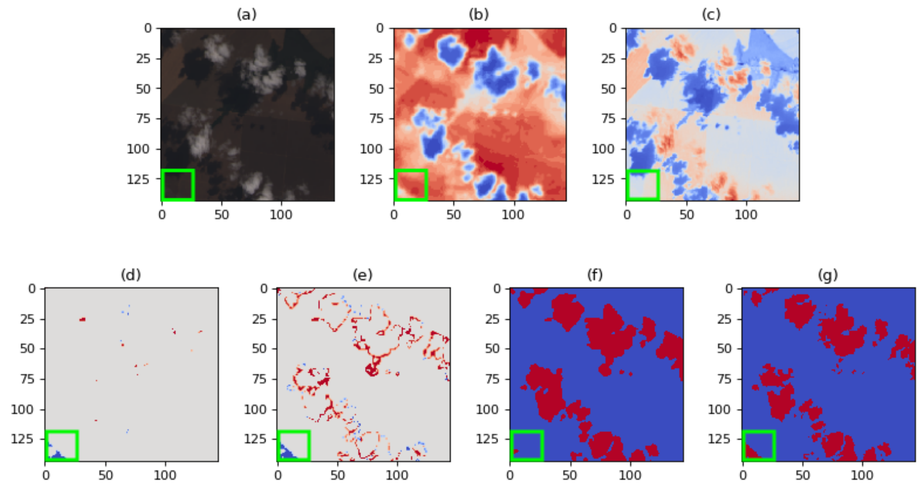


Figure 3-19: $C_8$-equivariant dense U-Net trained on visible-spectrum and LWIR input evaluated on an "easy" image segmentation example, with an optically thin cloud patch boxed in green. (a) Visible-spectrum image, (b) LWIR image, (c) SWIR image, (d) Difference between "truth" mask and mask generated by the $C_8$-equivariant dense U-Net trained on visible-spectrum and LWIR data, excluding pixels within 2 px of a cloud boundary, (e) Difference between "truth" mask and mask generated by the $C_8$-equivariant dense U-Net trained on visible-spectrum and LWIR data, (f) Mask generated by the $C_8$-equivariant dense U-Net trained on visible-spectrum and LWIR data, (g) "Truth" mask.

which (left) appears to be land. However, these false cloud patches are smaller than the false cloud patches found by the model trained only on visible-spectrum data.

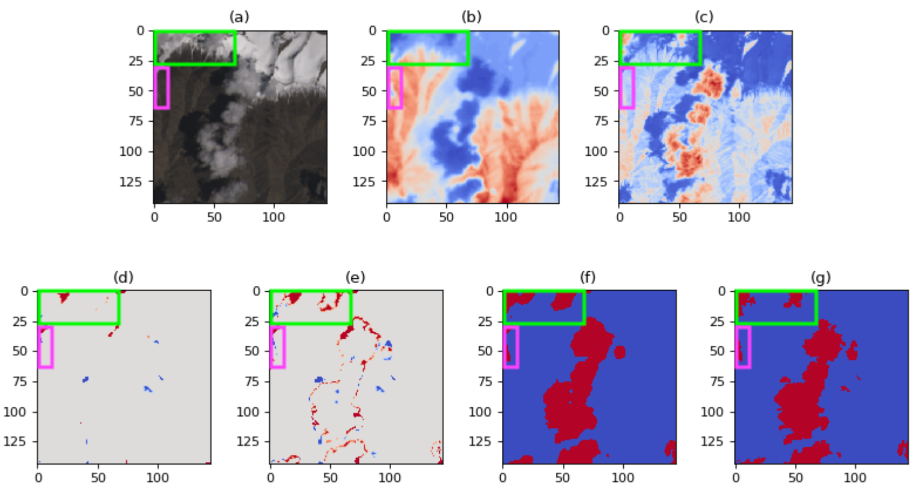


Figure 3-20: $C_8$-equivariant dense U-Net trained on visible-spectrum and LWIR input evaluated on a "hard" image segmentation example, with a cloud patch over snow boxed in green and an optically thin cloud patch boxed in pink. (a) Visible-spectrum image, (b) LWIR image, (c) SWIR image, (d) Difference between "truth" mask and mask generated by the $C_8$-equivariant dense U-Net trained on visible-spectrum and LWIR data, excluding pixels within 2 px of a cloud boundary, (e) Difference between "truth" mask and mask generated by the $C_8$-equivariant dense U-Net trained on visible-spectrum and LWIR data, (f) Mask generated by the $C_8$-equivariant dense U-Net trained on visible-spectrum and LWIR data, (g) "Truth" mask.

Table 3.19 presents the overall performance of the $C_8$-equivariant dense U-Net trained on visible-spectrum and LWIR data. Notably, like the $C_8$-equivariant dense U-Net trained only on visible-spectrum data, this model performs worse than all of the deep learning models presented in §3.1.3-3.1.6, but outperforms the random forest algorithm presented in §3.1.2.

Table 3.19: Performance metrics detailed in §2.3.1 for the $C_8$-equivariant dense U-Net trained on visible-spectrum and LWIR data evaluated on the cloud dataset, evaluated with no buffer and with a 2 px buffer at cloud boundaries.

| **Buffer** | **Acc.** | **Bal. Acc.** | **Sens.** | **Spec.** | **Prec.** | **Recall** | **$F_1$** | **IoU** |
|---|---|---|---|---|---|---|---|---|
| 0 px | 96.76% | 91.61% | 84.91% | 98.30% | 86.65% | 84.91% | 0.8577 | 0.7509 |
| **2 px** | **99.29%** | **97.85%** | **95.98%** | **99.72%** | **97.78%** | **95.98%** | **0.9687** | **0.9393** |

Table 3.20 presents the false positive terrain class ratios for the $C_8$-equivariant dense U-Net trained on visible-spectrum and LWIR data. Ice and snow pixels and water pixels are clearly overrepresented amongst the false positives found by this model. Interestingly, ice and snow pixels are more overrepresented amongst the false positives found by this model than amongst the false positives found by the model trained on only visible-spectrum input. This may be because ice, snow, and cold water pixels look similar to clouds in LWIR data, while bright land pixels look similar to clouds only in visible-spectrum data – so this change can be understood as a relative *improvement* on classes other than ice/snow and water for the model trained on visible-spectrum and LWIR data rather than a performance regression on ice/snow and water classes, because overall, this model outperforms the model trained only on visible-spectrum data.

Table 3.20: Ratios representing the frequency of each terrain class (cloud shadow over land, cloud shadow over water, water, ice/snow, land, and flooded) among pixels falsely identified as clouds by the $C_8$-equivariant dense U-Net trained on visible-spectrum and LWIR data to the frequency of each terrain class among all non-cloud pixels in the SPARCS dataset, as explained in §2.3.1, evaluated with no buffer and with a 2 px buffer at cloud boundaries.

| **Buffer** | **Cl. Shad. (Land)** | **Cl. Shad. (Water)** | **Water** | **Ice/Snow** | **Land** | **Flood.** |
|---|---|---|---|---|---|---|
| 0 px | 2.47 | 0.48 | 0.52 | 2.24 | 0.79 | 0.00 |
| 2 px | 0.95 | 0.00 | 1.93 | 4.47 | 0.56 | 0.00 |

### 3.2.3 Visible-Spectrum + SWIR

Figure 3-21 evaluates the performance of the $C_8$-equivariant dense U-Net trained on visible-spectrum and SWIR data on an "easy" image segmentation example. Unlike the masks generated by the $C_8$-equivariant dense U-Net trained on visible-spectrum data (shown in Figure 3-17(f)) and the $C_8$-equivariant dense U-Net trained on visible-spectrum and LWIR data (shown in Figure 3-19(f)), the model-generated mask shown in Figure 3-21(f) correctly identifies the optically thin cloud patch in the bottom left (boxed in green).

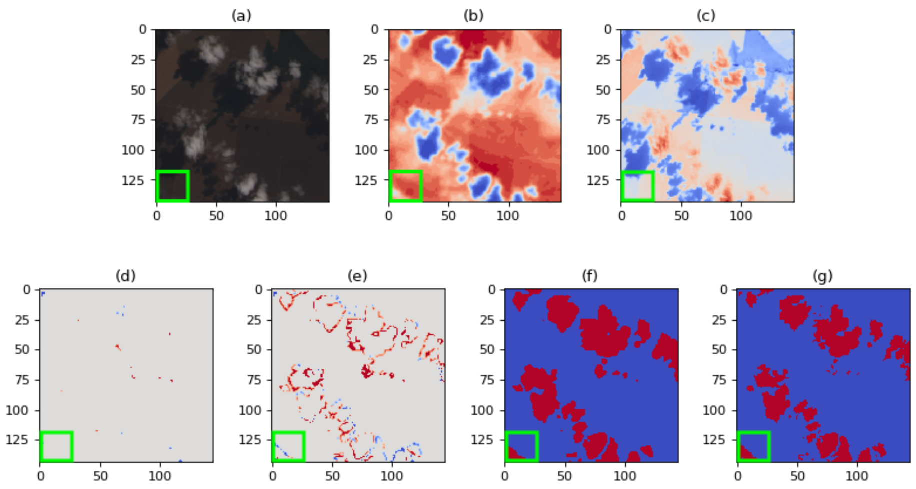


Figure 3-21: $C_8$-equivariant dense U-Net trained on visible-spectrum and SWIR input evaluated on an “easy” image segmentation example, with an optically thin cloud patch boxed in green. (a) Visible-spectrum image, (b) LWIR image, (c) SWIR image, (d) Difference between “truth” mask and mask generated by the $C_8$-equivariant dense U-Net trained on visible-spectrum and SWIR data, excluding pixels within 2 px of a cloud boundary, (e) Difference between “truth” mask and mask generated by the $C_8$-equivariant dense U-Net trained on visible-spectrum and SWIR data, (f) Mask generated by the $C_8$-equivariant dense U-Net trained on visible-spectrum and SWIR data, (g) “Truth” mask.

Figure 3-22 evaluates the performance of the $C_8$-equivariant dense U-Net trained on visible-spectrum and SWIR data on a “hard” image segmentation example with overlapping snow and cloud. Unlike the masks generated by the $C_8$-equivariant dense U-Net trained on visible-spectrum data (shown in Figure 3-18(f)) and the $C_8$-equivariant dense U-Net trained on visible-spectrum and LWIR data (shown in Figure 3-20(f)), the model-generated mask shown in Figure 3-22(f) does not mistake any snow patches for clouds, despite missing a few cloud patches over land, especially in the pink-boxed region containing an optically thin cloud patch.

The performance metrics for the $C_8$-equivariant dense U-Net trained on visible-spectrum and SWIR data are given in Table 3.21. Notably, this model improves on specificity and precision, but does worse on sensitivity and recall, when compared to the $C_8$-equivariant dense U-Net trained on visible-spectrum and LWIR data. This may point to a difficulty in detecting optically thin clouds without LWIR data, or

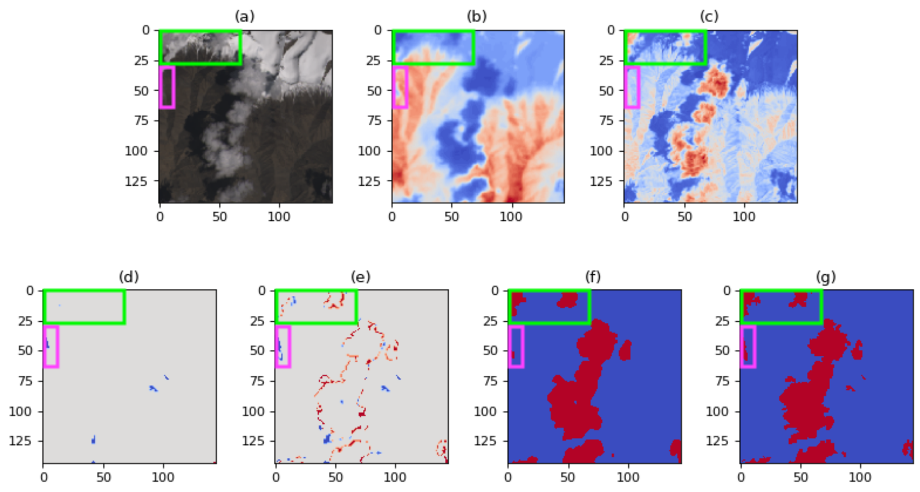


Figure 3-22: $C_8$-equivariant dense U-Net trained on visible-spectrum and SWIR input evaluated on a "hard" image segmentation example, with a cloud patch over snow boxed in green and an optically thin cloud patch boxed in pink. (a) Visible-spectrum image, (b) LWIR image, (c) SWIR image, (d) Difference between "truth" mask and mask generated by the $C_8$-equivariant dense U-Net trained on visible-spectrum and SWIR data, excluding pixels within 2 px of a cloud boundary, (e) Difference between "truth" mask and mask generated by the $C_8$-equivariant dense U-Net trained on visible-spectrum and SWIR data, (f) Mask generated by the $C_8$-equivariant dense U-Net trained on visible-spectrum and SWIR data, (g) "Truth" mask.

data from other bands not used in this thesis.

Table 3.21: Performance metrics detailed in §2.3.1 for the $C_8$-equivariant dense U-Net trained on visible-spectrum and SWIR data evaluated on the cloud dataset, evaluated with no buffer and with a 2 px buffer at cloud boundaries.

| **Buffer** | **Acc.** | **Bal. Acc.** | **Sens.** | **Spec.** | **Prec.** | **Recall** | **$F_1$** | **IoU** |
|---|---|---|---|---|---|---|---|---|
| 0 px | 96.89% | 90.43% | 82.04% | 98.83% | 90.10% | 82.04% | 0.8588 | 0.7526 |
| **2 px** | **99.21%** | **96.67%** | **93.39%** | **99.94%** | **99.50%** | **93.39%** | **0.9635** | **0.9296** |

The false positive terrain class ratios for the $C_8$-equivariant dense U-Net trained on visible-spectrum and SWIR data are given in Table 3.22. This model shows a significant relative improvement on distinguishing clouds from ice, snow, and water when compared to the model trained on visible-spectrum and LWIR data, and does relatively worse on distinguishing clouds from cloud shadow and land. This is likely because ice, snow, and water are moderately absorptive in the SWIR band [49], while ice particles and water droplets in clouds are small enough to scatter radiation in the SWIR band, so clouds look brighter than snow, ice, and water on the ground in the SWIR band, while cold water, ice, and snow look similar to clouds in the LWIR band.

Table 3.22: Ratios representing the frequency of each terrain class (cloud shadow over land, cloud shadow over water, water, ice/snow, land, and flooded) among pixels falsely identified as clouds by the $C_8$-equivariant dense U-Net trained on visible-spectrum and SWIR data to the frequency of each terrain class among all non-cloud pixels in the SPARCS dataset, as explained in §2.3.1, evaluated with no buffer and with a 2 px buffer at cloud boundaries.

| **Buffer** | **Cl. Shad. (Land)** | **Cl. Shad. (Water)** | **Water** | **Ice/Snow** | **Land** | **Flood.** |
|---|---|---|---|---|---|---|
| 0 px | 3.32 | 0.97 | 0.20 | 1.62 | 0.80 | 0.00 |
| 2 px | 3.44 | 0.00 | 0.14 | 0.85 | 1.01 | 0.00 |

### 3.2.4 Visible-Spectrum + LWIR + SWIR

Figure 3-23 reproduces Figure 3-11, demonstrating the performance of the $C_8$-equivariant dense U-Net trained on visible-spectrum, LWIR, and SWIR data on an "easy" image segmentation example. The model-generated mask (shown in Figure 3-23(f)) barely differs from the "truth" mask, except for at the cloud edges, as illustrated by the 2 px buffer "difference" map shown in Figure 3-23(d).

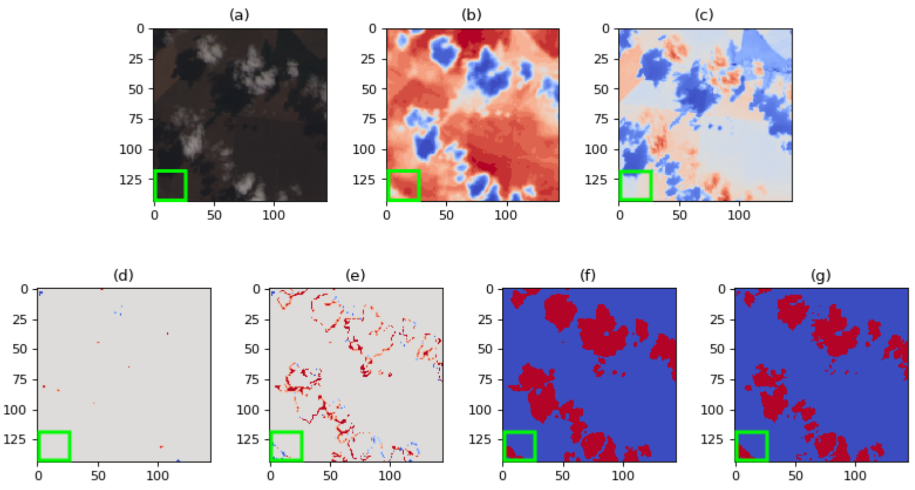


Figure 3-23: $C_8$-equivariant dense U-Net trained on visible-spectrum, LWIR and SWIR input evaluated on an "easy" image segmentation example, with an optically thin cloud patch boxed in green. (a) Visible-spectrum image, (b) LWIR image, (c) SWIR image, (d) Difference between "truth" mask and mask generated by the $C_8$-equivariant dense U-Net trained on visible-spectrum, LWIR and SWIR data, excluding pixels within 2 px of a cloud boundary, (e) Difference between "truth" mask and mask generated by the $C_8$-equivariant dense U-Net trained on visible-spectrum, LWIR and SWIR data, (f) Mask generated by the $C_8$-equivariant dense U-Net trained on visible-spectrum, LWIR and SWIR data, (g) "Truth" mask.

Figure 3-24 reproduces Figure 3-12, demonstrating the performance of the $C_8$-equivariant dense U-Net trained on visible-spectrum, LWIR, and SWIR data on a "hard" image segmentation example. The model-generated mask (shown in Figure 3-24(f)) misses a few cloud pixels, and has a few extra cloud pixels, but these mostly occur at the cloud borders, and no major missing or extra cloud patches are visible in the 2 px buffer "difference" map shown in Figure 3-24(d). The model-generated mask generally correctly discriminates between cloud and snow in the green-boxed region containing overlapping snow and cloud, and correctly identifies the optically thin cloud in the pink-boxed region.

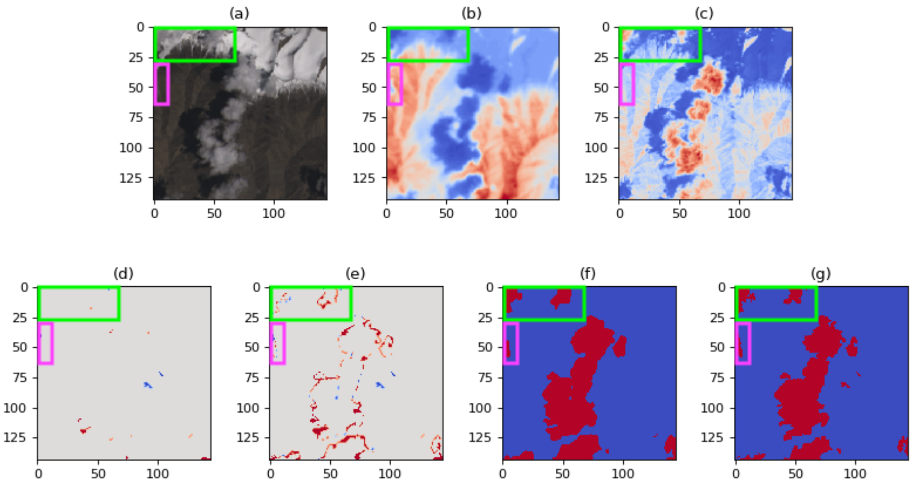


Figure 3-24: $C_8$-equivariant dense U-Net trained on visible-spectrum, LWIR and SWIR input evaluated on a "hard" image segmentation example, with a cloud patch over snow boxed in green and an optically thin cloud patch boxed in pink. (a) Visible-spectrum image, (b) LWIR image, (c) SWIR image, (d) Difference between "truth" mask and mask generated by the $C_8$-equivariant dense U-Net trained on visible-spectrum, LWIR and SWIR data, excluding pixels within 2 px of a cloud boundary, (e) Difference between "truth" mask and mask generated by the $C_8$-equivariant dense U-Net trained on visible-spectrum, LWIR and SWIR data, (f) Mask generated by the $C_8$-equivariant dense U-Net trained on visible-spectrum, LWIR and SWIR data, (g) "Truth" mask.

Table 3.23 reproduces Table 3.11 and shows the performance metrics for the $C_8$-equivariant dense U-Net trained on visible-spectrum, LWIR, and SWIR data. This model outperforms the $C_8$-equivariant dense U-Net trained on visible-spectrum data, the model trained on visible-spectrum and LWIR data, and the model trained on visible-spectrum and SWIR data.

Table 3.23: Performance metrics detailed in §2.3.1 for the $C_8$-equivariant dense U-Net trained on visible-spectrum, LWIR, and SWIR data evaluated on the cloud dataset, evaluated with no buffer and with a 2 px buffer at cloud boundaries.

| **Buffer** | **Acc.** | **Bal. Acc.** | **Sens.** | **Spec.** | **Prec.** | **Recall** | **F$_1$** | **IoU** |
|---|---|---|---|---|---|---|---|---|
| 0 px | 97.01% | 93.16% | 88.17% | 98.16% | 86.15% | 88.17% | 0.8715 | 0.7722 |
| **2 px** | **99.54%** | **98.58%** | **97.32%** | **99.84%** | **98.82%** | **97.32%** | **0.9806** | **0.9620** |

Table 3.24 reproduces Table 3.12 and shows the false positive terrain class ratios for the $C_8$-equivariant dense U-Net trained on visible-spectrum, LWIR, and SWIR data. These ratios fall in between the ratios for the model trained on visible-spectrum and LWIR data and the model trained on visible-spectrum and SWIR data. This is expected because the model trained on visible-spectrum, LWIR, and SWIR data retains the benefits of the SWIR band for distinguishing between clouds and ice, snow, and water and the benefits of the LWIR band for distinguishing between optically thin clouds and land, and as such does not have a relative advantage on either of these types of identification.

Table 3.24: Ratios representing the frequency of each terrain class (cloud shadow over land, cloud shadow over water, water, ice/snow, land, and flooded) among pixels falsely identified as clouds by the $C_8$-equivariant dense U-Net trained on visible-spectrum, LWIR, and SWIR data to the frequency of each terrain class among all non-cloud pixels in the SPARCS dataset, as explained in §2.3.1, evaluated with no buffer and with a 2 px buffer at cloud boundaries.

| **Buffer** | **Cl. Shad. (Land)** | **Cl. Shad. (Water)** | **Water** | **Ice/Snow** | **Land** | **Flood.** |
|---|---|---|---|---|---|---|
| 0 px | 3.23 | 0.76 | 0.16 | 1.60 | 0.82 | 0.00 |
| 2 px | 3.30 | 0.00 | 0.15 | 1.36 | 0.97 | 0.00 |

### 3.2.5 Summary

We evaluate the performance of the $C_8$-equivariant dense U-Net trained on four different combinations of bands, and find that the model trained on visible-spectrum, LWIR, and SWIR data performs the best qualitatively and quantitatively, with negligible additional computational cost over the other models.

#### Segmentation Performance

First, we qualitatively compare the performance of the $C_8$-equivariant dense U-Net trained on four different combinations of bands by evaluating each model on an "easy" image segmentation example with no snow, ice, cold water, or bright land pixels and on a “hard” image segmentation example with overlapping clouds and snow and then visually comparing the resulting model-generated masks to each other and to the “truth” mask. Figure 3-25 evaluates the $C_8$-equivariant dense U-Net trained on four different combinations of Landsat 8 bands on the “easy” image segmentation example with no snow, cold water, or bright non-cloud pixels. All four model-generated masks look very similar to the “truth” mask shown in Figure 3-25(d), except for slight differences identifying the optically thin clouds in the bottom left of the image (boxed in green).

Unexpectedly, the model trained on visible-spectrum and SWIR data (mask shown in Figure 3-25(g)) performs better than the model trained on visible-spectrum and LWIR data (mask shown in Figure 3-25(f)) on classifying the optically thin cloud. This optically thin cloud is not visible in the visible-spectrum or SWIR data, and is significantly dimmer than the rest of the clouds in the image in the LWIR data, so it is likely that the models classifying it correctly are successfully extrapolating from the rest of the image rather than depending on the spectral properties of the optically thin cloud patch. The overall improvement of the model trained on visible-spectrum data only, the model trained on visible-spectrum and LWIR data, and the model trained on visible-spectrum and SWIR data on this sample when compared to the “hard” example shown in Figure 3-26 illustrates the fact that augmenting visible-spectrum data with LWIR and SWIR data helps the most when classifying “tricky” cloud patches that are optically thin or overlap with snow and ice terrain.

Figure 3-26 evaluates the $C_8$-equivariant dense U-Net trained on four different combinations of Landsat 8 bands on a “hard” image segmentation sample with overlapping snow and cloud. The model trained on only visible-spectrum imagery (mask shown in Figure 3-26(e)) and the model trained on visible-spectrum and LWIR imagery (mask shown in Figure 3-26(f)) have difficulty distinguishing snow and cloud in the top left of the image, in the region boxed in green. The model trained on visible-spectrum imagery only also has trouble correctly classifying optically thin cloud on the left of the image, as does the model trained on visible-spectrum and SWIR data (mask shown in Figure 3-26(h)). This region is boxed in pink. The model trained on

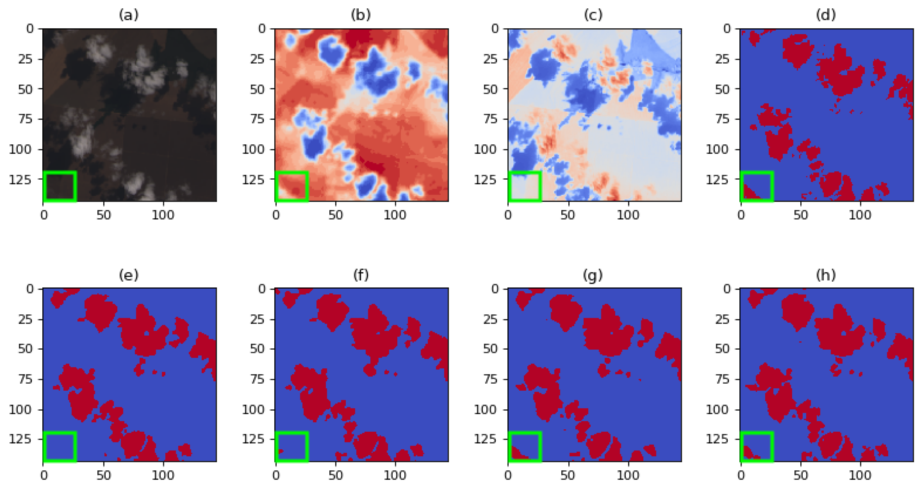


Figure 3-25: $C_8$-equivariant dense U-Net trained on visible-spectrum data only, visible-spectrum and LWIR data, visible-spectrum and SWIR data, and visible-spectrum, LWIR, and SWIR data evaluated on an "easy" image segmentation sample, with an optically thin cloud patch boxed in green. (a) Visible-spectrum input, (b) LWIR input, (c) SWIR input, (d) "Truth" mask, (e) Mask generated by $C_8$-equivariant dense U-Net trained on visible-spectrum data only, (f) Mask generated by $C_8$-equivariant dense U-Net trained on visible-spectrum and LWIR data, (g) Mask generated by $C_8$-equivariant dense U-Net trained on visible-spectrum and SWIR data, (h) Mask generated by $C_8$-equivariant dense U-Net trained on visible-spectrum, LWIR, and SWIR data.

visible-spectrum, LWIR, and SWIR data (mask shown in 3-26(h)) is able to correctly classify clouds in both the green-boxed region and the pink-boxed region. These results qualitatively demonstrate the challenges distinguishing snow and clouds without SWIR data, and the difficulty identifying optically thin clouds without LWIR or cirrus-band data.

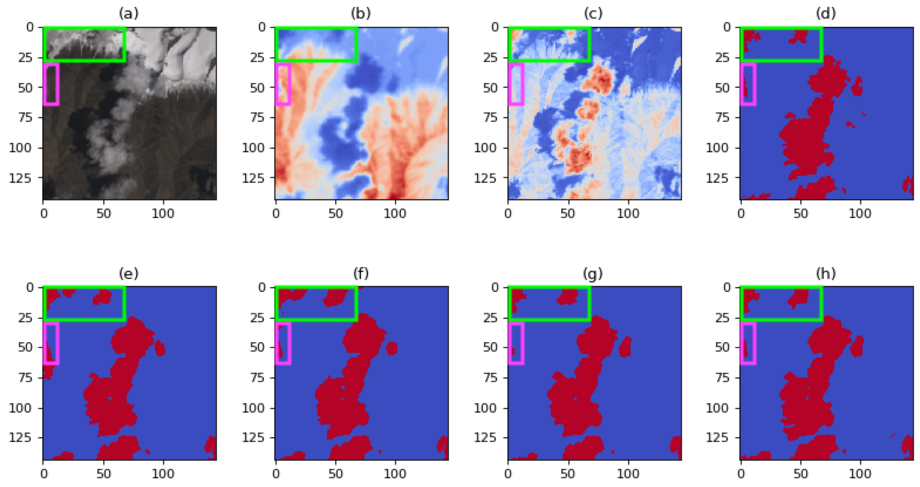


Figure 3-26: $C_8$-equivariant dense U-Net trained on visible-spectrum data only, visible-spectrum and LWIR data, visible-spectrum and SWIR data, and visible-spectrum, LWIR, and SWIR data evaluated on a "hard" image segmentation sample, with a cloud patch over snow boxed in green and an optically thin cloud patch boxed in pink. (a) Visible-spectrum input, (b) LWIR input, (c) SWIR input, (d) "Truth" mask, (e) Mask generated by $C_8$-equivariant dense U-Net trained on visible-spectrum data only, (f) Mask generated by $C_8$-equivariant dense U-Net trained on visible-spectrum and LWIR data, (g) Mask generated by $C_8$-equivariant dense U-Net trained on visible-spectrum and SWIR data, (h) Mask generated by $C_8$-equivariant dense U-Net trained on visible-spectrum, LWIR, and SWIR data.

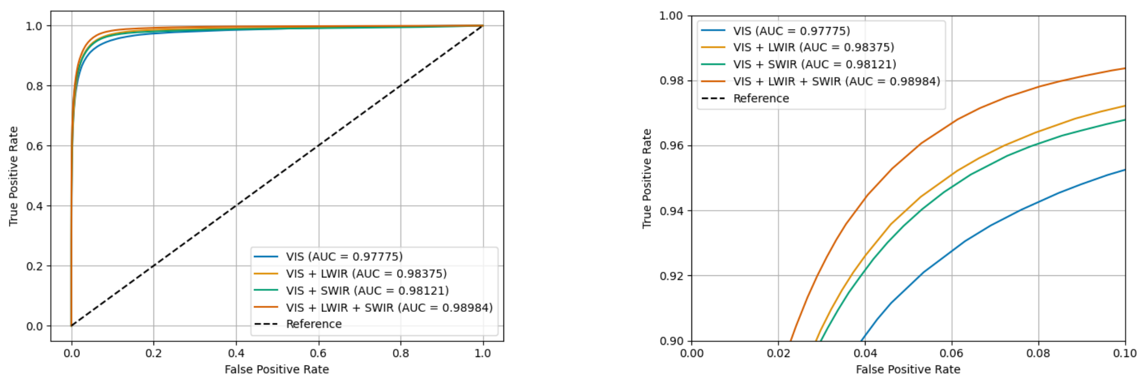
Next, we quantitatively compare the $C_8$-equivariant dense U-Net trained on four different combinations of bands. Table 3.25 summarizes the results of the $C_8$-equivariant dense U-Net trained on four different combinations of bands when evaluated on the SPARCS dataset with no buffer. The model trained on visible-spectrum and SWIR bands has the highest specificity and precision, but the model trained on the visible-spectrum, LWIR, and SWIR bands has the best performance on every other metric. Also, the model trained on visible-spectrum and SWIR bands outperforms the model trained on visible-spectrum and LWIR bands on accuracy and $F_1$ score, but both models are extremely similar in performance.

Table 3.25: Performance metrics detailed in §2.3.1 for the $C_8$-equivariant dense U-Net trained on visible-spectrum data (VIS) only, visible-spectrum and LWIR data, visible-spectrum and SWIR data, and visible-spectrum and LWIR and SWIR data, evaluated on the modified SPARCS dataset without using a buffer at cloud boundaries.

| **Model** | **Acc.** | **Bal. Acc.** | **Sens.** | **Spec.** | **Prec.** | **Recall** | **F$_1$** | **IoU** |
|---|---|---|---|---|---|---|---|---|
| VIS | 96.53% | 90.34% | 82.29% | 98.38% | 86.86% | 82.29% | 0.8451 | 0.7318 |
| VIS + LWIR | 96.76% | 91.61% | 84.91% | 98.30% | 86.65% | 84.91% | 0.8577 | 0.7509 |
| VIS + SWIR | 96.89% | 90.43% | 82.04% | **98.83%** | **90.10%** | 82.04% | 0.8588 | 0.7526 |
| VIS + LWIR + SWIR | **97.01%** | **93.16%** | **88.17%** | 98.16% | 86.15% | **88.17%** | **0.8715** | **0.7722** |

Figure 3-27 plots the receiver-operating characteristic (ROC) curve for the $C_8$-equivariant dense U-Net trained on four different combinations of bands. The ROC curve for the model trained on visible-spectrum, LWIR, and SWIR bands is consistently the closest to the $(0, 1)$ point indicating a perfect classifier, and has both a lower false positive rate than the other three models with the true positive rate held constant and a higher true positive rate than the other three models with the false positive rate held constant. Similarly, the curve representing model trained on visible-spectrum and LWIR data consistently stays closer to $(0, 1)$ than either the curve representing the model trained on visible-spectrum and SWIR data or the curve representing the model trained on visible-spectrum data only. This indicates that the fact that the model trained on visible-spectrum and SWIR data has a slightly better $F_1$ score than the model trained on visible-spectrum and LWIR data (see Table 3.25) is likely specific to the cloud threshold of 0.5 – with some thresholds, the model trained on visible-spectrum and LWIR data would outperform the model trained on visible-spectrum and SWIR data.

Additionally, the curve representing the model trained on visible-spectrum and SWIR data consistently stays closer to the $(0, 1)$ point than the model trained on visible-spectrum data only. Finally, the gap between the curve representing the model trained on visible-spectrum and LWIR data and the curve representing the model trained on visible-spectrum and SWIR data is much smaller than the other performance gaps between models. This demonstrates that adding more bands to the image input improves the performance of the $C_8$-equivariant dense U-Net, and that adding a LWIR band helps more than adding a SWIR band, but the difference is relatively small – this fits with the results presented in Table 3.26.



(a) Original view.

(b) Zoomed-in view.

Figure 3-27: Receiver-operating characteristic (ROC) curve for the $C_8$-equivariant dense U-Net trained on visible-spectrum data (VIS) only, visible-spectrum and LWIR data, visible-spectrum and SWIR data, and visible-spectrum and LWIR and SWIR data, evaluated on the modified SPARCS dataset without using a buffer at cloud boundaries.

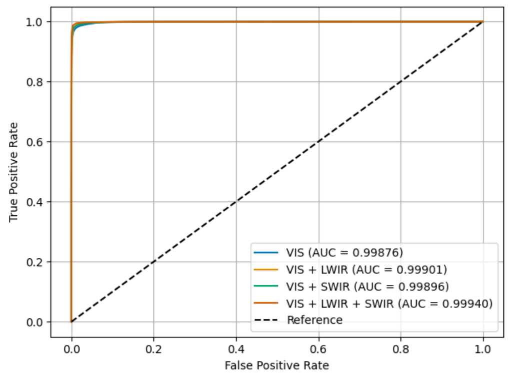
Table 3.26 summarizes the results of the $C_8$-equivariant dense U-Net trained on four different combinations of bands, evaluated with a 2 px buffer at the cloud boundaries. Although the model trained on the visible-spectrum and SWIR bands only has the highest specificity and precision, the model trained on the visible-spectrum, LWIR, and SWIR bands has the best performance on every other metric. Also, the model trained on visible-spectrum and LWIR data has a higher $F_1$ score than the model trained on visible-spectrum and SWIR data when a 2 px buffer at the cloud edges is taken into account. This likely indicates that the model trained on visible-spectrum and LWIR data is better-optimized for a cloud threshold of 0.5 when a 2 px buffer is taken into account at the cloud boundaries rather than when no buffer is taken into account.

Table 3.26: Performance metrics detailed in §2.3.1 for the $C_8$-equivariant dense U-Net trained on visible-spectrum data (VIS) only, visible-spectrum and LWIR data, visible-spectrum and SWIR data, and visible-spectrum and LWIR and SWIR data, evaluated on the modified SPARCS dataset with a 2 px buffer at cloud boundaries.

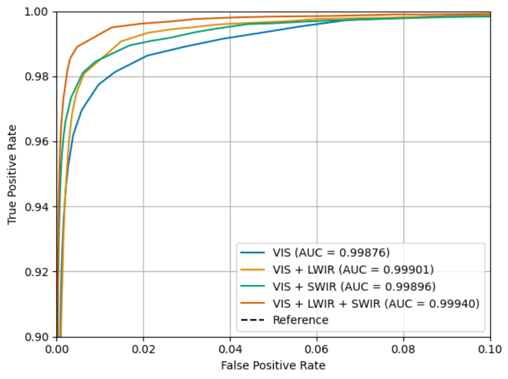
| **Model** | **Acc.** | **Bal. Acc.** | **Sens.** | **Spec.** | **Prec.** | **Recall** | $\mathbf{F_1}$ | **IoU** |
|---|---|---|---|---|---|---|---|---|
| VIS | 99.11% | 96.69% | 93.54% | 99.84% | 98.70% | 93.54% | 0.9605 | 0.9240 |
| VIS + LWIR | 99.29% | 97.85% | 95.98% | 99.72% | 97.78% | 95.98% | 0.9687 | 0.9393 |
| VIS + SWIR | 99.21% | 96.67% | 93.39% | **99.94%** | **99.50%** | 93.39% | 0.9635 | 0.9296 |
| VIS + LWIR + SWIR | **99.54%** | **98.58%** | **97.32%** | 99.84% | 98.82% | **97.32%** | **0.9806** | **0.9620** |

Figure 3-28 plots the receiver-operating characteristic (ROC) curve for the $C_8$-equivariant dense U-Net trained on four different combinations of bands when each model is evaluated with a 2 px buffer at the cloud boundaries. As in Figure 3-27, the curve representing the model trained on visible-spectrum, LWIR, and SWIR data consistently stays the closest to $(0, 1)$, and the curve representing the model trained on visible-spectrum data only is consistently the furthest from $(0, 1)$. However, when a 2 px buffer at the cloud boundaries is introduced, the curves representing the model trained on visible-spectrum and LWIR data and the model trained on visible-spectrum and SWIR data cross, reflecting different sensitivity/specificity tradeoffs between the two models. The curve representing the model trained on visible-spectrum and LWIR data appears to be shifted upwards and to the right when compared to the curve representing the model trained on visible-spectrum and SWIR data. This shift reflects the fact that the model trained on visible-spectrum and LWIR data performs better in the regime where the false positive rate is greater than about 0.01%, but the model trained on visible-spectrum and SWIR data performs better in the regime where the false positive rate is below 0.01%.

This means that the model trained on visible-spectrum and SWIR data is a better choice for applications where high specificity is extremely important, and the model trained on visible-spectrum and LWIR data is a better choice for applications where sensitivity is more important than specificity, if a 2 px buffer is used for evaluation. This fits with the result that the model trained on visible-spectrum and SWIR data has much higher specificity than sensitivity, while the model trained on visible-spectrum and LWIR data better balances sensitivity and specificity, resulting in a higher $F_1$ score when evaluated with a 2 px buffer, as shown in Table 3.26.



(a) Original view.



(b) Zoomed-in view.

Figure 3-28: Receiver-operating characteristic (ROC) curve for the $C_8$-equivariant dense U-Net trained on visible-spectrum data (VIS) only, visible-spectrum and LWIR data, visible-spectrum and SWIR data, and visible-spectrum and LWIR and SWIR data, evaluated on the modified SPARCS dataset with a 2 px buffer at cloud boundaries.

### Resource Utilization

Table 3.27 presents the resource utilization of the $C_8$-equivariant dense U-Net when trained on different combinations of bands. The peak memory utilization and inference time is very similar for all four models, because all models share the same underlying architecture and differ only in the number of input channels used in the first convolution layer. This demonstrates that the marginal computational cost of adding an additional band to an input into the $C_8$-equivariant dense U-Net is very low.

Table 3.28 presents the model complexity of the $C_8$-equivariant dense U-Net when trained on different combinations of bands. The model size is the same for all four

Table 3.27: Peak memory usage over 100 single-image classifications and average inference time over 1000 single-image classifications using both a CPU (Intel Xeon) and GPU (Nvidia K80) backend for the $C_8$-equivariant dense U-Net trained on visible-spectrum data only, on visible-spectrum and LWIR data, on visible-spectrum and SWIR data, and on visible-spectrum, LWIR, and SWIR data on the modified SPARCS dataset.

| **Model** | **GPU Mem. (MiB)** | **CPU Mem. (KiB)** | **GPU Inf. Time (s)** | **CPU Inf. Time (s)** |
|---|---|---|---|---|
| VIS | 503.7 | 129.0 | 0.0496 | 0.2281 |
| VIS + LWIR | 503.3 | 164.9 | 0.0522 | 0.2278 |
| VIS + SWIR | 503.3 | 129.5 | 0.0498 | 0.2278 |
| VIS + LWIR + SWIR | 503.5 | 144.8 | 0.0484 | 0.2281 |

models, and the number of parameters is very similar for all four models – adding a band to the input image results in a gain of only six trainable parameters out of hundreds of thousands. This again highlights the fact that the marginal computational cost of adding an additional band to an input into the $C_8$-equivariant dense U-Net is very low.

Table 3.28: Saved model size, total number of parameters, and number of trainable parameters for the $C_8$-equivariant dense U-Net trained on visible-spectrum data only, on visible-spectrum and LWIR data, on visible-spectrum and SWIR data, and on visible-spectrum, LWIR, and SWIR data, trained on the modified SPARCS dataset.

| **Model** | **Model Size** | **Total Parameters** | **Trainable Parameters** |
|---|---|---|---|
| VIS | 14.6 MB | 293775 | 290271 |
| VIS + LWIR | 14.6 MB | 293783 | 290277 |
| VIS + SWIR | 14.6 MB | 293783 | 290277 |
| VIS + LWIR + SWIR | 14.6 MB | 293791 | 290283 |

THIS PAGE INTENTIONALLY LEFT BLANK

# Chapter 4

# Road Segmentation Results

## 4.1 Evaluating Model Performance

We analyze the performance of the four deep learning algorithms described in §2.2.3-2.2.6 on the Massachusetts Roads Dataset, which can be downloaded from Kaggle at `https://www.kaggle.com/datasets/insaff/massachusetts-roads-dataset`. We qualitatively evaluate each model by visually comparing model-generated masks to "truth" masks and model input for two different example images. The first example image, which we refer to as the "easy" image sample, comes from the image labeled as `''img-8''` in the Kaggle-hosted version of the Massachusetts Roads Dataset test set. The second example image, which we refer to as the "hard" image sample, comes from the image labeled as `''img-4''` in the Kaggle-hosted version of the Massachusetts Roads Dataset test set. We quantitatively evaluate each model by calculating the metrics described in §2.3.1. Finally, we measure the number of parameters, saved model size, peak memory usage, and inference time of each algorithm in order to evaluate model complexity and resource usage.

In addition to the four deep learning algorithms evaluated in this chapter, we evaluated the luminosity thresholding algorithm described in §2.2.1 and the random forest algorithm described in §2.2.2 on the Massachusetts Roads Dataset. In both cases, the algorithm converged to a point with perfect specificity but zero sensitivity – that is, the algorithm labeled all pixels as non-road. This indicates that visible-spectrum pixel luminosity in a $3 \times 3$ window is insufficient for distinguishing road and non-road pixels; either more spatial context or more spectral information is necessary for road segmentation.

Finally, recall that in our road segmentation experiments, our U-Net used 16 input channels in the first convolution block, and our $C_8$-equivariant U-Net used 2 channels per orientation in the first convolution block. In contrast, the U-Net used in our cloud segmentation experiments and the dense U-Net used in both sets of experiments used 64 input channels in the first convolution block, and the $C_8$-equivariant U-Net used in our cloud segmentation experiments and the $C_8$-equivariant dense U-Net used in both sets of experiments used 8 channels per orientation in the first convolution block (see §2.4.1). We restricted the number of input channels in the road segmentation U-Net and the road segmentation $C_8$-equivariant U-Net to 16 channels and 2 channels per orientation, respectively, in order to fit the models on the GPU used for training.

### 4.1.1 U-Net

Figure 4-1 shows the application of the U-Net to an "easy" image sample with only one small parking lot segment (boxed in green), a gridlike road network, and very few occluded road segments (boxed in white and light blue). The road labels in the "truth" mask (shown in Figure 4-1(b)) are narrower than the roads in the U-Net-generated mask (shown in Figure 4-1(c)), and as a result, the U-Net-generated mask has a significant number of false positives that fall within 4 px of a road boundary, as seen in the difference map in Figure 4-1(d). The U-Net-generated mask also includes a few extra segments, and labels some segments as longer than those in the "truth" mask – these segments are apparent in the difference map generated using a 4 px buffer at road boundaries, shown in Figure 4-1(e).

Figure 4-2 shows the application of the U-Net to a "hard" image containing a parking lot (boxed in green), narrow and partially occluded roads (boxed in white and light blue), and curved roads in a variety of orientations. Roads pictured in the Massachusetts Roads Dataset vary in width, but most of the roads in the dataset are comparable in width to the road segments in the "easy" image example (see Figure 4-

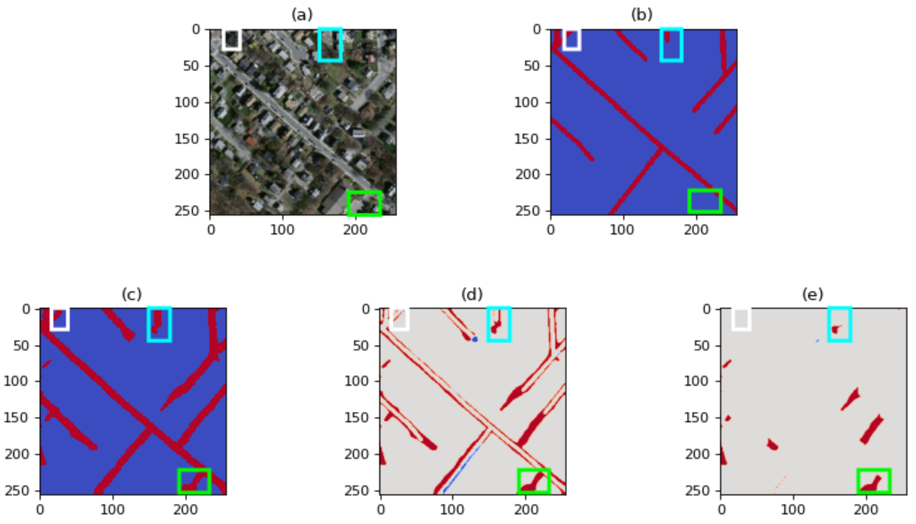


Figure 4-1: U-Net evaluated on an “easy” road segmentation example, with partially occluded road segments boxed in white and light blue, and with a parking lot segment boxed in green. (a) Visible-spectrum image, (b) “Truth” mask, (c) U-Net-generated mask, (d) Difference between “truth” mask and U-Net-generated mask, (e) Difference between “truth” mask and U-Net-generated mask, excluding pixels within 4 px of a road boundary.

1), and the narrow roads in this image are fairly atypical for the dataset. The relative lack of narrow roads in the Massachusetts Roads Dataset likely makes it more difficult for deep learning models to correctly identify narrow road segments.

As in Figure 4-1, the road segments in the U-Net generated mask (Figure 4-2(c)) are wider than those in the “truth” mask (Figure 4-2(b)). Interestingly, this trend holds even for the narrower roads in this image sample – although the U-Net misses most of the narrow roads entirely, as seen in the 4 px difference map (Figure 4-2(e)), the ones it does label are wider in the U-Net generated mask than in the “truth” mask or in the original input image. The U-Net also struggles to label the parking lot in the bottom left of the image, labeling several parts of the parking lot (which looks like a road, except for its shape) as disconnected roads. The “truth” mask, in contrast, does not label any part of the parking lot as a road.

Performance metrics for the U-Net are given in Table 4.1. Most notably, the U-Net has significantly higher recall than precision when evaluated with or without a 4 px buffer at the road boundaries, indicating a high number of false positives.

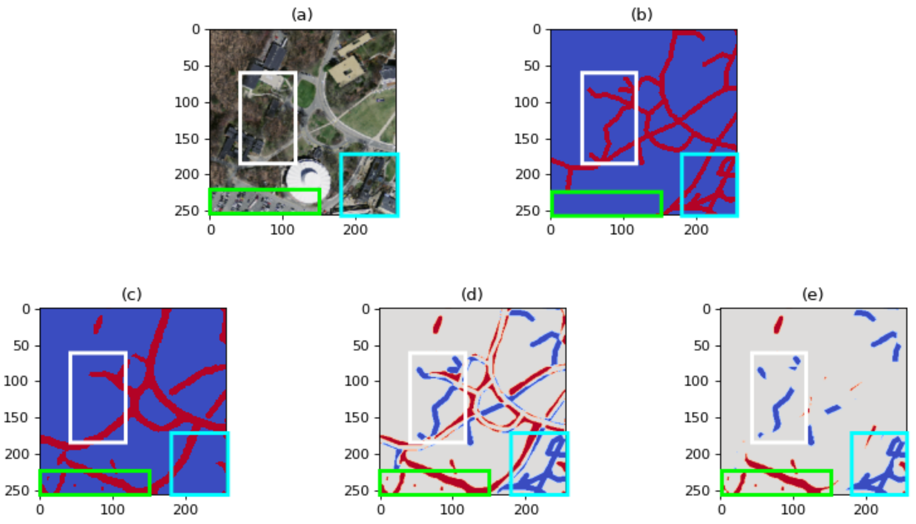


Figure 4-2: U-Net evaluated on a "hard" road segmentation example, with narrow, partially occluded road segments boxed in white and light blue, and with a parking lot boxed in green. (a) Visible-spectrum image, (b) "Truth" mask, (c) U-Net-generated mask, (d) Difference between "truth" mask and U-Net-generated mask, (e) Difference between "truth" mask and U-Net-generated mask, excluding pixels within 4 px of a road boundary.

This fits with the qualitative results shown in Figures 4-1 and 4-2– when no buffer is considered, the U-Net has a high number of false positives related to the width of its labeled road segments, and even when a 4 px buffer is considered, the U-Net falsely identifies road segments as longer than their true length (see Figure 4-1) and falsely identifies parking lots as roads (see Figure 4-2).

Table 4.1: Performance metrics detailed in §2.3.1 for the U-Net evaluated on the Massachusetts Roads Dataset with no buffer and with a 4 px buffer at road boundaries.

| **Buffer** | **Acc.** | **Bal. Acc.** | **Sens.** | **Spec.** | **Prec.** | **Recall** | $\mathbf{F_1}$ | **IoU** |
|---|---|---|---|---|---|---|---|---|
| 0 px | 80.22% | 78.71% | 77.02% | 80.39% | 17.83% | 77.02% | 0.2895 | 0.1693 |
| **4 px** | **96.22%** | **96.46%** | **96.85%** | **96.06%** | **86.08%** | **96.85%** | **0.9115** | **0.8374** |

### 4.1.2 Dense U-Net

Figure 4-3 evaluates the dense U-Net on an "easy" image sample with only one small parking lot segment, a gridlike road network, and few partially occluded road segments. Like the U-Net (see Figure 4-1), the mask generated by the dense U-Net (shown in Figure 4-3(c)) includes wider road segments than those in the "truth" mask (shown in Figure 4-3(b)). As a result, the mask generated by the dense U-Net includes a large number of false positives within 4 px of the road edges, as shown in the difference map in Figure 4-3(d). The mask generated by the dense U-Net also includes some road segments that are elongated from the true segments shown in the "truth" mask. The length differences between segments are most apparent in the difference map with a 4 px buffer (see Figure 4-3(e). Notably, the dense U-Net has fewer elongated segments than the U-Net does (see Figure 4-1).

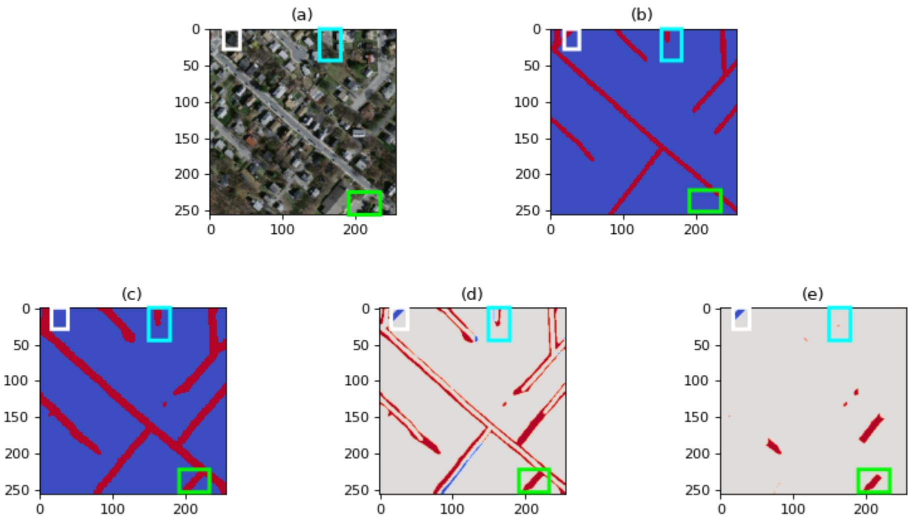


Figure 4-3: Dense U-Net evaluated on an "easy" road segmentation example, with partially occluded road segments boxed in white and light blue, and with a parking lot segment boxed in green. (a) Visible-spectrum image, (b) "Truth" mask, (c) Mask generated by dense U-Net, (d) Difference between "truth" mask and mask generated by dense U-Net, (e) Difference between "truth" mask and mask generated by dense U-Net, excluding pixels within 4 px of a road boundary.

Figure 4-4 evaluates the dense U-Net on a "hard" image sample with a parking lot, narrow and partially occluded road segments, and curved roads. Like the U-Net (see Figure 4-2), the dense U-Net misses nearly all of the narrow road segments, as seen in the mask generated by the dense U-Net (Figure 4-4(c)) and in the 4 px buffer difference map (Figure 4-4(e)). The dense U-Net also misclassifies certain parts of the parking lot in the bottom left of the image (boxed in green in Figure 4-4) as road segments. The misclassified segments appear to match up with the edge of the parking lot and some of the gaps between rows of parked cars, as seen in the image input (Figure 4-4(a)), indicating that the dense U-Net struggles to distinguish between roads and areas of parking lots where driving is possible.

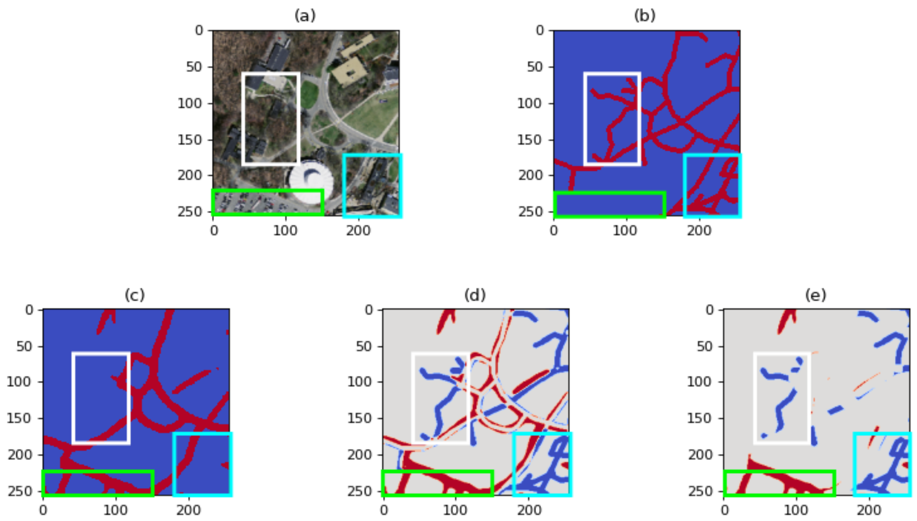


Figure 4-4: Dense U-Net evaluated on a “hard” road segmentation example, with narrow, partially occluded road segments boxed in white and light blue, and with a parking lot boxed in green. (a) Visible-spectrum image, (b) “Truth” mask, (c) Mask generated by dense U-Net, (d) Difference between “truth” mask and mask generated by dense U-Net, (e) Difference between “truth” mask and mask generated by dense U-Net, excluding pixels within 4 px of a road boundary.

The performance metrics for the dense U-Net are given in Table 4.2. When no buffer is used during evaluation, the U-Net outperforms the dense U-Net in terms of $F_1$ score, likely because the dense U-Net has more false positives close to the road boundaries, as evidenced by the poor precision of the dense U-Net when no buffer is used for evaluation. However, when a 4 px buffer is used to evaluate model performance, the dense U-Net outperforms the U-Net in terms of $F_1$ score and shows higher precision than the U-Net. This fits with the qualitative result that the dense U-Net is better at capturing the correct length of road segments than the U-Net, and so has fewer false positives than the U-Net when a 4 px buffer is used for evaluation.

Table 4.2: Performance metrics detailed in §2.3.1 for the dense U-Net evaluated on the Massachusetts Roads Dataset with no buffer and with a 4 px buffer at road boundaries.

| **Buffer** | **Acc.** | **Bal. Acc.** | **Sens.** | **Spec.** | **Prec.** | **Recall** | **F$_1$** | **IoU** |
|---|---|---|---|---|---|---|---|---|
| 0 px | 79.75% | 78.77% | 77.67% | 79.86% | 17.47% | 77.67% | 0.2853 | 0.1664 |
| **4 px** | **96.19%** | **96.45%** | **96.90%** | **96.01%** | **86.29%** | **96.90%** | **0.9129** | **0.8397** |

### 4.1.3 $C_8$-Equivariant U-Net

Figure 4-5 shows the performance of the $C_8$-equivariant U-Net on an "easy" image input with only one small parking lot segment, a gridlike road network, and very few partially occluded roads. Like the prior two models presented in §4.1.1-4.1.2, the $C_8$-equivariant U-Net generates a mask (see Figure 4-5(c)) with wider road segments than the road segments in the "truth" mask shown in Figure 4-5(b). The $C_8$-equivariant U-Net-generated mask includes some road segments which are longer than in the "truth" mask, and includes more of these segments than the dense U-Net-generated mask (see Figure 4-3) but fewer of these segments than the U-Net-generated mask (see Figure 4-1). Also, the mask generated by the $C_8$-equivariant U-Net looks relatively "noisy" – that is, the road segments in the mask are generally not straight and many of the road segments in the mask have small protrusions.

Figure 4-6 shows the performance of the $C_8$-equivariant U-Net on a "hard" image input with a parking lot, narrow and partially occluded road segments, and many curved road segments. The mask generated by the $C_8$-equivariant U-Net (shown in Figure 4-6) again looks relatively "noisy", in that it includes small road patches adjacent to or attached to road segments and that many of the road segments in the mask waver in width. Also, like the two models presented in §4.1.1-4.1.2, the $C_8$-equivariant U-Net generates a mask that misses most of the narrow and partially occluded roads, but misclassifies part of the parking lot as a road. Like the U-Net (see Figure 4-2), the $C_8$-equivariant U-Net generates a mask where disparate parts of the parking lot are identified as road segments, in contrast to the dense U-Net, which labels the parking lot edge and some of the gaps between rows of parked cars

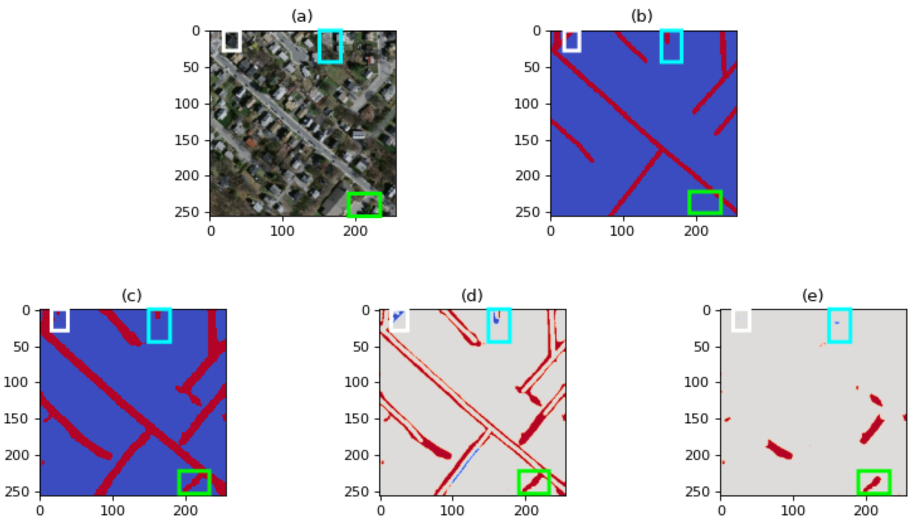


Figure 4-5: $C_8$-equivariant U-Net evaluated on an "easy" road segmentation example, with partially occluded road segments boxed in white and light blue, and with a parking lot segment boxed in green. (a) Visible-spectrum image, (b) "Truth" mask, (c) Mask generated by $C_8$-equivariant U-Net, (d) Difference between "truth" mask and mask generated by $C_8$-equivariant U-Net, (e) Difference between "truth" mask and mask generated by $C_8$-equivariant U-Net, excluding pixels within 4 px of a road boundary.

as smooth road segments (see Figure 4-4).

The performance metrics for the $C_8$-equivariant U-Net are given in Table 4.3. Interestingly, the $C_8$-equivariant U-Net outperforms the dense U-Net when no buffer is used during evaluation, but performs worse than the dense U-Net when a 4 px buffer is used for evaluation. This likely relates to the fact that the $C_8$-equivariant U-Net has higher precision but lower recall than the dense U-Net, indicating that it has fewer false positives and more false negatives than the dense U-Net – an advantage that becomes much less meaningful when a 4 px buffer is taken into account, eliminating many of the false positives found by the dense U-Net. The $C_8$-equivariant U-Net has a slightly lower $F_1$ score than the U-Net regardless of whether or not a 4 px buffer is used for evaluation, likely because the first group convolution done by the $C_8$-equivariant U-Net has only two channels per orientation, and compressing the input to two channels results in loss of information and limits the model's performance.

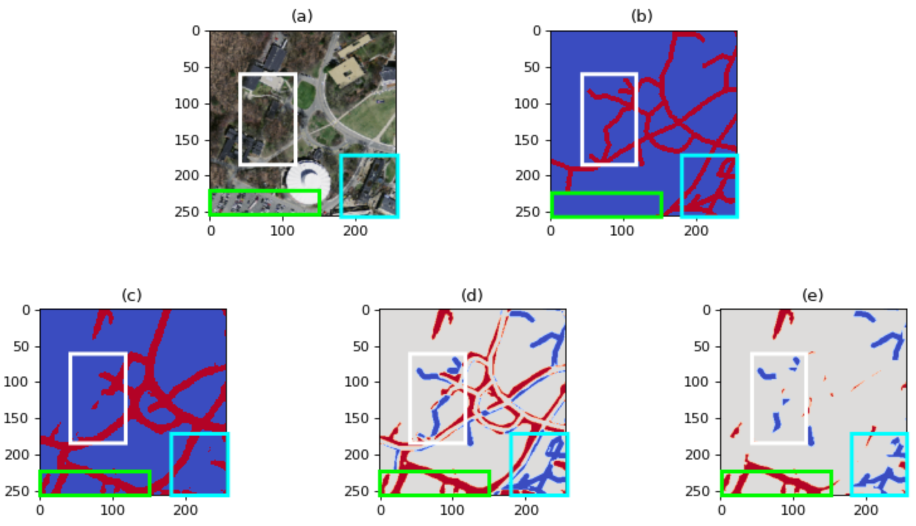


Figure 4-6: $C_8$-equivariant U-Net evaluated on a "hard" road segmentation example, with narrow, partially occluded road segments boxed in white and light blue, and with a parking lot boxed in green. (a) Visible-spectrum image, (b) "Truth" mask, (c) Mask generated by $C_8$-equivariant U-Net, (d) Difference between "truth" mask and mask generated by $C_8$-equivariant U-Net, (e) Difference between "truth" mask and mask generated by $C_8$-equivariant U-Net, excluding pixels within 4 px of a road boundary.

Table 4.3: Performance metrics detailed in §2.3.1 for the $C_8$-equivariant U-Net evaluated on the Massachusetts Roads Dataset with no buffer and with a 4 px buffer at road boundaries.

| **Buffer** | **Acc.** | **Bal. Acc.** | **Sens.** | **Spec.** | **Prec.** | **Recall** | **$F_1$** | **IoU** |
|---|---|---|---|---|---|---|---|---|
| 0 px | 80.51% | 77.74% | 74.63% | 80.84% | 17.93% | 74.63% | 0.2891 | 0.1690 |
| **4 px** | **96.29%** | **96.35%** | **96.46%** | **96.25%** | **86.38%** | **96.46%** | **0.9114** | **0.8372** |

### 4.1.4 $C_8$-Equivariant Dense U-Net

Figure 4-7 demonstrates the performance of the $C_8$-equivariant dense U-Net on an "easy" road segmentation example with gridlike roads, only one small parking lot segment, and very few partially occluded road segments. Qualitatively, the mask generated by the $C_8$-equivariant dense U-Net (shown in Figure 4-7(c)) looks very "smooth"; the road segments in the mask are mostly straight and intersect at right angles where appropriate. However, the mask generated by the $C_8$-equivariant dense U-Net, like the masks generated by the models presented in §4.1.1-4.1.3, includes some elongated road segments. Interestingly, the mask generated by the $C_8$-equivariant dense U-Net has more false positives relating to elongated road segments than the mask generated by the dense U-Net (see Figure 4-3), but this is because the "smooth" mask generated by the $C_8$-equivariant dense U-Net has no gaps in the elongated segments. Finally, the mask generated by the $C_8$-equivariant dense U-Net misses two partially occluded road segments (boxed in white and light blue in Figure 4-7), while other models only miss at most one – this highlights that the $C_8$-equivariant U-Net generally has more false negatives than other models.

Figure 4-8 demonstrates the performance of the $C_8$-equivariant dense U-Net on a "hard" road segmentation example with a parking lot, narrow and partially occluded roads, and curved roads. The mask generated by the $C_8$-equivariant dense U-Net (shown in Figure 4-8(c)) qualitatively looks very smooth, but misses a lot of the detail of the narrow and partially occluded road segments. As shown in the 4 px buffer difference map (Figure 4-8(e)), the $C_8$-equivariant dense U-Net also identifies two major false positive road segments, in the bottom left of the image and in the top center-left of the image. The false segment in the bottom left corresponds to the boundary of the parking lot (boxed in green in Figure 4-8), and looks qualitatively smoother than the false parking lot segments generated by the other three models. The false segment in the top center-left corresponds to a road segment not labeled in the "truth" mask, likely because it is a private road or part of a parking lot. The presence of these false positive segments demonstrates that the $C_8$-equivariant

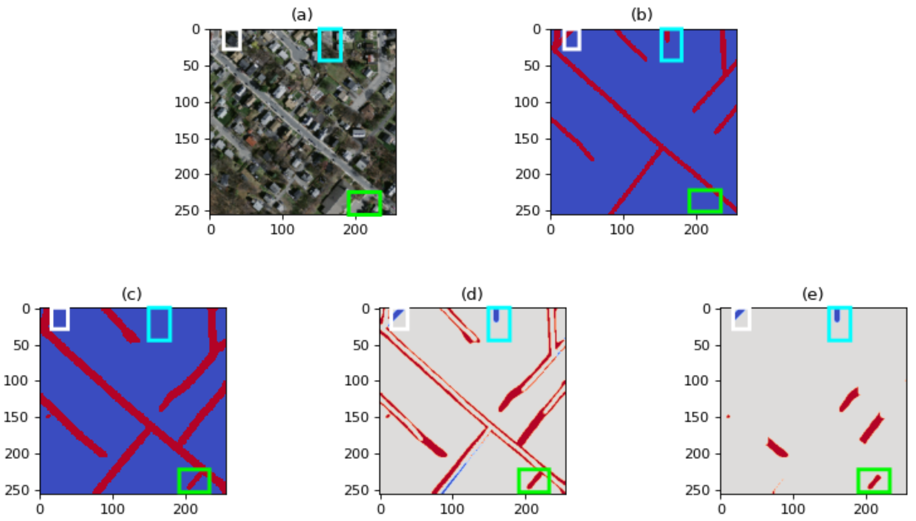


Figure 4-7: $C_8$-equivariant dense U-Net evaluated on an "easy" road segmentation example, with partially occluded road segments boxed in white and light blue, and with a parking lot segment boxed in green. (a) Visible-spectrum image, (b) "Truth" mask, (c) Mask generated by $C_8$-equivariant dense U-Net, (d) Difference between "truth" mask and mask generated by $C_8$-equivariant dense U-Net, (e) Difference between "truth" mask and mask generated by $C_8$-equivariant dense U-Net, excluding pixels within 4 px of a road boundary.

dense U-Net does sometimes generate false positives, but these false positives often correspond to areas where vehicular travel is possible.

Table 4.4 presents the performance metrics of the $C_8$-equivariant dense U-Net evaluated both with and without a 4 px buffer at the road boundaries. In both cases, the $C_8$-equivariant dense U-Net outperforms the three models presented in §4.1.1-4.1.3 in terms of $F_1$ score and precision, but has a lower recall than the three previously presented models. This fits with the result that masks generated by the $C_8$-equivariant dense U-Net look qualitatively "smoother" than the masks generated by other models, and typically have more false negatives but fewer false positives than other models, as seen in Figures 4-7 and 4-8.

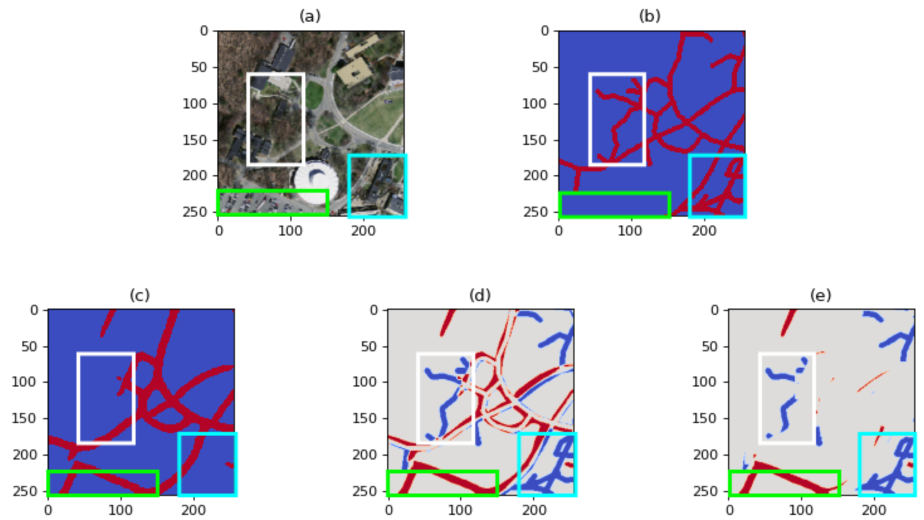


Figure 4-8: $C_8$-equivariant dense U-Net evaluated on a "hard" road segmentation example, with narrow, partially occluded road segments boxed in white and light blue, and with a parking lot boxed in green. (a) Visible-spectrum image, (b) "Truth" mask, (c) Mask generated by $C_8$-equivariant dense U-Net, (d) Difference between "truth" mask and mask generated by $C_8$-equivariant dense U-Net, (e) Difference between "truth" mask and mask generated by $C_8$-equivariant dense U-Net, excluding pixels within 4 px of a road boundary.

Table 4.4: Performance metrics detailed in §2.3.1 for the $C_8$-equivariant dense U-Net evaluated on the Massachusetts Roads Dataset with no buffer and with a 4 px buffer at road boundaries.

| **Buffer** | **Acc.** | **Bal. Acc.** | **Sens.** | **Spec.** | **Prec.** | **Recall** | **F$_1$** | **IoU** |
|---|---|---|---|---|---|---|---|---|
| 0 px | 83.52% | 76.85% | 69.35% | 84.35% | 20.49% | 69.35% | 0.3163 | 0.1878 |
| **4 px** | **97.58%** | **96.33%** | **94.37%** | **98.29%** | **92.49%** | **94.37%** | **0.9342** | **0.8765** |

### 4.1.5 Summary

#### Segmentation Performance

We first qualitatively evaluate the performance of each of the models presented in §4.1.1-4.1.4 by applying each model to an "easy" road segmentation example where roads are gridlike, there is only one small parking lot segment, and there are only two partially occluded road segments. We visually compare the masks generated by each model, which are shown in Figure 4-9. All four models (model-generated masks shown in Figure 4-9(c-f)) generate masks that mostly match the "truth" mask shown in Figure 4-9(b), except for the fact that all four model-generated masks contain an extra road segment not present in the "truth" mask, which is boxed in green. Careful inspection of the input image shows that the green-boxed region does actually contain a road segment, but this segment is missing from the "truth" mask because it appears to be part of a private parking area.

The mask generated by the $C_8$-equivariant dense U-Net (shown in Figure 4-9(f)) looks to have the qualitatively "smoothest" road segments of the model-generated masks; most segments in the mask generated by the $C_8$-equivariant dense U-Net are straight, continuous, and intersect at appropriate angles. However, the $C_8$-equivariant dense U-Net misses both partially occluded road segments, which are boxed in white and light blue. This illustrates the fact that the $C_8$-equivariant dense U-Net has the highest specificity and lowest sensitivity of the four methods – the $C_8$-equivariant dense U-Net can provide smooth road masks without the discontinuities and noisy-looking artifacts seen in the masks generated by the other models, but this comes at the cost of missing some partially occluded road segments. Missing segments can affect route planning, but so can discontinuous segments like those seen in the masks generated by the models other than the $C_8$-equivariant dense U-Net [46].

The mask generated by the U-Net (shown in Figure 4-9(c)) is the only model-generated mask to identify both partially occluded road segments (boxed in white and light blue), but it also falsely identifies some non-road pixels at the left edge of the image as roads, highlighting its high recall but low precision. As shown in Table

4.6, when a 4 px buffer is taken into account when evaluating image masks, the U-Net has the lowest precision of all four models.

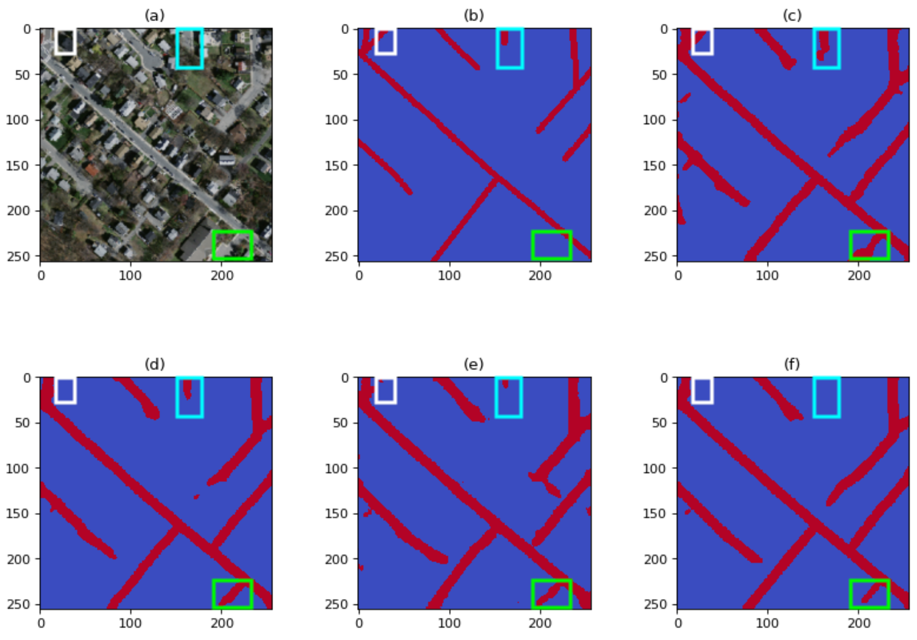


Figure 4-9: U-Net, dense U-Net, $C_8$-equivariant U-Net, and $C_8$-equivariant dense U-Net evaluated on an "easy" image segmentation sample, with partially occluded road segments boxed in white and light blue, and with a parking lot segment boxed in green. (a) Visible-spectrum input image, (b) "Truth" mask, (c) Mask generated by the U-Net, (d) Mask generated by the dense U-Net, (e) Mask generated by the $C_8$-equivariant U-Net, (f) Mask generated by the $C_8$-equivariant dense U-Net.

Next, we qualitatively evaluate the performance of each model presented in §4.1.1-4.1.4 by applying each model to a "hard" road segmentation example with a parking lot, several visually occluded road segments, and twisty roads in a variety of orientations, and visually compare the resulting masks, as shown in Figure 4-10. All four model-generated masks, shown in Figure 4-10(c-f), look qualitatively fairly different from the "truth" mask shown in Figure 4-10(b). First off, all model-generated masks contain thicker roads than the "truth" mask, likely because the road labels in the "truth" masks in the Massachusetts Roads Dataset are generated from rasterized road centerlines and are thus uniform in width, while road labels in model-generated masks tend to match the road width in the input image [38] [5]. This width difference between model-generated and "truth" masks also appears in Figure 4-9.

Second, all four model-generated masks miss the narrow, partially occluded road segments in the center of the image (boxed in white) and in the bottom right of the image (boxed in light blue), but label an additional road segment in the top center-left of the image just above the white-boxed region. Careful inspection of the input image shows that the road segments in the "truth" mask in the white-boxed and blue-boxed regions appear to be real roads, but some are less than 1 pixel in width, and these roads appear to mostly be driveways or narrow access roads, which are not always labeled in the "truth" masks in the Massachusetts Roads Dataset. The mislabeled road segment above the white box also appears to be a real road segment missing from the "truth" mask – this segment may be unlabeled because it is part of a parking lot or private road, or may represent an error in the "truth" mask.

Finally, the parking lot in the bottom left of the image (boxed in green) is not labeled as a road in the "truth" mask, but parking lots are difficult to visually distinguish from roads, and some parking lots may provide "through access", serving the same purpose as a road. All four model-generated masks struggle to distinguish the parking lot from a road, and all masks label at least part of the parking lot as a road. The $C_8$-equivariant dense U-Net (mask shown in Figure 4-10(f)) labels the parking lot boundary, but no other part of the parking lot, as a road, and generally boasts the visually "smoothest" mask on this example. In contrast, the other three models (masks shown in Figure 4-10(c-e)) all label at least part of the interior of the parking lot as a road, and have visually "noisier" masks, qualitatively demonstrating poorer performance than the $C_8$-equivariant dense U-Net at interpreting a complex road network and distinguishing between parking lots and roads.

The performance metrics for the models presented in §4.1.1-4.1.4, evaluated on the Massachusetts Roads Dataset without a buffer, are given in Table 4.5, with the best performance on each metric bolded. The $C_8$-equivariant dense U-Net performs the best on all metrics except for balanced accuracy, sensitivity, and recall, on which the dense U-Net performs the best. Both $C_8$-equivariant models have higher specificity and precision but lower sensitivity and recall than their non-equivariant equivalents. This fits with the qualitative results seen in Figures 4-9 and 4-10; the $C_8$-equivariant

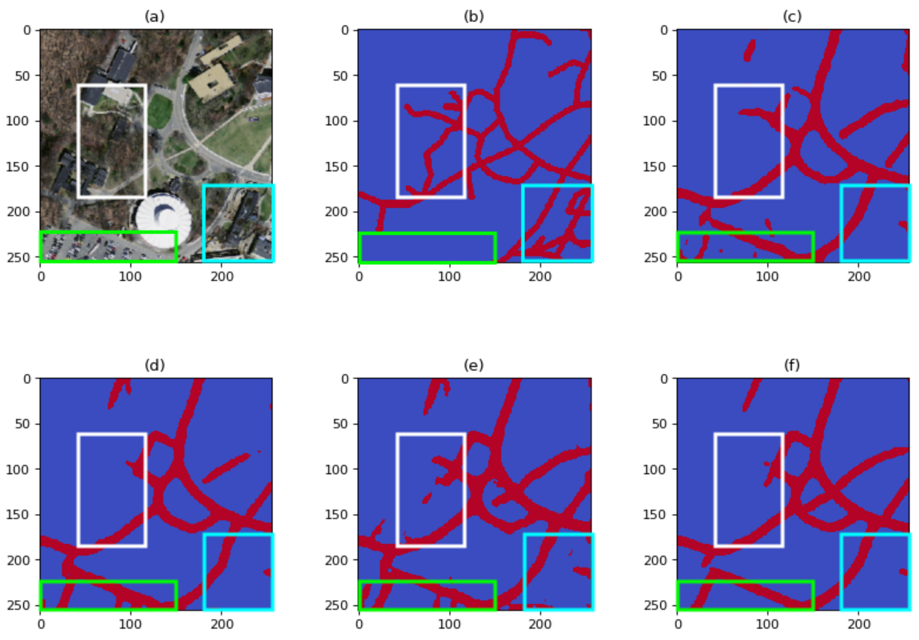


Figure 4-10: U-Net, dense U-Net, $C_8$-equivariant U-Net, and $C_8$-equivariant dense U-Net evaluated on a "hard" image segmentation sample, with narrow, partially occluded road segments boxed in white and light blue, and with a parking lot boxed in green. (a) Visible-spectrum input image, (b) "Truth" mask, (c) Mask generated by the U-Net, (d) Mask generated by the dense U-Net, (e) Mask generated by the $C_8$-equivariant U-Net, (f) Mask generated by the $C_8$-equivariant dense U-Net.

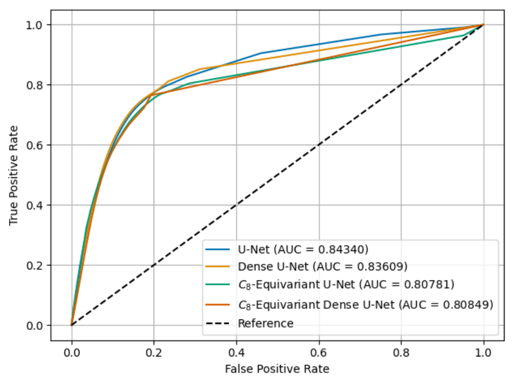
dense U-Net has "smoother" masks than the dense U-Net, and the $C_8$-equivariant U-Net misses some partially occluded road segments identified by the U-Net.

Table 4.5: Performance metrics detailed in §2.3.1 for the U-Net, dense U-Net, $C_8$-equivariant U-Net, and $C_8$-equivariant dense U-Net evaluated on the modified Massachusetts Roads Dataset, evaluated without using a buffer at road boundaries.

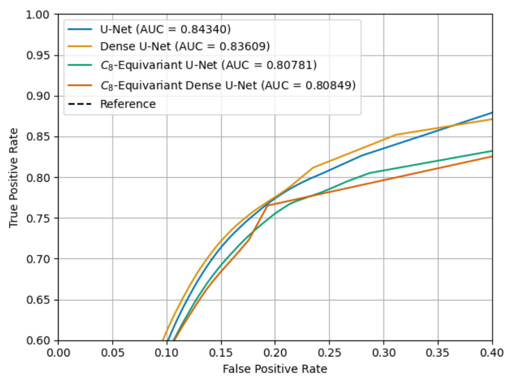
| **Model** | **Acc.** | **Bal. Acc.** | **Sens.** | **Spec.** | **Prec.** | **Recall** | **$F_1$** | **IoU** |
|---|---|---|---|---|---|---|---|---|
| U-Net | 80.22% | 78.71% | 77.02% | 80.39% | 17.83% | 77.02% | 0.2895 | 0.1693 |
| Dense U-Net | 79.75% | **78.77%** | **77.67%** | 79.86% | 17.47% | **77.67%** | 0.2853 | 0.1664 |
| $C_8$-Equivariant U-Net | 80.51% | 77.74% | 74.63% | 80.84% | 17.93% | 74.63% | 0.2891 | 0.1690 |
| $C_8$-Equivariant Dense U-Net | **83.52%** | 76.85% | 69.35% | **84.35%** | **20.49%** | 69.35% | **0.3163** | **0.1878** |

Figure 4-11 shows the receiver-operating characteristic (ROC) curve for each of the models presented in §4.1.1-4.1.4, evaluated without a buffer. Interestingly, the dense U-Net gets the closest to the $(0, 1)$ point, even though it has the worst $F_1$ score of the four models. This is likely because the Massachusetts Roads Dataset is highly imbalanced – under 10% of pixels are roads, so models achieve the best precision/recall tradeoff when specificity is high and sensitivity is low, and the $C_8$-equivariant models perform better in this high-specificity and low-sensitivity regime. This class imbalance explains how the $C_8$-equivariant dense U-Net has the best $F_1$ score by over 0.025 without having the highest balanced accuracy.

The performance metrics for the models presented in §4.1.1-4.1.4 when evaluated on the Massachusetts Roads Dataset with a 4 px buffer are given in Table 4.6, with the best performance on each metric bolded. The $C_8$-equivariant dense U-Net does best on all metrics other than balanced accuracy, sensitivity, and recall, just as when the models are evaluated without a buffer at road edges (see Table 4.5). This again highlights the $C_8$-equivariant dense U-Net's advantage on specificity and precision



(a) Original view.



(b) Zoomed-in view.

Figure 4-11: Receiver-operating characteristic (ROC) curves for the U-Net, dense U-Net, $C_8$-equivariant U-Net, and $C_8$-equivariant dense U-Net, evaluated on the modified Massachusetts Roads Dataset without using a buffer at road boundaries.

over sensitivity and recall, as demonstrated by its "smooth" masks which eliminate noisy false positives at the expense of sometimes missing partially occluded road segments, as seen in Figures 4-9 and 4-10.

Notably, the dense U-Net has the second-highest $F_1$ score when a 4 px buffer is taken into account, even though it has the lowest $F_1$ score when no buffer is taken into account. This likely relates to the dense U-Net's low specificity – the dense U-Net is biased towards false positives over false negatives, and many false positives occur due to differences in road width between masks, so including a 4 px buffer at road boundaries eliminates some of these false positives and improves the dense U-Net's performance.

Also, the U-Net slightly outperforms the $C_8$-equivariant U-Net both without a buffer (see Table 4.6) and with a 4 px buffer at the road boundaries. This is likely because the low number of channels per orientation used in the $C_8$-equivariant U-Net limits its performance; we expect that if the U-Net used the same number of *total channels* as the *channels per orientation* used by the $C_8$-equivariant U-Net, the $C_8$-equivariant U-Net would significantly outperform the U-Net. Our dense models have fewer parameters per channel than the U-Net and $C_8$-equivariant U-Net; as a result, we are able to use more channels in the dense models, and the $C_8$-equivariant dense U-Net clearly outperforms the dense U-Net, demonstrating that with a sufficient number of channels, $C_8$-equivariant models can outperform non-equivariant models. In the future, we plan to test the U-Net and $C_8$-equivariant U-Net with 64 input channels and 8 channels per orientation, respectively – see §6.2 for a full description of our planned future work.

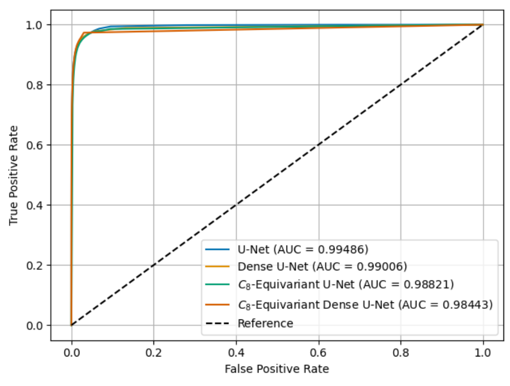
Table 4.6: Performance metrics detailed in §2.3.1 for the U-Net, dense U-Net, $C_8$-equivariant U-Net, and $C_8$-equivariant dense U-Net evaluated on the modified Massachusetts Roads Dataset, evaluated with a 4 px buffer at road boundaries.

| **Model** | **Acc.** | **Bal. Acc.** | **Sens.** | **Spec.** | **Prec.** | **Recall** | **F$_1$** | **IoU** |
|---|---|---|---|---|---|---|---|---|
| U-Net | 96.22% | **96.46%** | 96.85% | 96.06% | 86.08% | 96.85% | 0.9115 | 0.8374 |
| Dense U-Net | 96.19% | 96.45% | **96.90%** | 96.01% | 86.29% | **96.90%** | 0.9129 | 0.8397 |
| $C_8$-Equivariant U-Net | 96.29% | 96.35% | 96.46% | 96.25% | 86.38% | 96.46% | 0.9114 | 0.8372 |
| $C_8$-Equivariant Dense U-Net | **97.58%** | 96.33% | 94.37% | **98.29%** | **92.49%** | 94.37% | **0.9342** | **0.8765** |

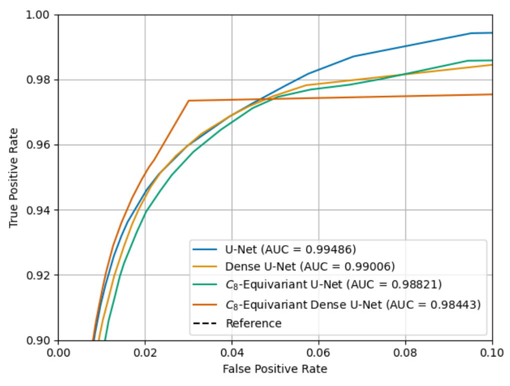
Figure 4-12 shows the ROC curves representing the models presented in §4.1.1-4.1.4 evaluated on the Massachusetts Roads Dataset with a 4 px buffer. The $C_8$-equivariant dense U-Net gets closest to the $(0, 1)$ point indicating perfect classification, followed by the dense U-Net, then by the U-Net, then finally by the $C_8$-equivariant U-Net. This is the same order as the order of the $F_1$ scores given in Table 4.6. However, the curves representing the dense U-Net and U-Net are extremely close, closer than indicated by the $F_1$ scores in Table 4.6. This likely indicates that with a road detection threshold other than 0.5, the U-Net would perform better.

## Resource Utilization

Table 4.7 presents the resource utilization of the models presented in §4.1.1-4.1.4. The peak memory allocation using a GPU backend is much higher for the dense models, and is lower for the $C_8$-equivariant models than it is for their non-equivariant



(a) Original view.



(b) Zoomed-in view.

Figure 4-12: Receiver-operating characteristic (ROC) curves for the U-Net, dense U-Net, $C_8$-equivariant U-Net, and $C_8$-equivariant dense U-Net, evaluated on the modified Massachusetts Roads Dataset without a 4 px buffer at road boundaries.

equivalents; as a result, the dense U-Net has the highest peak memory allocation using a GPU backend. The dense models have more channels than their non-dense equivalents, which contributes to their consumption of GPU memory, and the $C_8$-equivariant models have fewer parameters than their non-equivariant equivalents, slightly reducing the amount of GPU memory they consume.

The dense models have slower inference time than their non-dense equivalents using both a CPU and GPU backend, and the $C_8$-equivariant models also have slower inference time than their non-equivariant equivalents using both backends. As a result, the $C_8$-equivariant dense U-Net has the slowest inference time regardless of backend. This is unsurprising – the dense models have 4x more channels than the non-dense models, so should have slower inference time, and as seen in §3.1.7, $C_8$-equivariant models generally take longer to classify each image, likely because of overhead introduced by the `e2cnn` library related to checking group representations and differences in speed optimization between PyTorch and the `e2cnn` library.

Table 4.8 shows the size of the saved model, number of parameters, and number of trainable parameters for the models presented in §4.1.1-4.1.4. The dense models, which have more channels than the non-dense models, have more parameters and require more storage than their non-dense equivalents. The $C_8$-equivariant models have fewer parameters and require less storage than their non-equivariant equivalents.

Table 4.7: Peak memory usage over 100 single-image classifications and average inference time over 1000 single-image classifications using both a CPU (Intel Xeon) and GPU (Nvidia K80) backend for the U-Net, dense U-Net, $C_8$-equivariant U-Net, and $C_8$-equivariant dense U-Net on the modified Massachusetts Roads Dataset.

| **Model** | **GPU Mem. (GiB)** | **CPU Mem. (KiB)** | **GPU Inf. Time (s)** | **CPU Inf. Time (s)** |
|---|---|---|---|---|
| U-Net | 0.286 | 83.7 | 0.0054 | 0.1506 |
| Dense U-Net | 1.532 | 111.3 | 0.0146 | 0.5673 |
| $C_8$-Equivariant U-Net | 0.272 | 131.5 | 0.0147 | 0.1735 |
| $C_8$-Equivariant Dense U-Net | 1.486 | 83.7 | 0.0557 | 0.6673 |

This highlights that $C_8$-equivariance reduces the parameter space and thus the number of parameters per model when the number of channels is held equal, and also that an increase in channels increases the number of model parameters and model size.

Table 4.8: Saved model size, total number of parameters, and number of trainable parameters for the U-Net, dense U-Net, $C_8$-equivariant U-Net and $C_8$-equivariant dense U-Net trained on the modified Massachusetts Roads Dataset.

| **Model** | **Model Size** | **Total Parameters** | **Trainable Parameters** |
|---|---|---|---|
| U-Net | 12.5 MB | $1.54 \times 10^6$ | $1.54 \times 10^6$ |
| Dense U-Net | 21.6 MB | $2.64 \times 10^6$ | $2.64 \times 10^6$ |
| $C_8$-Equivariant U-Net | 7.4 MB | $1.32 \times 10^5$ | $1.32 \times 10^5$ |
| $C_8$-Equivariant Dense U-Net | 13.3 MB | $2.94 \times 10^5$ | $2.90 \times 10^5$ |

THIS PAGE INTENTIONALLY LEFT BLANK

# Chapter 5

# Discussion

In this chapter, we quantitatively compare our cloud segmentation and road segmentation results to the literature, and also quantitatively compare our cloud segmentation results to our road segmentation results. We use these comparisons to motivate qualitative discussion of the differences between the road and cloud domains, and to motivate promising areas of future work.

## 5.1 Cloud Segmentation Results vs. Literature

We designed our cloud detection experiments to use the SPARCS dataset, which was originally created for the SPARCS CNN [25]. Our cloud detection models, unlike the SPARCS CNN, use only a limited selection of Landsat 8 bands, and predict clouds only rather than predicting several different terrain classes. Additionally, our cloud detection models classify 144 x 144 pixel patches of input images by default rather than classifying 256 x 256 pixel patches like the SPARCS CNN [25]. Nevertheless, we use the same training and test data, and make comparisons between the different models. Table 5.1 summarizes our results alongside the results found by Hughes and Kennedy in [25]. Note that only metrics calculated with a 2 px buffer at cloud boundaries are available for the SPARCS CNN, so we present all metrics using a 2 px buffer at cloud boundaries.

As shown in Table 5.1, the SPARCS CNN, which is trained on Landsat 8 data

Table 5.1: Performance metrics calculated for the SPARCS CNN [25] and for the cloud segmentation models presented in Chapter 3. All results are calculated on the SPARCS dataset using a 2 px buffer at the cloud boundaries.

| **Model** | **Precision** | **Recall** | $\mathbf{F_1}$ | **IoU** | **Trainable Parameters** |
|---|---|---|---|---|---|
| SPARCS CNN [25] | 95.73% | 96.42% | 0.9607 | 0.9244 | $2.05 \times 10^7$ |
| Luminosity Thresholding | 60.20% | 43.18% | 0.5029 | 0.3359 | – |
| Random Forest | 92.11% | 96.91% | 0.9445 | 0.8948 | – |
| U-Net | 98.46% | 96.54% | 0.9749 | 0.9511 | $2.46 \times 10^7$ |
| Dense U-Net | 98.47% | 96.89% | 0.9768 | 0.9546 | $2.64 \times 10^6$ |
| $C_8$-Equivariant U-Net | 98.78% | 97.18% | 0.9797 | 0.9602 | $2.10 \times 10^6$ |
| $C_8$-Equivariant Dense U-Net | 98.82% | 97.32% | 0.9806 | 0.9620 | $2.90 \times 10^5$ |
| $C_8$-Equivariant Dense U-Net (VIS-only) | 98.70% | 93.54% | 0.9605 | 0.9240 | $2.90 \times 10^5$ |
| $C_8$-Equivariant Dense U-Net (VIS + LWIR) | 97.78% | 95.98% | 0.9687 | 0.9393 | $2.90 \times 10^5$ |
| $C_8$-Equivariant Dense U-Net (VIS + SWIR) | 99.50% | 93.39% | 0.9635 | 0.9296 | $2.90 \times 10^5$ |

from 10 different bands, outperforms our luminosity thresholding algorithm, random forest algorithm, and $C_8$-equivariant dense U-Net trained on visible-spectrum data only in terms of $F_1$ score, but underperforms all of our other models in terms of $F_1$ score. One possible reason for this difference in performance is that the SPARCS CNN is designed to distinguish clear-sky, water, snow/ice, and cloud shadow classes in addition to identifying clouds, while our models are fully optimized for cloud segmentation. However, the SPARCS CNN has significantly more parameters than any of our models other than the U-Net, so it should have enough parameters to segment images into five different classes.

It is more likely that most of our deep learning models outperform the SPARCS CNN because of a combination of translation equivariance, dense blocks, and rotation equivariance. The SPARCS CNN uses max pooling operators, which lead to aliasing and inhibit translation equivariance – meaning that the SPARCS CNN may classify input images differently based on small translational shifts in cloud locations [53] [25]. Our models use blur convolutions before max pooling and thus are less susceptible to this type of aliasing [53]. Additionally, our $C_8$-equivariant and dense models make further performance gains over the SPARCS CNN and our U-Net by leveraging rotational equivariance and the improved gradient flow through dense networks [30] [50].

## 5.2 Comparison of Road Segmentation Results to Literature

Bandara *et al.* developed the spatial and interaction space graph reasoning (SPIN) module, which uses graph reasoning over spatial space to learn associations between different spatial regions in an input image, and uses graph reasoning over a projected interaction space to improve discrimination between roads and other features [5]. Bandara *et al.* then tested their SPIN Road Mapper architecture against six other model architectures on the Massachusetts Roads Dataset [5].

We designed our road segmentation experiments to use the same experimental setup and parameters as Bandara *et al.*, with two exceptions: we used dice loss (see §2.4.2) rather than a combination of softIOU and orientation loss, and we trained our models for 90 epochs instead of 120 epochs because we saw an empirical increase in training and validation error after 90 epochs during training. Dice loss is known to have better gradient properties than softIOU loss [45], so it makes sense that our models, which were trained using dice loss, would converge more quickly and require fewer training epochs than the models in [5], which were trained with a combination of softIOU and orientation loss. Table 5.2 summarizes our results and the results found by Bandara *et al.* [5].

Our four deep learning methods significantly outperform all seven of the methods examined in [5] when a 4 px buffer at road boundaries is used during evaluation, but our methods significantly underperform those in [5] when no buffer is used for evaluation. This difference likely has to do with the width of the road segments in model-generated masks; each of our four algorithms generated masks with signifi-

Table 5.2: Performance metrics for the models evaluated in [5] and for the road segmentation models presented in Chapter 4. All metrics are calculated with a 4 px buffer at road boundaries unless otherwise specified.

| **Model** | **Precision** | **Recall** | $\mathbf{F_1}$ | **IoU** | **IoU (no buffer)** |
|---|---|---|---|---|---|
| Seg-Net [5][4] | 77.34% | 79.84% | 0.7857 | 0.6471 | 0.5859 |
| U-Net [5][42] | 82.46% | 84.34% | 0.8339 | 0.7151 | 0.6097 |
| LinkNet [5][11] | 83.25% | 84.63% | 0.8393 | 0.7232 | 0.6312 |
| HourGlass [5][40] | 81.26% | 81.86% | 0.8156 | 0.6886 | 0.6137 |
| Stack-HourGlass [5][40] | 80.12% | 83.87% | 0.8196 | 0.6943 | 0.6221 |
| Batra *et al.* [5][7] | 83.34% | 84.61% | 0.8397 | 0.7237 | 0.6444 |
| SPIN RoadMapper [5] | 83.90% | 85.06% | 0.8447 | 0.7312 | 0.6524 |
| U-Net | 86.08% | 96.85% | 0.9115 | 0.8374 | 0.1693 |
| Dense U-Net | 86.29% | 96.90% | 0.9129 | 0.8397 | 0.1664 |
| $C_8$-Equivariant U-Net | 86.38% | 96.46% | 0.9114 | 0.8372 | 0.1690 |
| $C_8$-Equivariant Dense U-Net | 92.49% | 94.37% | 0.9342 | 0.8765 | 0.1878 |

cantly wider road segments than the "truth" masks, but the masks generated by the models examined in [5] include road segments of the same width as the segments in the "truth" masks. Using a 4 px buffer eliminates nearly all of the false positives related to road segment width, and allows a fairer comparison between the two sets of models.

The U-Net examined in [5] uses the original U-Net architecture from [42], while our U-Net uses an adjusted architecture (see 2.2.3) which adds batch-normalizations and replaces max-pooling operations with max-blur-pooling in order to reduce aliasing. Nevertheless, these architectural adjustments do not fully explain the large gap in performance between the two models. It is likely that the difference in loss functions is the primary cause of the performance gap – it seems that the models from [5] are optimized to generate masks with narrow road segments, at the cost of slightly reduced accuracy. We believe that orientation loss is a key contributor to the tendency to generate masks with narrow road segments, because we have found that the U-Net and our other model architectures did not converge using softIOU loss and thus believe that the loss function in [5] must be dominated by orientation loss. In the future, we plan to reproduce the results of [5] in order to better investigate the impact of orientation loss; see §6.2 for further discussion of our planned future work.

Even with the differences in architecture, we believe that the two U-Nets provide an important baseline for interpreting each set of experiments. In the experiments performed by Bandara *et al.*, the SPIN RoadMapper is the best-performing model on all metrics and it outperforms the U-Net by 0.0108 in terms of $F_1$ score [5]. In our experiments, the $C_8$-equivariant dense U-Net has the best $F_1$ score and outperforms the U-Net by 0.0227 in terms of $F_1$ score – the gap between the U-Net and the best-performing model is twice as large. We believe that our $C_8$-equivariant dense U-Net shows a great deal of promise, and that combining the $C_8$-equivariant dense U-Net with the SPIN module developed in [5] is an important area of future work.

## 5.3 Comparison of Road and Cloud Segmentation Results

We can improve our understanding of the benefits and challenges of applying rotation-equivariant models to different domains by comparing the cloud segmentation results presented in Chapter 3 to the road segmentation results presented in Chapter 4. Table 5.3 summarizes the results of our U-Net, dense U-Net, $C_8$-equivariant U-Net, and $C_8$-equivariant dense U-Net when applied to both the SPARCS dataset for cloud segmentation, using a 2 px buffer at cloud boundaries and the Massachusetts Roads Dataset for road segmentation, using a 4 px buffer at road boundaries.

Even though the buffer at road boundaries (4 pixels) is larger than the buffer used at cloud boundaries (2 pixels), the cloud models significantly outperform the road models in terms of $F_1$ score. This is likely related to differences between the SPARCS and Massachusetts Roads Dataset; in general, the labels available for the SPARCS dataset seem to be much more accurate than those available for the Massachusetts Roads Dataset.

More interestingly, the $C_8$-equivariant U-Net modestly outperforms the U-Net on the SPARCS dataset, but performs worse than the U-Net on the Massachusetts Roads

Table 5.3: Performance metrics for the cloud segmentation models presented in Chapter 3 and the road segmentation models presented in Chapter 4. All metrics are calculated with a 2 px buffer at the cloud boundaries for cloud segmentation models and with a 4 px buffer at the road boundaries for road segmentation models.

| **Model** | **Precision** | **Recall** | **F$_1$** | **IoU** | **Trainable Parameters** |
|---|---|---|---|---|---|
| U-Net (Cloud) | 98.46% | 96.54% | 0.9749 | 0.9511 | 2.46 $\times 10^7$ |
| Dense U-Net (Cloud) | 98.47% | 96.89% | 0.9768 | 0.9546 | 2.64 $\times 10^6$ |
| $C_8$-Equivariant U-Net (Cloud) | 98.78% | 97.18% | 0.9797 | 0.9602 | 2.10 $\times 10^6$ |
| $C_8$-Equivariant Dense U-Net (Cloud) | 98.82% | 97.32% | 0.9806 | 0.9620 | 2.90 $\times 10^5$ |
| U-Net (Road) | 86.08% | 96.85% | 0.9115 | 0.8374 | 1.54 $\times 10^6$ |
| Dense U-Net (Road) | 86.29% | 96.90% | 0.9129 | 0.8397 | 2.64 $\times 10^6$ |
| $C_8$-Equivariant U-Net (Road) | 86.38% | 96.46% | 0.9114 | 0.8372 | 1.32 $\times 10^5$ |
| $C_8$-Equivariant Dense U-Net (Road) | 92.49% | 94.37% | 0.9342 | 0.8765 | 2.90 $\times 10^5$ |

Dataset. The poor performance of the $C_8$-equivariant U-Net on the Massachusetts Roads Dataset can be partially attributed to the fact that it only uses 2 input channels in its first convolution block, and has 4x fewer channels than both the $C_8$-equivariant dense U-Net and the $C_8$-equivariant U-Net used on the SPARCS dataset. This is also reflected in the number of trainable parameters of each model. Because the $C_8$-equivariant U-Net uses 4x fewer channels on the Massachusetts Roads Dataset than on the SPARCS dataset, it has roughly 16x fewer parameters – for a model with $c$ channels, the number of parameters scales with $c^2$, as the number of parameters in each convolution scales with the number of channels in the input and the number of channels in the output.

Compared with the $C_8$-equivariant U-Net, the $C_8$-equivariant dense U-Net shows large gains over the dense U-Net on the Massachusetts Roads Dataset, improving $F_1$ score by 0.0213 when a buffer is used and by 0.0310 when no buffer is used for evaluation. However, the $C_8$-equivariant dense U-Net demonstrates only a small improvement over the dense U-Net on the SPARCS dataset, improving $F_1$ score by 0.0038 when a buffer is used and by 0.0086 when no buffer is used for evaluation. We believe that this difference in the level of improvement shown by the $C_8$-equivariant dense U-Net across the cloud segmentation and road segmentation domains is related to differences in topology between road networks and clouds. As shown in §2.2.2, the random forest algorithm performs fairly well on cloud segmentation on the SPARCS dataset despite only considering a 3x3 patch of pixels during classification. In contrast, the random forest algorithm achieves an $F_1$ score of zero when applied to the Massachusetts Roads Dataset, demonstrating that large-scale spatial context is necessary to accurately extract road networks.

It follows that rotation-equivariance is relatively more helpful on the road segmentation domain than on the cloud segmentation domain. It also seems that spatial information is relatively more important (and spectral information less important) for road segmentation than for cloud segmentation. However, the Massachusetts Roads Dataset only includes visible-spectrum imagery, which is insufficient to fully study the spectral characteristics of road networks. A more comprehensive investigation of road segmentation using multispectral data is needed before final conclusions can be drawn about the importance of spectral information for road segmentation.

THIS PAGE INTENTIONALLY LEFT BLANK

# Chapter 6

# Summary and Future Work

## 6.1 Summary of Results

In this thesis, we present four different machine learning algorithms, including two $C_8$-equivariant machine learning models, for segmenting satellite and aerial imagery. We evaluate these algorithms on both the cloud and road segmentation domains and we compare these algorithms to two rule-based algorithms on the cloud segmentation domain. We evaluate the performance of the $C_8$-equivariant dense U-Net on cloud segmentation when trained on visible-spectrum data only, visible-spectrum data augmented with LWIR data, visible-spectrum data augmented with SWIR data, and visible-spectrum data augmented with LWIR and SWIR data.

We demonstrate that the $C_8$-equivariant dense U-Net produces the most accurate segmentation maps on both the cloud segmentation domain and road segmentation domain, achieving an $F_1$ score of 0.9806 on the cloud segmentation dataset when evaluated with a 2 px buffer at the cloud boundaries, and achieving an $F_1$ score of 0.9342 on the road segmentation dataset when evaluated with a 4 px buffer at the road boundaries. Human interpreters show self-consistency of around 96% on the SPARCS cloud segmentation dataset when evaluated with a 2 px buffer at the cloud boundaries [25]; in contrast, the $C_8$-equivariant dense U-Net showed an accuracy of 99.54% when evaluated on the SPARCS dataset with a 2 px buffer at the cloud boundaries. The $C_8$-equivariant dense U-Net has only around 290,000 trainable parameters – the fewest of all deep learning models evaluated on the cloud segmentation dataset and the second-fewest of all deep learning models evaluated on the road segmentation dataset.

On the road segmentation domain, the $C_8$-equivariant U-Net uses the least memory, taking only 0.28 GiB on a Google Colab GPU, and the U-Net classifies images the most quickly, taking only 0.0054 seconds to classify an image using a Google Colab GPU and 0.1506 seconds to classify an image using a CPU backend. On the cloud segmentation domain, the $C_8$-equivariant U-Net again uses the least memory, taking only 448.3 MiB on a Google Colab GPU, the U-Net classifies images the most quickly when using a GPU backend (taking only 0.0092 seconds), and the dense U-Net classifies images the most quickly when using a CPU backend, taking only 0.1801 seconds.

Nevertheless, the $C_8$-equivariant dense U-Net is a strong fit for deployment on resource-constrained platforms. On the cloud segmentation domain, it takes only 0.2281 seconds to classify an image using a CPU backend, and requires under 15 MB of memory, making it a good fit for missions without onboard GPUs. On the road segmentation domain, it takes 0.6673 seconds to classify an image using a CPU backend, and requires under 13.5 MB of memory. The $C_8$-equivariant dense U-Net maintains strong performance when trained on input images for cloud segmentation with only three or four bands; when trained on visible-spectrum data only it has an $F_1$ score of 0.9605, when trained on visible-spectrum and LWIR data it achieves an $F_1$ score of 0.9687, and when trained on visible-spectrum and SWIR data it achieves an $F_1$ score of 0.9635. This makes the $C_8$-equivariant dense U-Net a good fit for resource-constrained missions like CubeSats, which may not be able to host instruments capable of resolving visible-spectrum, LWIR, and SWIR imagery.

## 6.2 Future Work

In this thesis, we explore the use of rotation-equivariant machine learning on satellite and aerial imagery. We find many promising areas of future inquiry related to this topic. We will group these avenues of future research into three major areas:

architectural investigations, on-orbit considerations, and future applications.

### 6.2.1 Architectural Investigations

In §5.2, we evaluate $C_8$-equivariant machine learning algorithms on segmenting the Massachusetts Roads Dataset, and compare our results to the state-of-the-art in the literature. Current state-of-the-art approaches for road detection and road network extraction combine segmentation and orientation learning [7] [5]. It would be interesting to replace the segmentation layers in a state-of-the-art model like SPIN RoadMapper with $C_8$-equivariant layers and evaluate how this affects performance.

As a first step towards combining $C_8$-equivariant segmentation layers with orientation learning, we plan to replicate Bandara *et al.*'s implementation of SPIN RoadMapper in order to investigate the impact of orientation loss on road segmentation, especially on the width of road segments in the output maps generated by SPIN RoadMapper [5].

More generally, researchers are publishing new and promising image segmentation and object detection architectures each day; ConvNext [33] and ConvUNext [20] are two new such state-of-the-art models. One potential area of future inquiry involves developing a $C_8$-equivariant version of these models and analyzing the resulting changes in performance across different image segmentation and object detection domains.

Finally, as explained in Chapter 4, we have not yet been able to test our U-Net and $C_8$-equivariant U-Net on segmenting the Massachusetts Roads Dataset with 64 input channels and 8 channels per orientation, respectively. We plan to train these models in order to better understand the costs and benefits of $C_8$-equivariance for road segmentation. However, these models will not fit on the GPU we have been using for training, so we will need to train these models using only CPUs, or use different computational infrastructure.

### 6.2.2 On-Orbit Considerations

Our models were trained on a high-performance computing cluster and were evaluated on a laptop; these models are not yet ready to be deployed to real-time operating systems (RTOS) and flight hardware. In the future, we plan to adjust our models to be more easily deployed on-orbit. In particular, we plan to implement functionality to export our $C_8$-equivariant models to have a pure PyTorch backend with no other dependencies, and then use existing PyTorch tooling to translate our models to TorchScript, which can be run directly from C++ on embedded hardware with no Python dependencies.

Our models are trained on Landsat and aerial data rather than on a mission-specific dataset. Before deploying our models on-orbit, it will likely be useful to create a new dataset with images collected directly from target regions and to retrain the models on the mission-specific dataset. This will help optimize model performance and is especially important for on-orbit applications like identifying clouds over water; our cloud segmentation models are trained on Landsat data and as such all training images were taken over land.

### 6.2.3 Future Applications

We believe that there are a number of promising applications for on-orbit rotation-equivariant machine learning. For example, on-orbit rotation-equivariant machine learning could be used to learn and identify road morphologies associated with development, construction, and natural disaster. This would be useful for tracking road construction, which can be correlated with illegal logging [52], and for mapping areas affected by natural disasters to aid relief efforts [14].

Another application of rotation-equivariant machine learning is learning on time-series data in order to track the movement of clouds over time. Another cloud-related application is learning cloud morphology. If these two applications are combined, the ability to identify the movement of and morphology of clouds would be useful for near-term weather prediction. Finally, learning to identify cloud shadows in addition to clouds, generating a three-class segmentation map, would be useful for identifying regions free of cloud and cloud shadow – for many remote sensing applications, regions shadowed by clouds are undesirable [35]. We plan to leverage the SPARCS dataset, which has existing cloud shadow labels, to train models that can identify both clouds and cloud shadows.

THIS PAGE INTENTIONALLY LEFT BLANK

# Appendix A

# $C_n$-equivariant Convolutions

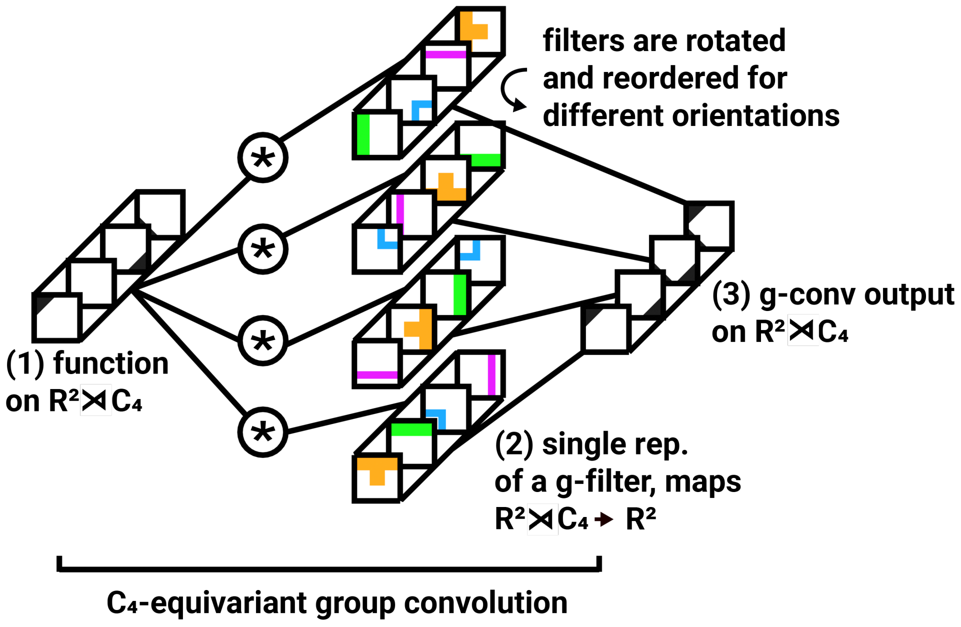
In §2.2.5, we describe our $C_8$-equivariant U-Net architecture; in this appendix, we illustrate the properties of $C_n$-equivariant deep learning models, which are described and rigorously proved in [51]. For simplicity of illustration, we illustrate a $C_4$-equivariant CNN with one channel per orientation, which consists solely of an input convolution, group convolution, and orientation pooling layer. Nevertheless, the properties of this network generalize to models equivariant to an arbitrary cyclic group $C_n$ with an arbitrary depth and arbitrary number of channels per orientation.

First, Figure A-1 shows a $C_4$-equivariant group convolution. The input and output feature maps of this convolution are functions on the semidirect product group $\mathbb{R}^2 \rtimes C_4 \leq SE(2)$ [51], meaning that $C_4$ (the cyclic group of rotations of $\frac{k360^\circ}{4}$, where $k$ is an integer) is acting on $\mathbb{R}^2$ (in this case representing 2D images), to form the group over which the input and output feature maps are functions. This group is in turn a subgroup of $SE(2)$, the special Euclidean group, which represents "rigid motions" (rotations and translations) in the 2D plane. In contrast, in traditional CNNs, the inputs and outputs of convolutions are functions on $\mathbb{R}^2$, and rotational symmetry is not necessarily preserved.

For the group convolution shown in Figure A-1, there are 4 steerable filters (because $|C_4| = 4$) which make up a single group filter, or g-filter. These filters are rotated by $\frac{360^\circ}{4}$ and reordered to create 4 different filter representations. Each of these filter representations is convolved with the input feature map, and then the resulting layers are linearly combined to create a single representation in the output feature map. This process is shown in further detail in Figure A-2.

In Figure A-2, part of the group convolution from Figure A-1 is shown. The input feature map is convolved with a single representation of the group filter, and the output is shown. Then, the layers of the output are "flattened", or linearly combined, to create the first representation in the output feature map. The other three representations in the output feature map are generated by convolution with the other three filter representations.

Figure A-1: A $C_4$-equivariant group convolution. Image credit: Alex Meredith, MIT.



In Figure A-3, we show an input convolution that "lifts" a 2D image on $\mathbb{R}^2$ to the group $\mathbb{R}^2 \rtimes C_4$ by convolving it with a single steerable filter that has been repeatedly rotated to become a filter on $C_4$. The output of this convolution is a feature map on $\mathbb{R}^2 \rtimes C_4$ that can be used as an input for a group convolution.

In Figure A-4, we show an "orientation pooling" operation, which pools a feature map on $\mathbb{R}^2 \rtimes C_4$ back to a 2D image on $\mathbb{R}^2$ in order to create an output mask. "Orientation pooling" simply evaluates a pooling operation orientation-wise – for example, in "max orientation pooling", the maximum over $n$ orientations is taken for each pixel, which is equivalent to channel-wise max pooling in a typical CNN if the orientation

Figure A-2: Part of the $C_4$-equivariant group convolution shown in Figure A-1. (a) The input function is convolved with a single representation of a g-filter, (b) A linear combination is performed on the output of the convolution done in (a), which results in a single orientation map which is part of the group convolution output (see Figure A-1). Image credit: Alex Meredith, MIT.

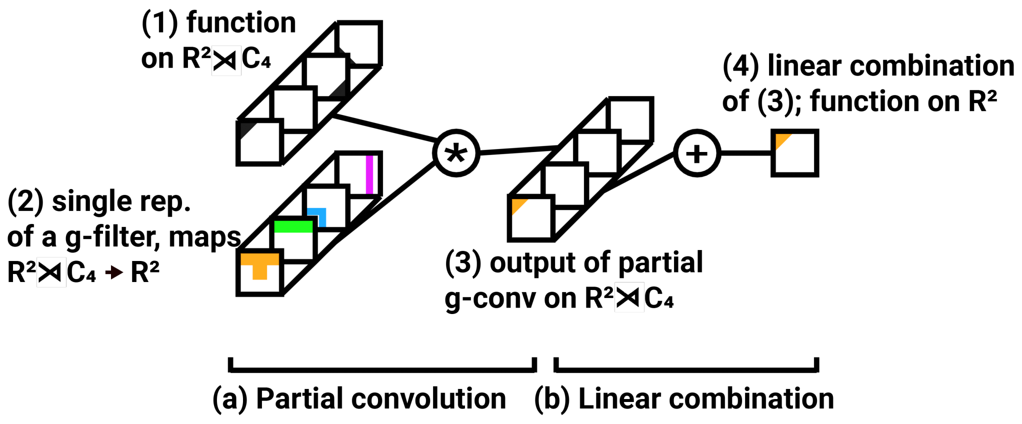
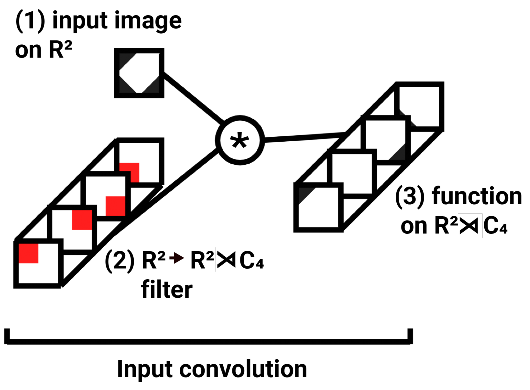

(1) function on R²⋊C₄
(2) single rep. of a g-filter, maps R²⋊C₄ → R²
(3) output of partial g-conv on R²⋊C₄
(4) linear combination of (3); function on R²
(a) Partial convolution
(b) Linear combination

Figure A-3: A $C_4$-equivariant input convolution, which “lifts” an input on $\mathbb{R}^2$ to $\mathbb{R}^2 \rtimes C_4$. Image credit: Alex Meredith, MIT.



maps are taken to be channels.

Figure A-4: An orientation max pooling operation, which generates an output mask on $\mathbb{R}^2$. Image credit: Alex Meredith, MIT.

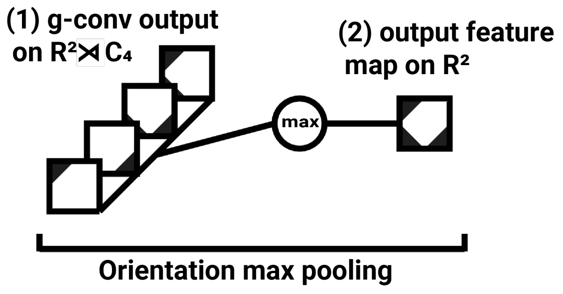
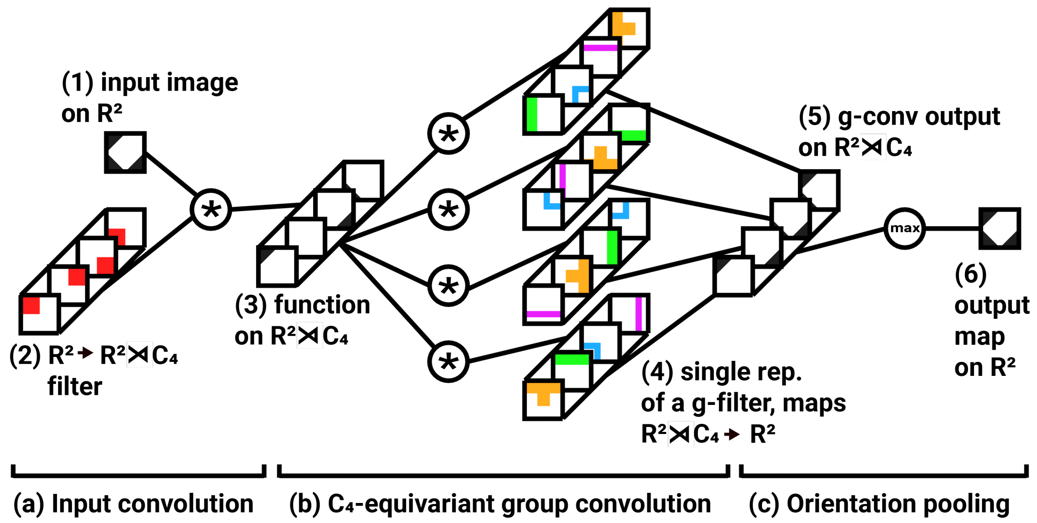


Figure A-5, which reproduces Figure 2-7, combines the input convolution from A-3, the group convolution from A-1, and the orientation pooling orientation from A-4, showing a simple $C_4$-equivariant CNN.

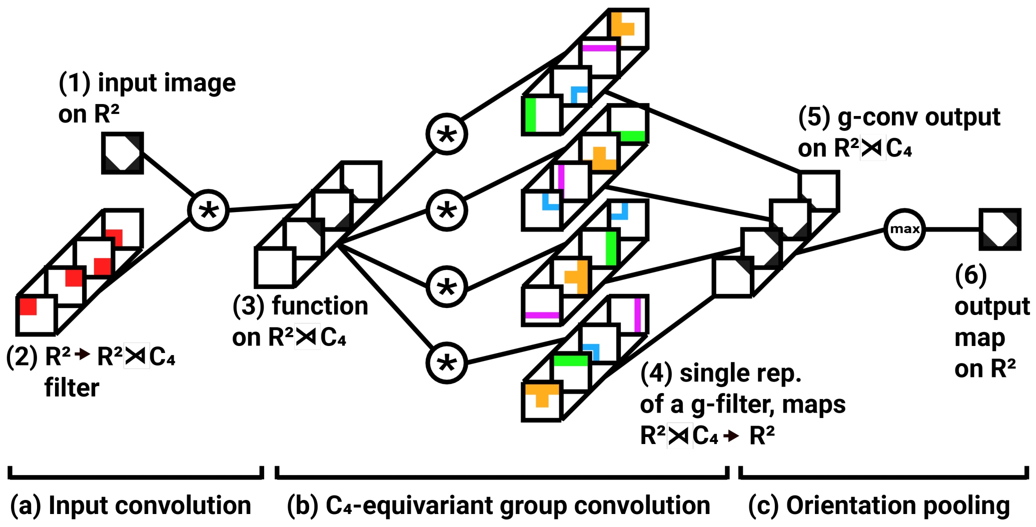
Figure A-5: A simple $C_4$-equivariant network. (a) An input convolution that lifts an input image to $\mathbb{R}^2 \rtimes C_4$, (b) A $C_4$-equivariant group convolution, (c) Orientation pooling to produce an output map on $\mathbb{R}^2$. Image credit: Alex Meredith, MIT.



In Figure A-6, we show what happens when the $C_4$-equivariant network from Figure A-5 is applied to an input image that has been rotated by 90°. The intermediate feature map outputted by the input convolution shows the same features as the intermediate feature map in Figure A-5; however, the features are rotated and reordered. The rotation and reordering is preserved by the group convolution – the output feature maps from Figures A-5 and A-6 also share the same features, just rotated and reordered in the same way as the intermediate feature maps.

Finally, the output of the max pooling operation in Figure A-6 is the same as the output of the max pooling operation in Figure A-5, but rotated by 90° like the input image. Because orientation-wise average and max pooling are ordering-agnostic, the rotation by 90° of the intermediate feature maps is preserved by the max-pooling operation, but the ordering of the feature maps is discarded by the orientation pooling layer. From input to output, the network shown in Figures A-5 and A-6 is $C_4$-equivariant.

Figure A-6: The same $C_4$-equivariant network shown in Figure A-5, but applied to an input image that has been rotated by 90°. (a) An input convolution that lifts an input image to $\mathbb{R}^2 \rtimes C_4$, (b) A $C_4$-equivariant group convolution, (c) Orientation pooling to produce an output map on $\mathbb{R}^2$. Image credit: Alex Meredith, MIT.



THIS PAGE INTENTIONALLY LEFT BLANK

# Appendix B

# SoftIoU tables

This appendix presents the segmentation performance metrics for the deep learning models presented in §2.2.3-2.2.6, applied to the Massachusetts Roads Dataset and trained with Bandara *et al.*'s implementation of softIOU loss [5]. Notably, the U-Net and dense U-Net used to generate results for this section were trained with 8x fewer channels than the models presented in Chapter 4; the U-Net was trained with 2 input channels and the dense U-Net was trained with 16 input channels. The $C_8$-equivariant U-Net and $C_8$-equivariant dense U-Net were trained with 2 input channels per orientation and 16 input channels per orientation, respectively.

As demonstrated in Tables B.1 and B.2, none of the four models trained using softIOU loss achieved an accuracy better than 60% with no buffer or 80% when using a 4 pixel buffer at road boundaries, even after 90 epochs; none of the models demonstrated good convergence, meaning all four models had similar levels of training, validation, and test error at the beginning of training and after 90 epochs.

Table B.1: Performance metrics detailed in §2.3.1 for the U-Net, dense U-Net, $C_8$-equivariant U-Net, and $C_8$-equivariant dense U-Net evaluated on the modified Massachusetts Roads Dataset using softIOU loss, evaluated without using a buffer at road boundaries.

| **Model** | **Acc.** | **Bal. Acc.** | **Sens.** | **Spec.** | **Prec.** | **Recall** | $\mathbf{F}_1$ | **IoU** |
|---|---|---|---|---|---|---|---|---|
| U-Net | 31.13% | 41.24% | 52.77% | 29.72% | 4.66% | 52.77% | 0.0856 | 0.0447 |
| Dense U-Net | 56.13% | 45.12% | 32.30% | 57.94% | 5.52% | 32.30% | 0.0942 | 0.0494 |
| $C_8$-Equivariant U-Net | 45.23% | 38.76% | 31.21% | 46.31% | 4.31% | 31.21% | 0.0757 | 0.0393 |
| $C_8$-Equivariant Dense U-Net | 38.79% | 37.91% | 36.89% | 38.93% | 4.29% | 36.89% | 0.0768 | 0.0399 |

Table B.2: Performance metrics detailed in §2.3.1 for the U-Net, dense U-Net, $C_8$-equivariant U-Net, and $C_8$-equivariant dense U-Net evaluated on the modified Massachusetts Roads Dataset using softIOU loss, evaluated with a 4 px buffer at road boundaries.

| **Model** | **Acc.** | **Bal. Acc.** | **Sens.** | **Spec.** | **Prec.** | **Recall** | $\mathbf{F}_1$ | **IoU** |
|---|---|---|---|---|---|---|---|---|
| U-Net | 54.52% | 70.19% | 100.00% | 40.37% | 34.29% | 100.00% | 0.5107 | 0.3429 |
| Dense U-Net | 76.37% | 85.60% | 99.92% | 71.28% | 42.90% | 99.92% | 0.6003 | 0.4289 |
| $C_8$-Equivariant U-Net | 65.15% | 78.95% | 100.00% | 57.90% | 33.07% | 100.00% | 0.4970 | 0.3307 |
| $C_8$-Equivariant Dense U-Net | 57.42% | 72.73% | 96.67% | 48.79% | 29.33% | 96.67% | 0.4500 | 0.2904 |

# Bibliography

[1] Engaging cluster documentation, 2023.

[2] Abolfazl Abdollahi, Biswajeet Pradhan, Gaurav Sharma, Khairul Nizam Abdul Maulud, and Abdullah Alamri. Improving road semantic segmentation using generative adversarial network. *IEEE Access*, 9:64381–64392, 2021.

[3] Tri Dev Acharya and Intae Yang. Exploring landsat 8. *International Journal of IT, Engineering and Applied Sciences Research (IJIEASR)*, 4(4):4–10, 2015.

[4] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 39(12):2481–2495, 2017.

[5] Wele Gedara Chaminda Bandara, Jeya Maria Jose Valanarasu, and Vishal M Patel. Spin road mapper: Extracting roads from aerial images via spatial and interaction space graph reasoning for autonomous driving. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 343–350. IEEE, 2022.

[6] EC Barrett and Colin K Grant. The identification of cloud types in landsat mss images. Technical report, 1976.

[7] Anil Batra, Suriya Singh, Guan Pang, Saikat Basu, CV Jawahar, and Manohar Paluri. Improved road connectivity by joint learning of orientation and segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10385–10393, 2019.

[8] Wesley Berg, Shannon T Brown, Boon H Lim, Steven C Reising, Yuriy Goncharenko, Christian D Kummerow, Todd C Gaier, and Sharmila Padmanabhan. Calibration and validation of the tempest-d cubesat radiometer. *IEEE Transactions on Geoscience and Remote Sensing*, 59(6):4904–4914, 2020.

[9] Sijing Cai, Yunxian Tian, Harvey Lui, Haishan Zeng, Yi Wu, and Guannan Chen. Dense-unet: a novel multiphoton in vivo cellular image segmentation model based on a convolutional neural network. *Quantitative imaging in medicine and surgery*, 10(6):1275, 2020.

[10] Gabriele Cesa, Leon Lang, and Maurice Weiler. A program to build e (n)-equivariant steerable cnns. In *International Conference on Learning Representations*, 2021.

[11] Abhishek Chaurasia and Eugenio Culurciello. Linknet: Exploiting encoder representations for efficient semantic segmentation. In *2017 IEEE Visual Communications and Image Processing (VCIP)*, pages 1–4. IEEE, 2017.

[12] Taco Cohen and Max Welling. Group equivariant convolutional networks. In *International conference on machine learning*, pages 2990–2999. PMLR, 2016.

[13] Angela Beth Crews. *Calibration and Validation for CubeSat Microwave Radiometers*. PhD thesis, Massachusetts Institute of Technology, 2019.

[14] Ilke Demir, Krzysztof Koperski, David Lindenbaum, Guan Pang, Jing Huang, Saikat Basu, Forest Hughes, Devis Tuia, and Ramesh Raskar. Deepglobe 2018: A challenge to parse the earth through satellite images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 172–181, 2018.

[15] Richard A Frey, Steven A Ackerman, Robert E Holz, Steven Dutcher, and Zach Griffith. The continuity modis-viirs cloud mask. *Remote Sensing*, 12(20):3334, 2020.

[16] Amelia Gagnon, Samantha Hasler, Juliana Chew, William Blackwell, Vince Leslie, and Kerri Cahoy. Data validation of the nasa time-resolved observations of precipitation structure and storm intensity with a constellation of smallsats (tropics) pathfinder microwave radiometer. 2022.

[17] Simon Graham, David Epstein, and Nasir Rajpoot. Dense steerable filter cnns for exploiting rotational symmetry in histology images. *IEEE Transactions on Medical Imaging*, 39(12):4124–4136, 2020.

[18] Steven Guan, Amir A Khan, Siddhartha Sikdar, and Parag V Chitnis. Fully dense unet for 2-d sparse photoacoustic tomography artifact removal. *IEEE journal of biomedical and health informatics*, 24(2):568–576, 2019.

[19] Jiaming Han, Jian Ding, Nan Xue, and Gui-Song Xia. Redet: A rotation-equivariant detector for aerial object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2786–2795, June 2021.

[20] Zhimeng Han, Muwei Jian, and Gai-Ge Wang. Convunext: An efficient convolution neural network for medical image segmentation. *Knowledge-Based Systems*, 253:109512, 2022.

[21] Tong He, Zhi Zhang, Hang Zhang, Zhongyue Zhang, Junyuan Xie, and Mu Li. Bag of tricks for image classification with convolutional neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 558–567, 2019.

[22] Andrew Heidinger and William C. Straka III. Abi cloud mask version 3.0, 2012.

[23] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017.

[24] Yifeng Huang, Zhirong Tang, Dan Chen, Kaixiong Su, and Chengbin Chen. Batching soft iou for training semantic segmentation networks. *IEEE Signal Processing Letters*, 27:66–70, 2019.

[25] M Joseph Hughes and Robert Kennedy. High-quality cloud masking of landsat 8 imagery using convolutional neural networks. *Remote Sensing*, 11(21):2591, 2019.

[26] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. PMLR, 2015.

[27] Shreeyam Kacker, Alex Meredith, Kerri Cahoy, and Georges Labreche. Machine learning image processing algorithms onboard ops-sat. 2022.

[28] Shreeyam Kacker, Alex Meredith, Joe Kusters, Hannah Tomio, Violet Felt, and Kerri Cahoy. On-orbit rule-based and deep learning image segmentation strategies. In *AIAA SCITECH 2022 Forum*, page 0646, 2022.

[29] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[30] Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein. Visualizing the loss landscape of neural nets. *Advances in neural information processing systems*, 31, 2018.

[31] Xiaoya Li, Xiaofei Sun, Yuxian Meng, Junjun Liang, Fei Wu, and Jiwei Li. Dice loss for data-imbalanced nlp tasks. *arXiv preprint arXiv:1911.02855*, 2019.

[32] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017.

[33] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11976–11986, 2022.

[34] Gilles Louppe. Understanding random forests: From theory to practice. *arXiv preprint arXiv:1407.7502*, 2014.

[35] Sebastián Martinuzzi, William A Gould, and Olga M Ramos González. Creating cloud-free landsat etm+ data sets in tropical landscapes: cloud and cloud-shadow removal. *US Department of Agriculture, Forest Service, International Institute of Tropical Forestry. Gen. Tech. Rep. IITF-32.*, 32, 2007.

[36] Patrick Minnis, Szedung Sun-Mack, Yan Chen, Fu-Lung Chang, Christopher R Yost, William L Smith, Patrick W Heck, Robert F Arduini, Sarah T Bedka, Yuhong Yi, et al. Ceres modis cloud product retrievals for edition 4—part i: Algorithm changes. *IEEE Transactions on Geoscience and Remote Sensing*, 59(4):2744–2780, 2020.

[37] Joshua Mitton and Roderick Murray-Smith. Rotation equivariant deforestation segmentation and driver classification. *NeurIPS 2021 Workshop on Tackling Climate Change with Machine Learning*, 2021.

[38] Volodymyr Mnih. *Machine learning for aerial image labeling.* PhD thesis, 2013.

[39] Volodymyr Mnih and Geoffrey E Hinton. Learning to label aerial images from noisy data. In *Proceedings of the 29th International conference on machine learning (ICML-12)*, pages 567–574, 2012.

[40] Alejandro Newell, Kaiyu Yang, and Jia Deng. Stacked hourglass networks for human pose estimation. In *European conference on computer vision*, pages 483–499. Springer, 2016.

[41] Victor Francisco Rodriguez-Galiano, Bardan Ghimire, John Rogan, Mario Chica-Olmo, and Juan Pedro Rigol-Sanchez. An assessment of the effectiveness of a random forest classifier for land-cover classification. *ISPRS journal of photogrammetry and remote sensing*, 67:93–104, 2012.

[42] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[43] R Tapakis and AG Charalambides. Equipment and methodologies for cloud detection and classification: A review. *Solar Energy*, 95:392–430, 2013.

[44] Qing Z Trepte, Patrick Minnis, Szedung Sun-Mack, Christopher R Yost, Yan Chen, Zhonghai Jin, Gang Hong, Fu-Lung Chang, William L Smith, Kristopher M Bedka, et al. Global cloud detection for ceres edition 4 using terra and aqua modis data. *IEEE Transactions on Geoscience and Remote Sensing*, 57(11):9410–9449, 2019.

[45] Lachlan Tychsen-Smith and Lars Petersson. Improving object localization with fitness nms and bounded iou loss. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6877–6885, 2018.

[46] Adam Van Etten. Spacenet road detection and routing challenge — part i, 2017.

[47] Petro Vorotyntsev, Yuri Gordienko, Oleg Alienin, Oleksandr Rokovyi, and Sergii Stirenko. Satellite image segmentation using deep learning for deforestation detection. In *2021 IEEE 3rd Ukraine Conference on Electrical and Computer Engineering (UKRCON)*, pages 226–231. IEEE, 2021.

[48] Kiri L Wagstaff, Alphan Altinok, Steve A Chien, Umaa Rebbapragada, Steve R Schaffer, David R Thompson, and Daniel Q Tran. Cloud filtering and novelty detection using onboard machine learning for the eo-1 spacecraft. In *Proc. IJCAI Workshop AI in the Oceans and Space*, 2017.

[49] Stephen G Warren. Optical properties of ice and snow. *Philosophical Transactions of the Royal Society A*, 377(2146):20180161, 2019.

[50] Maurice Weiler and Gabriele Cesa. General e (2)-equivariant steerable cnns. *Advances in Neural Information Processing Systems*, 32, 2019.

[51] Maurice Weiler, Fred A Hamprecht, and Martin Storath. Learning steerable filters for rotation equivariant cnns. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 849–858, 2018.

[52] Zar Chi Win, Nobuya Mizoue, Tetsuji Ota, Guangyu Wang, John L Innes, Tsuyoshi Kajisa, and Shigejiro Yoshida. Spatial and temporal patterns of illegal logging in selectively logged production forest: A case study in yedashe, myanmar. *Journal of Forest Planning*, 23(2):15–25, 2018.

[53] Richard Zhang. Making convolutional networks shift-invariant again. In *International conference on machine learning*, pages 7324–7334. PMLR, 2019.

[54] Jun Zhou, Hu Yang, and Kent Anderson. Snpp atms on-orbit geolocation error evaluation and correction algorithm. *IEEE Transactions on Geoscience and Remote Sensing*, 57(6):3802–3812, 2019.

[55] Zhe Zhu, Shixiong Wang, and Curtis E Woodcock. 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, 2015.

[56] Anze Zupanc. Improving cloud detection with machine learning, 2017.