

# Sensor Independent Deep Learning for Detection Tasks with Optical Satellites

A thesis submitted for the degree of Doctor of Philosophy

By

Alistair M. Francis

University College London

August 2021

---

*I, Alistair M. Francis, confirm that the work presented in this thesis is my own.  
Where information has been derived from other sources, I confirm that this has  
been indicated in the thesis.*



# Abstract

The design of optical satellite sensors varies widely, and this variety is mirrored in the data they produce. Deep learning has become a popular method for automating tasks in remote sensing, but currently it is ill-equipped to deal with this diversity of satellite data. In this work, sensor independent deep learning models are proposed, which are able to ingest data from multiple satellites without retraining. This strategy is applied to two tasks in remote sensing: cloud masking and crater detection. For cloud masking, a new dataset—the largest ever to date with respect to the number of scenes—is created for Sentinel-2. Combination of this with other datasets from the Landsat missions results in a state-of-the-art deep learning model, capable of masking clouds on a wide array of satellites, including ones it was not trained on. For small crater detection on Mars, a dataset is also produced, and state-of-the-art deep learning approaches are compared. By combining datasets from sensors with different resolutions, a highly accurate sensor independent model is trained. This is used to produce the largest ever database of crater detections for any solar system body, comprising 5.5 million craters across Isidis Planitia, Mars using CTX imagery. Novel geospatial statistical techniques are used to explore this database of small craters, finding evidence for large populations of distant secondary impacts. Across these problems, sensor independence is shown to offer unique benefits, both regarding model performance and scientific outcomes, and in the future can aid in many problems relating to data fusion, time series analysis, and on-board applications. Further work on a wider range of problems is needed to determine the generalisability of the proposed strategies for sensor independence, and extension from optical sensors to other kinds of remote sensing instruments could expand the possible applications of this new technique.

---

# Impact Statement

The research in this thesis presents significant benefits, both inside and outside academia. Several parts of this work have been disseminated in peer reviewed material. CloudFCN, a cloud masking model designed for Landsat 8 [1]. The dataset of sub-km craters on Mars [2]. The dataset of Sentinel-2 cloud masks [3]. There is also a paper currently under peer review dealing with sensor independent cloud masking. Additionally, work has been communicated in various conference presentations, including, among others, the European Geophysical Union, the European Planetary Science Conference, and ESA’s Phi-Week.

The datasets developed in Chapters 4 and 7 are both publicly available for download. These datasets will allow researchers to develop their own cloud masking and crater detection models, providing the data needed to train and test them. Similarly, much of the code used in the research is publicly available under permissive licenses, and allows others to use, extend or adapt the software developed for the thesis. In particular, the Sentinel-2 cloud masking catalogue has already had over 2000 unique downloads to date [3].

The dataset of sub-km Martian craters also provided an opportunity for outreach work with 16-18 year old students at a local school. The project included over 10 hours of seminars on Martian surface science and machine learning, before the hands-on work creating the dataset. This gave the 16 students the chance to see how real research was conducted, and take part themselves. They are now co-authors on a peer reviewed article [2], which will benefit them immensely if they choose to go onto further study.

Sensor independent deep learning is an important advance for industry and commercial applications. The labour and resources required to develop deep learning models for different sensors is substantial. By demonstrating the utility of sensor independence, companies and institutions may be able to reduce the amount of time spent on data collection and labelling, so that they can focus more on the deployment of their products and services.

The cloud masking models designed in this thesis are immediately applicable, and have already been deployed as part of a high-resolution albedo retrieval system in ESA Contract No. 4000130413/20/I-DT. Given that the measured performance of the model exceeds those currently used as Sentinel-2 and Landsat 8 cloud masks, wider uptake of the cloud mask developed here would improve the reliability of predicted clouds considerably.

---

# Acknowledgements

I would like to express my deepest gratitude to my parents, who have been nothing but supportive throughout the PhD. When I moved back home during the pandemic, my parents offered me their space, their energy bills, and—perhaps most significantly—their food, to keep me going. As well as my parents, I must thank my brother, Jamie, and his partner, Kat, who have given me interesting conversations and much-needed coffee, as well as their beautiful dog, Zuma, who brings joy (and fur and saliva) to all who meet him.

I would also like to thank my friends, Jacqueline and Ollie, for reading and commenting on the thesis, and for our countless daily messages and quiz nights throughout successive lockdowns.

My thanks to those at the Phi-Lab, and ESRIN, who made my visit there so enjoyable and productive, including Pierre-Philippe Mathieu who facilitated my visit, and introduced me to such a brilliant environment for organic collaboration. Special thanks must go to John Mrziglod, for his continued and vital help on many parts of this work, not least his annotation of several hundred images, and his help with some preliminary explorations of sensor independence. And also to Jenny, for showing me almost all of the bars that Rome had to offer.

My thanks to the 16 students who helped annotate Martian craters: Jonathan, Thomas, Romilly, Jennifer, Matthew, Jasmine, Viran, Arianne, Oliver, Aaron, Damien, Meg, Wiggert, Alice and especially Reuben, who then provided further assistance in research relating to Isidis Planitia, for which I am very grateful. As well as the students, the teachers Matt Horncastle and James Waller were a brilliant help when organising and running the project, and my warmest thanks go to them and the whole of the College of Richard Collyer, Horsham. Such a project would not have happened were it not for the time and energy that Will Dunn and the rest of the ORBYTS team put into running such a fantastic scheme.

Finally, I wish to thank my supervisors. Jan-Peter Muller, whose guidance, advice, ideas, and most of all his enthusiasm for science and research, are exemplary, and made him a pleasure to work alongside, and Panos Sidiropoulos, for his technical wisdom, his sound research instincts, and the many evening calls spent discussing the finer points of data science.

---

# Contents

<b>1</b>	<b>Introduction</b>	<b>21</b>
1.1	Motivation . . . . .	21
1.2	Contributions . . . . .	22
1.3	A Wider Context . . . . .	25
1.4	Thesis Outline . . . . .	34
<b>2</b>	<b>Literature Review</b>	<b>37</b>
2.1	Remote Sensing Instruments . . . . .	37
2.2	Machine Learning . . . . .	41
2.3	Cloud Masking . . . . .	60
2.4	Craters on Mars . . . . .	65
2.5	Isidis Planitia . . . . .	66
2.6	Crater Detection . . . . .	67
<b>3</b>	<b>CloudFCN</b>	<b>69</b>
3.1	Motivation . . . . .	69
3.2	Methods . . . . .	70
3.3	Theoretical Considerations . . . . .	76
3.4	Experimental Results . . . . .	79
3.5	Discussion . . . . .	88
3.6	Interim Conclusions . . . . .	89
<b>4</b>	<b>Cloud Masking Dataset</b>	<b>91</b>
4.1	Motivation . . . . .	91
4.2	IRIS Annotation Tool . . . . .	93
4.3	Dataset Creation . . . . .	99
4.4	Dataset Description . . . . .	101
4.5	Dataset Statistics . . . . .	103
4.6	Validation and Uncertainties . . . . .	105
4.7	Interim Conclusions . . . . .	107

<b>5</b>	<b>SEnSel</b>	<b>109</b>
5.1	Motivation . . . . .	109
5.2	Methods . . . . .	110
5.3	Experimental Results . . . . .	119
5.4	Discussion . . . . .	131
5.5	Interim Conclusions . . . . .	136
<b>6</b>	<b>Crater Detection: Initial Study</b>	<b>139</b>
6.1	Motivation . . . . .	139
6.2	Data . . . . .	140
6.3	Classifying Craters . . . . .	142
6.4	Detecting Craters . . . . .	145
6.5	Improvements . . . . .	150
6.6	Experimental Results . . . . .	154
6.7	Interim Conclusions . . . . .	158
<b>7</b>	<b>ORBYTS Crater Dataset</b>	<b>161</b>
7.1	Motivation . . . . .	161
7.2	Methods . . . . .	162
7.3	Data Description . . . . .	168
7.4	Interim Conclusions . . . . .	171
<b>8</b>	<b>Crater Detection Revisited</b>	<b>173</b>
8.1	Motivation . . . . .	173
8.2	Datasets . . . . .	174
8.3	Models . . . . .	177
8.4	Results . . . . .	180
8.5	Interim Conclusions . . . . .	184
<b>9</b>	<b>Cratering over Isidis Planitia</b>	<b>185</b>
9.1	Motivation . . . . .	185
9.2	Crater Detection over Isidis . . . . .	187
9.3	Data Validation . . . . .	189
9.4	Statistical Analysis . . . . .	194
9.5	Interim Conclusions . . . . .	208



<b>10 Discussion</b>	<b>213</b>
10.1 Data . . . . .	214
10.2 Model Selection . . . . .	215
10.3 Science and Technology . . . . .	216
<b>11 Conclusions</b>	<b>217</b>
11.1 Findings & Contributions . . . . .	217
11.2 Future Work . . . . .	219
<b>A Computational Efficiency</b>	<b>243</b>
<b>B Detailed Results on Sentinel-2</b>	<b>245</b>
<b>C Concerning YOLOv5</b>	<b>247</b>



# List of Figures

1.1	Example of clouds in Sentinel-2 . . . . .	24
1.2	Example of craters on Mars . . . . .	24
1.3	Automation and data rate . . . . .	27
1.4	Prerequisites for value . . . . .	30
2.1	Schematic of Sentinel-2 platform . . . . .	38
2.2	Precision-Recall curve . . . . .	57
2.3	Example of decision tree . . . . .	59
3.1	Flowchart of CNN used in cloud segmentation . . . . .	72
3.2	Two examples of data from the Carbonite-2 satellite . . . . .	81
3.3	Examples of cloud masks on the Landsat 8 dataset . . . . .	85
3.4	Effects of white noise and quantisation . . . . .	87
3.5	Performance of CloudFCN against noise . . . . .	87
4.1	Configuration pane for model in IRIS . . . . .	94
4.2	User interface in the IRIS software . . . . .	97
4.3	Google Form used to add classification labels to dataset . . . . .	100
4.4	Class distribution per pixel . . . . .	104
4.5	Distribution of difficulty ratings . . . . .	104
4.6	Classification tag percentages . . . . .	104
4.7	Calibration set statistics . . . . .	106
4.8	Validation set statistics . . . . .	106
5.1	The bands of several multispectral instruments . . . . .	112
5.2	Schematic of SEnSel . . . . .	113
5.3	Flowchart for unsupervised autoencoder training of SEnSel . . . . .	116
5.4	Predictions on Sentinel-2 test set . . . . .	123
5.5	Predictions on Landsat 8 test set . . . . .	127
5.6	Predictions on Landsat 7 test set . . . . .	129
5.7	Predictions on Sentinel-3 SLSTR scene . . . . .	132
6.1	HRSC dataset over Nanedi Valles . . . . .	140
6.2	CSFD of dataset . . . . .	141
6.3	Flowchart of CNN used for crater classification . . . . .	144
6.4	Full CDA pipeline . . . . .	147
6.5	Schematic of localisation network output geometry . . . . .	147

6.6	Behaviour of CDA at different fields of view . . . . .	150
6.7	Localisation errors due to multiple craters . . . . .	151
6.8	Illustration of rationale for hard negatives . . . . .	152
6.9	Precision and recall before and after hard negative training . . . . .	153
6.10	IoU of bounding boxes before and after hard negatives . . . . .	155
6.11	Crater predictions over image 3_25 . . . . .	157
7.1	Visualisation of IoU . . . . .	163
7.2	Annotations over image C . . . . .	164
7.3	CSFDs of dataset . . . . .	166
7.4	MC-11 East and the ORBYTS dataset sites . . . . .	170
8.1	CTX test data over Isidis Planitia . . . . .	175
8.2	YOLOv5 predictions . . . . .	181
8.3	Mask R-CNN predictions . . . . .	182
9.1	CTX Mosaic grid and Isidis Planitia . . . . .	188
9.2	Example of a very eroded crater on Isidis . . . . .	191
9.3	CSFDs for models over Isidis . . . . .	191
9.4	Isochron fitting comparison . . . . .	192
9.5	Heatmap of crater detections over Isidis . . . . .	193
9.6	Visualisation of annuli around large craters . . . . .	195
9.7	Density deviation against annular distance . . . . .	196
9.8	Visualisation of NSNs . . . . .	198
9.9	Ratio of $R_{NSN}$ against $R_{NSN}^{expected}$ . . . . .	199
9.10	Geometry of NSN phase angle . . . . .	201
9.11	Distribution of NSN phase angles . . . . .	202
9.12	Schematic for calculation of peak angles . . . . .	203
9.13	Distribution of NSN phase angles per quadrant for $D < 50m$ . . . . .	205
9.14	Distribution of NSN phase angles per quadrant for $D > 50m$ . . . . .	205
9.15	Map of Mars with extrapolated orbits from quadrants . . . . .	207
9.16	Geometry of relative NSN angle . . . . .	208
9.17	Relative NSN angle against annular distance . . . . .	209

# List of Tables

2.1	Spectral bands of Sentinel-2 . . . . .	38
2.2	Spectral bands of Landsat 8 . . . . .	40
3.1	Receptive field of CloudFCN layers . . . . .	78
3.2	Full results of the Carbonite-2 experiments . . . . .	82
3.3	Cloud detection results for Landsat 8 CCA dataset . . . . .	84
4.1	Publicly available cloud masking datasets. . . . .	92
4.2	Views defined in IRIS . . . . .	96
5.1	Results on Sentinel-2 dataset . . . . .	122
5.2	Results on Landsat 8 dataset . . . . .	126
5.3	Results on Landsat 7 dataset . . . . .	128
5.4	Results on CloudPeru2 dataset . . . . .	131
6.1	$F_1$ scores for binary classification of craters . . . . .	145
6.2	Performance of CDA on HRSC dataset . . . . .	156
6.3	Times taken for CDA to process an image . . . . .	158
7.1	Scene-wise analysis of agglomerative clustering . . . . .	167
7.2	Positional accuracy of annotations . . . . .	167
7.3	Comparison with expert annotations . . . . .	167
7.4	Scene-wise information about dataset . . . . .	169
8.1	Hyperparameters varied in evolution of YOLOv5 . . . . .	178
8.2	Performance of YOLOv5 on test set . . . . .	183
8.3	Performance of Mask R-CNN on test set . . . . .	183
8.4	Computation time for crater detection models . . . . .	183
9.1	Crossing angles for the three observed peaks . . . . .	204
9.2	Cross angles for per quadrant . . . . .	207
10.1	Overview of differences in methodology, data, and outcomes between the two projects. . . . .	213
A.1	Computational efficiency of SEnSel . . . . .	243
B.1	Detailed results over Sentinel-2 test set . . . . .	246



# Glossary

**AP** Average Precision

**BA** Balanced Accuracy

**BT** Brightness Temperature

**CCA** Cloud Cover Assessment

**CCD** Charge Coupled Device

**CDA** Crater Detection Algorithm

**CloudFCN** Cloud Fully Convolutional Network

**COCO** Common Objects in Context

**CPU** Central Processing Unit

**CSV** Comma Separated Values

**CONIDA** National Commission for Aerospace Research and Development

**CSFD** Crater Size-Frequency Distribution

**CNN** Convolutional Neural Network

**CTX** Context Camera

**ESA** European Space Agency

**ETM+** Enhanced Thematic Mapper Plus

**FCN** Fully Convolutional Network

**GML** Geography Markup Language

**GPU** Graphics Processing Unit

**HiRISE** High Resolution Imaging Science Experiment

**HRSC** High Resolution Stereo Camera

**IoU** Intersection over Union

**IPCC** Intergovernmental Panel on Climate Change

**IRIS** Intelligent Reinforcement for Image Segmentation

**LHB** Late Heavy Bombardment

**mAP** Mean Average Precision

**mAP@50%** Mean Average Precision at 50% IoU

**MRO** Mars Reconnaissance Orbiter

**MSE** Mean Square Error

**MSI** MultiSpectral Imager

**NASA** National Aeronautics and Space Administration

**NDSI** Normalised Difference Snow Index

**NDVI** Normalised Difference Vegetation Index

**NDWI** Normalised Difference Water Index

**NSN** Nearest Similar Neighbour

**NIR** Near Infrared

**OA** Overall Accuracy

**OLCI** Ocean and Land Colour Instrument

**OLI** Operational Land Imager

**ORBYTS** Original Research By Young Twinkle Students

**PASCAL VOC** Pattern Analysis, Statistical Modelling and Computational Learning -  
Visual Object Categories

**PR Curve** Precision-Recall Curve

**R-CNN** Region-based Convolutional Neural Network

**ReLU** Rectified Linear Unit

**RF** Random Forest

**RGB** Red Green Blue

**RMSE** Root Mean Square Error

**SEnSeI** Spectral Encoder for Sensor Independence

**SGD** Stochastic Gradient Descent

**SNAP** Sentinel Application Platform

**SNR** Signal-to-Noise Ratio

**SPARCS** Spatial Procedures for Automated Removal of Cloud and Shadow

**SLSTR** Sea and Land Surface Temperature Radiometer



**SVM** Support Vector Machine

**SWIR** Shortwave Infrared

**TES** Thermal Emission Spectrometer

**THEMIS** Thermal Emission Imaging System

**TIR** Thermal Infrared

**TIRS** Thermal Infrared Sensor

**USGS** United States Geological Survey

**YOLO** You Only Look Once



# 1 | Introduction

## 1.1 Motivation

As technology has progressed and the number of satellites orbiting Earth and other planetary bodies has grown, the sheer volume of image data being captured has climbed dramatically. In contrast to the few precious film cartridges used by the *Apollo 8* astronauts capturing the celebrated *Earthrise* images, the scale of our data collection activities is now staggering. For example, the Sentinel-2A and 2B satellites together create around 1.7 TB of downlinked data per day [4]. The volume of data we are now in receipt of is perhaps the chief motivator for the automation of data processing, both in satellite operations, and in downstream science studies.

Alongside this massive expansion in data quantity, computer science research has evolved rapidly to meet the extra demands placed on computing systems. In recent years, since AlexNet was shown to give state-of-the-art performance as an image classifier in 2012 [5], deep learning has come to dominate image analysis in many fields. A key advantage of deep learning over other methods is its ability to learn highly non-linear functions across hugely varied input spaces, by efficiently leveraging the information held in massive amounts of training examples.

Optical satellite imagery, at first glance, appears to be one of those domains ideal for deep learning. Indeed, as this thesis and the work of many others shows, deep learning has the potential to revolutionise many aspects of research in the field. However, an assumption that the deep learning methods developed for different applications can be straightforwardly translated into the field of remote sensing is naïve. The diversity of satellite sensors, many with bespoke designs and cutting-edge hardware, acts as an obstacle in the translation of deep learning into remote sensing work. A central (but rarely explicitly considered) pillar of deep learning’s flexibility—that we can use the same model across images from different camera systems, without any concern as to whether it came from a *Panasonic* camera, or an *iPhone*—is not available to remote sensing.

The impact of this peculiarity of remote sensing may at first seem mundane: we can’t use the model on multiple satellites, but we can just train models on the individual sensors instead. However, doing so diminishes the most powerful advantage deep learning has, namely, its ability to learn useful things from truly vast quantities of training data.

Of course, labelled training data can still be created, but it is not only specific to the problem, it is also specific to the sensor. Over the years, even as more and more labelled training data is generated, the effective quantity of training data for any given application will not grow, because we will be starting from scratch for each new sensor. As well as the cumulative effort in labelling data, there is also a huge amount of duplicated effort in developing very similar models for different sensors, and testing them separately. Deep learning techniques are also often delayed when it comes to a new satellite; whilst other methods can be designed before the satellite is even launched, work on deep learning approaches must wait until data is downlinked, and then made into labelled training data. Perhaps for this reason, very few deep learning models have been deployed by ground stations in satellite data processing chains, because automated processing routines must be operational from very soon after launch.

What is needed, then, is a way to break through this impasse so that remote sensing research can fully benefit from deep learning. I define *sensor independence*<sup>1</sup> as the property of a model to simultaneously work across a variety of sensors, even if those sensors are not similar with respect to resolution or measured wavelengths. Sensor independence would allow for the seamless combination of training sets, in order to realise the value in all available labelled datasets. Additionally, sensor independent models are much more easily extended to new satellites, making them ideal candidates for on-board AI deployment, or satellite downlinking and processing pipelines, for which labelled training data cannot be collected prior to launch.

## 1.2 Contributions

The aim of this thesis is to demonstrate the value of sensor independence by using it as the basis for designing, training, testing and applying models in remote sensing. I show that these models work simultaneously across multiple satellites, without any retraining or adjustment. Further, I created sensor independent deep learning models which surpass standard deep learning models, by virtue of their access to larger quantities of training data than would otherwise be possible.

The concept of sensor independence is not unique to any one problem in remote sensing, and can be applied very widely. To show its worth it was important to present a diverse set of examples of its use. However, the technical work required to create labelled

---

<sup>1</sup>*Interoperability* is determined by the satellites, describing when they are similar enough to be used by the same algorithm or model. I propose a subtly different term, *sensor independence*, which places the burden of compatibility not on the satellite sensors, but on the model, working across sensors which are not necessarily similar.

datasets, develop and optimise models, fully analyse their performance characteristics, and then use them to generate new science, must be repeated for each problem. Therefore, rather than doing many small, limited case studies, I chose two well-known problems which I explored deeply, but which still exemplify as wide a range as possible of deep learning tasks with optical satellite imagery. Namely, cloud masking in multispectral imagery on Earth and crater detection in panchromatic imagery on Mars.

Deep learning is a somewhat heuristic pursuit. Therefore, making general claims as to the efficacy of a given strategy like sensor independence for all problems is difficult. Instead, by exploring the commonalities and differences which emerge between my work on cloud masking and crater detection, I aim to draw out some general guidelines for employing sensor independent deep learning with optical satellite imagery.

Within both of the individual case studies, significant advances are also made. In cloud masking (see Figure 1.1), I have designed and tested a novel model design, Cloud-FCN (Cloud Fully Convolutional Network), exhibiting excellent performance on Landsat 8 using existing datasets. Extending this to Sentinel-2 led to the creation of a new dataset (containing the most images of any satellite cloud masking dataset) which was labelled using a new tool, Intelligent Reinforcement for Image Segmentation (IRIS), the design of which I contributed to. The final major contribution of the cloud masking work was Spectral Encoder for Sensor Independence (SEnSeI). This model can be used to make existing deep learning models sensor independent, and led to a model which had high performance across five sensors, having only been trained on two of them, showing the power of sensor independent techniques.

The second application is the automated detection of craters on Mars (see Figure 1.2). Initially, I designed my own model for crater detection, but unfortunately several problems arose which made it unsuitable for further use. With the lessons learnt from this exercise, however, a new dataset was created with the help of 16 young volunteers as part of the Original Research By Young Twinkle Scientists (ORBYTS) scheme. With this dataset and several pre-existing ones, existing object detection frameworks were tweaked and trained, including Mask Region-based Convolutional Neural Network (R-CNN) and You Only Look Once v5 (YOLOv5). By pooling these datasets to make sensor independent models, performance was shown to increase substantially. The high performance crater detection model was then used to detect 5.5 million small craters across Isidis Planitia, Mars (the largest catalogue of crater detections on any planetary body to date), and novel geospatial statistical techniques were used to analyse the effects of secondary cratering on the crater population.

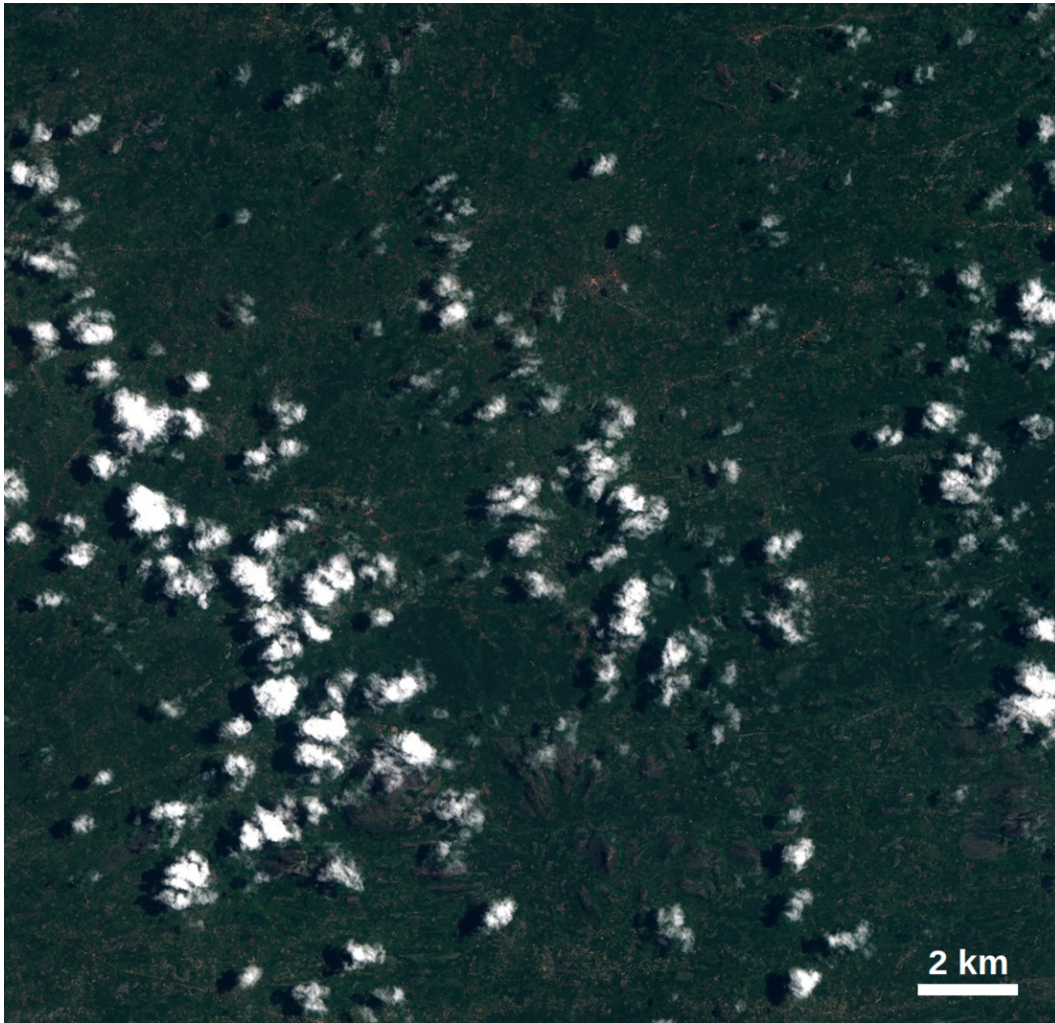


Figure 1.1: Example of clouds over a forested region of Mozambique to illustrate need for cloud masking. This image comes from the Sentinel-2 satellite and is part of the dataset described in Chapter 4.



Figure 1.2: High resolution image from the HiRISE instrument over a region of Hesperia Planum. A huge number of small craters such as those shown here exist over Mars' surface. Credit: NASA/JPL/UAritona.

## 1.3 A Wider Context

In this section I will give a broad narrative for why sensor independence is an important step in deep learning research within remote sensing, beyond the more straightforward contributions offered in the previous section. My argument is summarised as follows: deep learning applications can be helpfully split into either technological or scientific ones, based on the kind of potential value they offer us, and the current uptake of solutions (Section 1.3.1). Chasing performance when science is the goal can in fact lead to a reduction in the value of research, because prioritising raw performance and model inter-comparisons means sacrifices are made with respect to real-world usability (Section 1.3.2). Remote sensing is particularly at risk from this, because of the specific structure of its data (Section 1.3.3). The pursuit of sensor independence targets this issue, by offering a way to maintain or improve performance whilst also *increasing* real-world usability, rather than decreasing it (Section 1.3.4).

### 1.3.1 Science & Technology

There are a set of computer vision problems for which—in terms of numerical accuracy—“superhuman” ability has been achieved (e.g. classification of the ImageNet dataset [6] and breast cancer screening [7]). However, for the majority of real-world image processing tasks, where data quality is limited and the solution is inherently ambiguous, even state-of-the-art deep learning models fail to meet human-like capabilities [8]. Despite a lack of evidence that new techniques (e.g. deep learning) are greatly surpassing human-level performance in most computer vision tasks, they do indeed offer capabilities that exceed previous automation techniques. Therefore, researchers have rightly continued to apply deep learning to existing problems, but with mostly incremental results over other solutions, and only rarely approaching “human-level” performance.

This fact, that deep learning is not actually revolutionising scientific pursuits, and that we are not undergoing a paradigm shift in our ability to understand the world around us, is sometimes at odds with the hype surrounding the technology. The overestimation of deep learning’s current capabilities is common amongst the public [9]. Discourse surrounding the recent successes of deep learning in the technology sector work to bolster the misconception that full, errorless automation of even highly complex, scientific tasks is not a question of ‘if’, but an impatient ‘when’.

For example, *Forbes* told its readers in a 2019 article that: “...humanity stands on the brink of a technology triggered information revolution” and that “...as the machine

learning algorithms improve rapidly, the power and promise of software that learn by example, patterns and models seems immense” [10]. Deep learning is certainly changing the digital world at a breakneck pace, but the assumption that this will be mirrored in science is based on the assertion that science is predominantly a data processing exercise, when in fact it is a creative, exploratory endeavour which is certainly aided by data analysis, but not defined by it.

Rather than assuming deep learning will always provide value to remote sensing, then, it is worthwhile to specify the kinds of tasks for which deep learning is well-suited. I argue that research into automation within remote sensing tends to produce value in two ways: *technological value* and *scientific value*. Technological value is incremental, in the sense that it is found by improving the quality of an already-existing system, or doing the same task but better in a novel way. Meanwhile, scientific value comes when automation unlocks a new discovery, by making it possible to analyse data in an altogether new way (of course, problems can exhibit a mix of both of these modes). To illustrate, consider the two case studies in this thesis.

Cloud masks already exist and are widely used (so incremental improvements to results are what is important), their primary purpose is to identify clouds as accurately as possible so that downstream applications can either avoid or focus on them. This is a chiefly technological pursuit—although it still improves science through its use. Meanwhile, small crater detection is only really fruitful when thought of in terms of direct scientific value, because it has never been done at scale: what do the model’s biases and errors tell us about the data and the terrain it covers? Can this completely new resource (a massive database of small craters) be used to answer a question that couldn’t be answered before? The quantitative performance of a model in this setting is less important than the qualitative opportunities for research it provides.

Other examples of problems that produce technological value include: image coregistration, surface albedo retrieval, digital elevation model creation, derived product calculation (e.g. the Normalised Difference Vegetation Index—NDVI), data quality flagging, and noise correction. All of these tasks have an important commonality, in that they are already relied upon for many applications, and are already widely automated (perhaps not with deep learning, but automated nonetheless). In other words, if we use *data rate* to denote the amount of data being processed with an automated tool, then the data rate of all these tasks is already approaching the maximum potential data rate, and is already much higher than would be possible to do manually.

Meanwhile, tasks that offer scientific value are characterised by a *low* current data rate, but one which has the potential to be increased. It could also be that the current



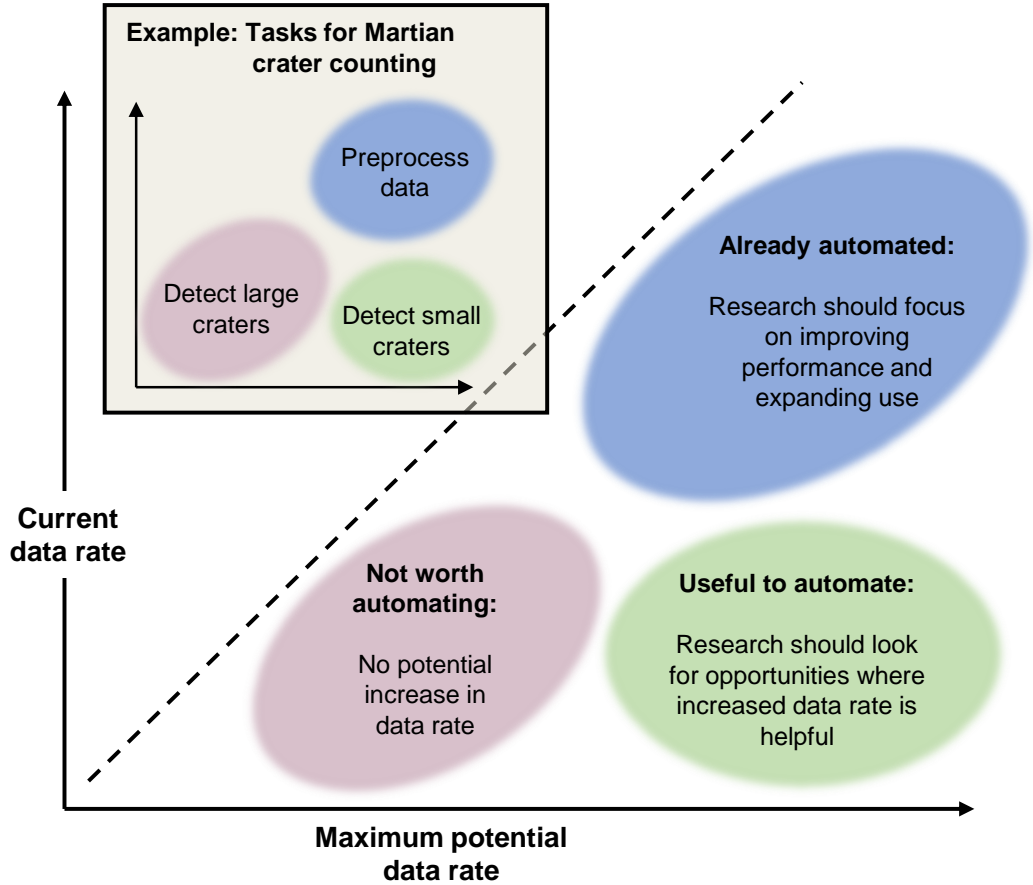


Figure 1.3: Graphical depiction of tasks for which automation is suitable. Technological value can be found in problems with high current and potential data rates. Scientific value is common for problems that we have not yet automated, but which have a high potential data rate. Finally, there are a class of problems for which automation is not useful, with low maximum potential data rates.

data rate is relatively high in specific situations, but not more broadly. Small crater cataloguing and boulder detection (on the Lunar and Martian surface), crop type segmentation on Earth, (high-resolution) change detection, and so on, are examples of such tasks. If anyone proposes a solution which works well for any of these problems over huge amounts of data, and actually applies it in a way that massively increases the current data rate, then they are presented with an entirely new avenue of data analysis, potentially leading to novel discoveries.

Figure 1.3 depicts where these categories of problem fall, when thought of in terms of data rate. By considering the landscape of possible applications of automation in this way, another category of task is made clear: those problems for which there is a low current data rate, but which also have a relatively low maximum data rate, even if they were fully automated. Large ( $\geq 1$  km diameter) crater detection on Mars is one such problem. Scientists have already comprehensively catalogued every crater above

around 1–2 km diameter across the whole of Mars, by hand [11]. There is no reason to automate this process, as it will neither enhance performance, nor unlock millions of previously undetected craters that we did not have time to mark manually. When we fail to consider whether a task is actually worth automating, we risk developing predictive models without a technological or scientific purpose, which will neither improve current workflows, nor answer new questions.

### 1.3.2 Data, Performance and Utility

So far, I have proposed that there are two different kinds of value that deep learning offers to remote sensing. This section explores what the pre-requisites are for realising that value, regardless of whether it is fundamentally technological or scientific. The presence of *data*, *performance* and *utility*, are all necessary for value. There must be enough data to train (assuming its a supervised technique) and test a model. A model must achieve a high enough accuracy (or meet some minimum standard) to give an acceptably small error population. And, the model must have utility, in the sense that it solves the problem that we desire to be solved (not some unrepresentative or simplified version of it), and does so in a practical, timely way. Unless an application has sufficient data, a high enough performance, and practical utility, then the work will not add value.

Deep learning applications in remote sensing fail to meet one or more of these criteria quite often. We are rarely short of data in and of itself, with masses of it downlinked by satellites all the time. But for many problems, labelled data is scarce, or unsuitable. If supervised approaches are used, this must be addressed before working on a solution. Even unsupervised techniques must be tested and shown to perform well against some measure, and so we need some kind of ground truth. As well as an adequate quantity of data, the quality of satellite imagery also affects an application’s success. This is limited by the satellites’ sensing capabilities, and the geometry and physical characteristics of the things being detected. Deep learning methods cannot extract information from a satellite image if that information does not exist in the first place. Spatial, temporal and spectral resolutions are all hard barriers which must be accounted for.

Performance is directly affected by all the aspects of data that were just outlined. Better and larger datasets will create models with higher performance, and that performance will be measured more accurately with trustworthy and representative test sets. Besides the impact of data quantity and quality, model design is the other main driver of model performance, with a plethora of architectures proposed for problems (covered more in Section 2.2). Pre-processing of the input data, or post-processing of the results

coming out of the model, can also add to performance. Less obviously, performance is also determined by the measure one chooses to use and report in testing. This can have significant consequences for the value of a model, where an inappropriate choice of metric(s) can lead to a model which is unsuited for real-world use, even though its performance is reported as very high.

Utility is the extent to which a model is practical to apply, and whether it produces results that can be used for their intended purpose. Utility is, of course, modulated by performance, as a model which has an unacceptably high error rate does not have utility. Other than raw performance, specific difficulties will arise in any application, which limit the utility of an approach. For example, consider a crop type classification algorithm. Does the algorithm output the species of plant, or its physical characteristics (e.g. broad vs. thin leaf plants)? Is this classification scheme the correct one for the question one wishes to answer? Can the model be run fast enough for the application? Perhaps if the user desires near real-time outputs, then computation time becomes an important constraint. If the model outputs a small but systematic error in some circumstances, e.g. it regularly misclassifies a rare kind of legume for a grain, then for an exercise in which this rare legume is important, it will not be usable. Utility broadly describes these practical considerations, without which even the most accurate algorithm is unlikely to see real-world application.

As we can see, data, performance and utility are not independent of one another. Performance can only be achieved with sufficient data. Utility can only be achieved with sufficient performance. In this way, they form a chain, beginning with data, then performance, and then utility, although other factors can influence each of them outside of this relationship (see Figure 1.4). One might assume, given this relationship, that any positive change to data or performance would lead to an increase in utility, however this is not the case. I would argue that in many situations, during the planning and implementation of deep learning research in remote sensing, the positive correlation between them is broken, and utility is actually *sacrificed* for data and performance, and through this we inhibit the generation of high-value outcomes.

This trade-off against utility can materialise in the common strategy of pursuing a “toy problem” before considering the real-world problem. This strategy allows us to find prototype solutions without all the complications of real-world application, in a setting that is (hopefully) similar enough to the real thing that there is some correlation between performance in the toy problem and the real world. Usually, constructing a toy problem involves selecting a subset of data which is conducive to high performance, by only using images without noise or artefacts, or by removing difficult examples. It could also be seen

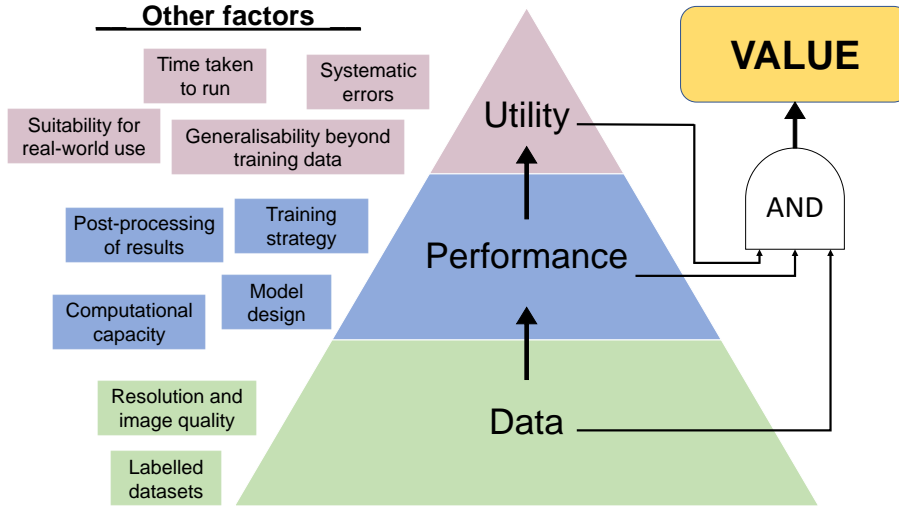


Figure 1.4: Prerequisites for an approach to provide value. Performance requires data, and utility requires performance, but other factors (examples of which are on the left) affect each of these.

as a toy problem if a dataset is used which is sampled in a way that is unrepresentative of the real-world task, e.g. cloud masks over specific countries or regions. Unfortunately, few studies move from the safety of this toy problem back into a real-world application successfully, because the nuances which distinguish the real problem from the toy one are less easily dealt with than anticipated, and thus the technique’s utility remains low, even whilst performance in the toy problem is high.

For example, a significant portion of crater detection work focuses on the detection of large craters. Small crater detection must contend with resolution limits, noise, and small, fuzzy datasets with low agreement—even between experts [12]—for training and testing. But small crater detection is nevertheless a task with concrete utility, expanding our capability to count craters by orders of magnitude. Meanwhile, large crater detection is much simpler due to the higher resolution relative to crater diameter, and is afforded relatively complete and trustworthy datasets (such as the global catalogue of all Martian craters above 1 km diameter [11]) to train and test models on. However, large crater detection is not a particularly useful task, given every crater above 1 km diameter has already been found on Mars (unless it could be shown that this ability is transferable to detecting smaller craters). By moving to large crater detection, studies sacrifice utility (and thus value) for data and performance.

When engaging in deep learning research, then, it is always vital to ask two questions in order to be certain that utility is not sacrificed for performance. First, whether the problem being tackled is representative of a real-world problem that people desire a solution to (as opposed to a toy problem which is similar to, but not the same as, the

real-world task it emulates). Second, whether the low performance of existing solutions is the reason that scientific enquiry is being held back regarding this topic. If the answer to either is negative, then perhaps research into expanding the utility of existing techniques would be a more fruitful endeavour than seeking higher performance in a comparative exercise with others.

### 1.3.3 Satellite Data's Structure

Having established a framework for how research into automation can provide value, I will now move on to discuss how the specific structure of optical satellite data shapes the direction, nature, and impact of deep learning research. My argument here assumes some understanding of multispectral imagery and satellites, and so the reader may find it helpful to refer to Section 2.1.

Orbital imagery of Earth and other planets is distinct from other image domains. The resolution of remote sensing data is hugely varied, with sensors able to take images of Earth at a few tens of centimetres per pixel (e.g. the 41 cm/pixel resolution of GeoEye-1 [13]), then through the range of tens of metres (e.g. Landsat 8's 30 m/pixel resolution) and then even coarser at the range of hundreds of metres or kilometres (e.g. Sentinel-3's SLSTR sensor, with bands at 500 m and 1 km resolution). A similar range exists for cameras orbiting Mars and other planetary bodies. Satellite imagery, then, spans over three orders of magnitude in resolution.

Of course, images taken with handheld cameras can also display this range of resolutions. Consider a picture of a person in the foreground, with a landscape in the distance behind them. Each object in an image such as this is strongly correlated with its resolution range; if the landscape behind the person is close enough to the camera for very high resolutions, then its three dimensional nature becomes more apparent, and it no longer appears as a single background, but as multiple foreground objects. Conversely, if the person moves far away from the camera so that they are barely resolved, it is unlikely that they are of interest to the viewer as an individual object. Mammalian vision has evolved to strongly distinguish between foreground and background if possible (e.g. in monkeys [14] and mice [15]), because generally objects close to the viewer are of more immediate importance. Likewise, many computer vision tasks using handheld camera imagery focus on describing targets in the foreground of the image, and treat the background as just that—a background. Conversely, in satellite imagery, everywhere you look is foreground, because the surface is effectively flat when seen from space. This does not mean that depth information cannot be measured, of course, but that the vertical range of the surface is very small compared to the distance from the sensor.

As well as the lack of foreground-background, satellite imagery also lacks a sidedness. Handheld camera imagery almost always has a floor at the bottom of the image, with things above it, often casting shadows onto said floor, and possibly a sky or wall or ceiling in the upper section of the image. This leads to a correlation between the regions of the camera frame and certain targets of interest. Bird’s-eye view satellite imagery has no correlation between position in the frame and the objects one finds there. Of course, in some images, there may be a much higher density of interesting things in one area, but across many images, the region of the sensor that those interesting things appear in is random.

These are some of the spatial peculiarities of satellite imagery, but we can also consider its spectral characteristics. Satellite sensors are flown with specific scientific or technological goals in mind, which frequently require novel designs and bespoke hardware. The wavelengths of light that sensors capture vary widely between satellites, with some using single (panchromatic) bands (e.g. the Context Camera, CTX, on the Mars Reconnaissance Orbiter), and others using the typical RGB combination that mimics our own eye’s cones (e.g. Carbonite-2). Beyond this, many satellites employ non-visible optical bands, throughout the near infrared, shortwave infrared, and thermal infrared. Each sensor may have different combinations of these bands, informed by their specific user needs. Contrast this diversity with other image domains, in many of which RGB is near-ubiquitous, like in handheld camera imagery or in image files found online.

I would argue that these unique characteristics of optical satellite data have impacted the way in which deep learning research is conducted in remote sensing. In fact, returning to the triad of data, performance and utility, the structure of remote sensing data presents challenges for both data and utility. Performance is also affected, but largely as a consequence of the issues with data, and so is left out here.

**Data:** In all supervised classification problems, labelled datasets are created for the dual purpose of training and testing predictive models. The labour required to annotate datasets is substantial in all fields of computer vision [16], but is particularly relevant in remote sensing, because each dataset is specific to not only a given problem, but also a given sensor. In practical terms, this has led to a somewhat fractured set of data collection projects, each producing relatively modest datasets that target a specific satellite. These cannot easily be combined to add cumulative value to a predictive model, although exceptions to this do exist, for example the many high-resolution satellites that use similar sensor designs with RGB channels only.

The geometrical peculiarities of remote sensing data mean that—for many object detection tasks—the objects of interest are close to the satellite’s resolution limit. On Earth, many man-made structures and vehicles, which are at length scales of metres or tens of metres, tend to span many fewer pixels than the foreground objects traditionally detected in computer vision. Similarly, crater detection will always be dominated by resolution effects, because there are exponentially more small craters than large ones [17], meaning any chosen resolution will have more craters at the very lowest visible size range than above it.

**Utility:** When applying a trained model to real-world data, non-sensor independent models are immediately constrained to only those sensors on which they have been trained, (or perhaps to those which have very similar data). This directly reduces an algorithm’s utility. Another issue is that algorithms used for data processing by satellite operators are usually designed before launch, so that they can be applied immediately when the satellite begins to downlink data. Machine learning and deep learning techniques—despite commonly exhibiting better results than other methods—are rarely used in this setting. This is often because without access to training data prior to launch, it is not easy to develop a model which relies on supervised training. A similar situation exists for on-board deployment of machine learning.

### 1.3.4 Sensor Independence

Sensor independence is a timely and important progression for deep learning in remote sensing because of how it cuts across all of the previously mentioned issues. Sensor independence is a way of boosting value for both technological and scientific problems. It does so by targeting several of the prerequisites for value (Figure 1.4), in a way that is informed and shaped by the specific structure of satellite data.

Sensor independence improves the situation with data straightforwardly, by expanding the datasets available for training and testing of models. This can then increase performance where models were previously limited by the quantity of training data. At the same time as this improvement in data and performance, utility is also expanded by the ability to use the trained model on many satellites, opening up possibilities for studies using multiple remote sensing instruments. This means that sensor independence can add value in both technological and scientific problems. Sensor independence is able to do this because it does not seek to reduce the scope of the work, like in a toy problem. Instead, it broadens the technique’s generalisability.

The framing of research questions in other fields of computer vision presupposes a generalisability that is not currently mirrored in remote sensing. Whilst the effort required by researchers in remote sensing is multiplied by the number of sensors, the importance of their results, when published, is in some sense divided by the number of sensors. This is because the results of a model on a specific sensor are not generalisable to the same problem on another satellite’s data. Sensor independence seeks to cut through this and allow for research that is impactful across all (or at least many) optical instruments. If successful, this could massively increase the pace of development of deep learning in remote sensing, allowing for the full potential of the technology to be realised, and for researchers to spend more time asking and answering scientific questions, rather than on creating datasets, designing models, and validating them.

## 1.4 Thesis Outline

After this introduction, Chapter 2 details the concepts, theory, and prior work on both cloud masking and crater detection, and gives a review of the machine learning techniques used in the work. In the case of crater detection, given the scientific outputs of the case study, there is also a review of our understanding of craters and how they can be used for age-dating of planetary surfaces, as well as a description of Isidis Planitia, the site used to study small craters.

The technical work in this thesis is split into two parts. After the literature review in Chapter 2, cloud masking is tackled in three chapters: First, Chapter 3 is an initial foray into cloud masking with CloudFCN, a novel convolutional model, and its results on Landsat 8 and Carbonite-2 are detailed. Second, there is a description of the Sentinel-2 cloud mask catalogue dataset in Chapter 4. This details the motivation behind the dataset, the software developed for its annotation, and the end product. Finally, in the third cloud masking chapter (Chapter 5), sensor independence is introduced with the design of the SEnSeI architecture, and experiments across multiple satellites and deep learning models are shown. This final chapter also discusses the implications of those cloud masking results.

After the three chapters on cloud masking, the thesis then moves onto Martian crater detection. First, in Chapter 6, a partially unsuccessful attempt to develop a novel deep learning method for crater detection is explored, and reasons as to why it failed are explored. Next, Chapter 7 introduces the ORBYTS (Original Research By Young Twinkle Students) crater dataset, with the multi-annotator labelling method described. Third, in Chapter 8, existing state-of-the-art object detection frameworks are employed and



compared, with sensor independence also being used, leading to high-performance crater detection models. Finally, Chapter 9 is a scientific study over the crater population of Isidis Planitia, Mars—a natural extension of small crater detection and an example of the scientific potential for sensor independent deep learning.

After the chapters covering the work on cloud masking and crater detection, a discussion is given in Chapter 10 which draws on comparisons between the two projects, both their similarities and where they diverge. This interpretation of results leads to the conclusions in Chapter 11, which reviews the contributions of this thesis, and assesses the potential of sensor independent deep learning for optical satellites.



## 2 | Literature Review

### 2.1 Remote Sensing Instruments

The variety of remote sensing instruments is considerable. The work in this thesis focuses on sensors which passively measure electromagnetic radiation (as opposed to active observation, in which a signal is emitted by the instrument towards an object, so as to measure the reflected signal, e.g. synthetic aperture radar). Passive remote sensing instruments take many forms, especially at higher energy scales, such as X-rays, for which focusing and reliable detection becomes more difficult. This thesis considers instruments which detect light at wavelengths spanning roughly between the blue edge of our vision (around 400 nm) up to the thermal infrared, at around 12  $\mu\text{m}$ .

#### 2.1.1 Earth Camera Systems

The two **Sentinel-2** satellites were built by the European Space Agency (ESA), and are operated as part of the EU's Copernicus Programme. Sentinel-2A and -2B have near-identical designs (see Figure 2.1), and for most applications are functionally identical (though there are some differences in measured reflectances [18], and some bands' central wavelengths differ by a few nanometres [19]). The instrument on-board both satellites is known as the MultiSpectral Instrument (MSI), although in this thesis, the system is simply referred to as Sentinel-2. The sensors have 13 spectral bands to capture data over a 290 km swath width. The spectra of the 13 bands range from a wavelength just below blue (the "Coastal Aerosol" band, B01, 443 nm) up to the Shortwave Infrared (the "SWIR 2" band, B12, 2.19  $\mu\text{m}$ ), see Table 2.1 for more details.

The Sentinel-2A and -2B missions were launched in June 2015 and March 2017 respectively. They are in a sun-synchronous orbit at 768 km altitude, with a 98.6° inclination [20]. When passing the orbit's descending node, the local time is 10:30 am, which balances the desire for bright illumination conditions with reducing the total cloud cover (given that clouds tend to form throughout the day, and dissipate overnight). The two satellites are at a 180° phase to one another, creating a combined repeat coverage time of 5 days. Images are captured over all land masses, and their surrounding coastal waters, and distributed to users in 110 km<sup>2</sup> tiles.

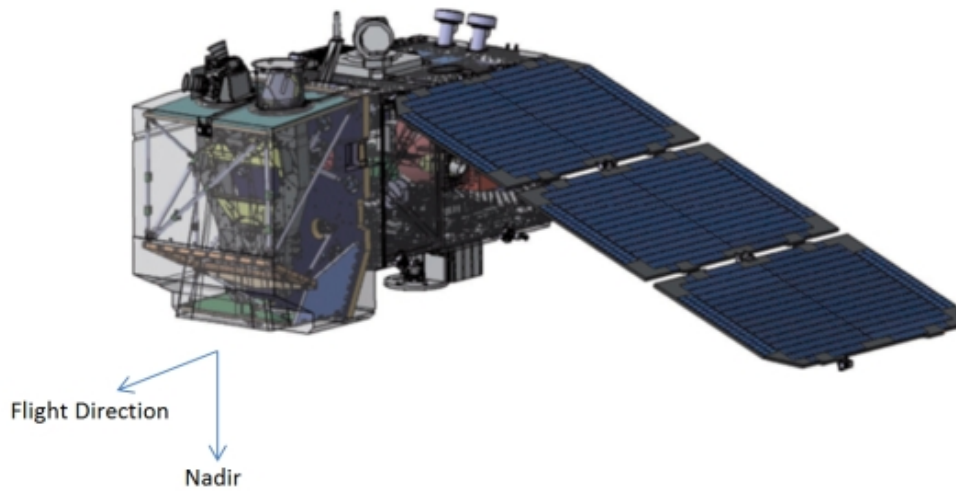


Figure 2.1: Schematic of Sentinel-2 platform. MSI's aperture is at the bottom of the spacecraft. Credit: ESA [20]

Band	Name	Central Wavelength (nm)	Bandwidth (nm)	Resolution (m)
B1	Coastal Aerosol	443	21	60
B2	Blue	490	66	10
B3	Green	560	36	10
B4	Red	665	31	10
B5	Red Edge 1	705	15	20
B6	Red Edge 2	740	15	20
B7	Red Edge 3	783	20	20
B8	NIR	842	106	10
B8A	Red Edge 4	864	21	20
B9	Water Vapour	945	20	60
B10	Cirrus	1375	31	60
B11	SWIR 1	1610	91	20
B12	SWIR 2	2190	175	20

Table 2.1: All 13 Sentinel-2 bands. There are small differences in the central wavelengths between Sentinel-2A and -2B, but it is rarely larger than 2 nm, so an average value is given here. Bandwidths are full width at half maximum. Data collected from [19].

The **Landsat 8** mission is the latest in the Landsat series of satellites, launched in February 2013 by the National Aeronautics and Space Agency (NASA) and operated by the United States Geological Survey (USGS). Two instruments fly on-board the Landsat 8 mission: The Operational Land Imager (OLI) and the Thermal InfraRed Sensor (TIRS). OLI measures radiance across 9 bands, whilst TIRS has two bands at longer (thermal) wavelengths [21]. Table 2.2 gives further details about the Landsat 8 bands. Flying at an altitude of 706 km and an inclination of  $98.2^\circ$ , Landsat 8 passes the descending node of its orbit at a local time of 10:00 am [21], similar to that of Sentinel-2.

Before Landsat 8, there were many prior Landsat missions. **Landsat 7** was launched in April 1999 and is still gathering data with its Enhanced Thematic Mapper + (ETM+) instrument at the time of writing. The design of ETM+ is relatively similar to that of Landsat 8's OLI, given that interoperability between the instruments was desirable. The main differences are that Landsat 7 uses only one thermal band, covering roughly the combined wavelengths of both of Landsat 8's thermal bands. It also lacks the ultra-blue, or "Coastal Aerosol" band at 443 nm found on Landsat 8 and Sentinel-2, as well as the "Cirrus" band at  $1.37\text{ }\mu\text{m}$  [22].

Sentinel-2 and the Landsat missions are key for analysis at the resolution of tens of metres. However, a balance must be struck between resolution and repeat times, given that coarser resolution sensors can provide data over a wider swath using the same downlink bandwidth. **Sentinel-3** is ESA's flagship mission for optical data at the resolution of a few hundred metres. Sentinel-3A and B were launched in 2016 and 2018 respectively, with two more planned satellites C and D over the coming years. On-board Sentinel-3, two multispectral instruments operate: Ocean and Land Colour Instrument (OLCI) [23], and Sea and Land Surface Temperature Radiometer (SLSTR) [24], along with a suite of other instruments.

OLCI is a push broom sensor, with 21 bands on a swath width of 1270 km, and a resolution of 300 m/pixel along and across track [25]. Meanwhile, SLSTR has a resolution of 500 m in 6 bands in the visible and NIR, and 1 km for 3 thermal bands. SLSTR has two viewing angles, one straight down (nadir,  $0^\circ$ ) and one looking back in the direction it came from (oblique,  $55^\circ$ ) [26]. This geometry affords OLCI and SLSTR revisit times of 2 days and 1 day respectively [25, 26].

As well as satellites returning data with resolutions of tens and hundreds of metres, there are also a huge number of sensors taking images at much higher resolution. The recent growth in relatively inexpensive satellites and launches has enabled institutions and companies to begin flying cameras with a much lower cost of entry [28]. The optical cameras on-board these satellites are often designed with off-the-shelf components, and

Band	Name	Central Wavelength (nm)	Bandwidth (nm)	Resolution (m)
B1	Coastal Aerosol	443	16	60
B2	Blue	482	60	10
B3	Green	561	57	10
B4	Red	656	37	10
B5	NIR	865	28	20
B6	SWIR1	1609	84	20
B7	SWIR2	2201	187	20
B8	Panchromatic	590	172	10
B9	Cirrus	1373	20	60
B10	TIRS1	10900	600	60
B11	TIRS2	12000	1000	20

Table 2.2: Landsat 8 bands from both OLI and TIRS. Note that the wavelength does not always increase with band number, as the Panchromatic band (B8) has a wavelength in the visible. Bandwidths are full width at half maximum. Data collected from [27]

commonly take RGB imagery. Many commercial applications rely on higher resolutions than satellites like Landsat and Sentinel-2 can offer, and as a result, the images taken by these instruments tend to have a resolution of roughly a metre, but a much smaller swath width. For example, *Planet Labs* have launched several hundred satellites, taking data from 4 bands (RGB and NIR), with their SkySat fleet taking images at a nominal 0.5 m resolution [29].

### 2.1.2 Mars Camera Systems

The **High Resolution Stereo Camera (HRSC)** is on-board the ESA’s Mars Express mission [30], which began orbiting Mars in 2003 and continues to function. HRSC captures multiple push broom images at different angles, which are provided both as separate 2D images, and also combined to make 3D Digital Terrain Models (DTMs) [31]. On the nadir view, looking straight down, HRSC produces images with a resolution of close to 12.5 m/pixel for most scenes. Sidiropoulos et al. [32] found that, by 2015, HRSC had mapped a substantial portion (64%) of the Martian surface at 12.5 m resolution, although this coverage was below that of the **Context Camera (CTX)**.

CTX, flying on NASA’s Mars Reconnaissance Orbiter (MRO) has been gathering panchromatic image data from Mars since 2006 [33]. With a sun-synchronous orbit of roughly 300 km altitude, CTX takes images of the planet’s surface at 3pm local time, at a nominal resolution of 6 m per pixel. Like many satellite camera systems, CTX has a push broom sensor configuration—a single line of pixels orthogonal to the direction of flight, which take successive measurements as the satellite travels. The detector is a CCD, 5000 pixels across, sensitive to light between 500-700 nm [34]. As [33] notes, CTX’s

key scientific strength is the combination of moderately high resolution and near-global coverage. Indeed, as of 2017, CTX had a global coverage of 99.1% [35].

Sharing space on-board MRO with CTX, the **High Resolution Imaging Science Experiment (HiRISE)** is designed to take high resolution (around 30 cm per pixel), colour imagery of Mars. The instrument uses 14 detectors, ten of which are red (550-850 nm), two blue-green (400-600 nm) and two NIR (800-1000 nm) [36]. Together, these create a thin colour image in the central 20% of the swath, between two wider flanks of grayscale from the red band [37]. HiRISE has provided the scientific community with an invaluable and unique catalogue of images at resolutions that far surpass any other previous camera system. However, this comes at a cost, given that HiRISE had only mapped 2.4% of the surface as of 2016 [38], and will never be able to conduct a complete survey of the planet. For this reason, HiRISE has often been used for focused analyses of specific small-scale features, or areas on the planet’s surface, (e.g. recurring slope lineae [39]). HiRISE was used extensively to map the landing site of the Perseverance Rover in Jezero crater [40], as it allowed the mission operations teams to identify features and regions of geological interest.

The **Thermal Emission Imaging System (THEMIS)** flies on-board the 2001 Mars Odyssey, the oldest spacecraft still functioning in Martian orbit. Designed as a successor to the Thermal Emission Spectrometer (TES) onboard Mars Global Surveyor, THEMIS measures thermal infrared radiation at 9 wavelengths and 100 m/pixel resolution, along with 5 visible bands at 18 m/pixel [41]. By comparing observations of the thermal infrared bands at day and night, mineralogical composition and thermal inertia can be measured. Of more relevance to this thesis, given its use in Chapter 8, is the global daytime mosaic, made from a combination of the 9 thermal infrared bands [42]. This averaged global mosaic gives the temperature of the Martian surface in the daytime. Because of the increased temperature in areas exposed to direct sunlight, shadowed regions appear colder (and thus darker). Visually, morphological surface features such as craters and ridges look similar in the THEMIS IR daytime mosaic and visible imagery. Robbins et al. [43] used this mosaic to complete their global catalogue of craters above 1 km diameter.

## 2.2 Machine Learning

Machine learning is a family of algorithms with which optimisation is achieved automatically on some task. The purpose of these algorithms is not restricted to any given family of problems, nor are the algorithm’s design, or the scheme through which they

are optimised, restricted. In practice, the algebraic mechanics of a specific model can be rather opaque, given that machine learning models commonly have a huge number of parameters. Due to this internal complexity, machine learning models are often said to be a “black box” because one cannot easily probe why a model gave the answer it did. Much research has gone into improving the explainability of machine learning prediction (e.g. [44]), but one must accept that—depending on the context—the mappings of input variables to output predictions can be highly complex and unintuitive. Even if the large numbers of parameters make the predictions of machine learning models difficult to explain, their basic mathematical structure—the form that the internal equations take—is often simple.

Traditional mathematical modelling techniques rely on an expert’s understanding of the system: By applying useful constraints to the model’s equations, the expert creates a system where there are only a few parameters and each have some physical relevance. However, for problems in which the underlying relationships are not well understood, or where the input space has an impractically large number of dimensions to consider, it is difficult to design physically-informed models manually. For these problems, modern machine learning techniques are promising, because they apply very few constraints to the mappings between input and output. For this reason, in many scientific fields which work with highly complex data (e.g. remote sensing), machine learning has found an important role.

This review begins by introducing the basic concepts of neural network design, and then moves onto specific ideas pertinent to the thesis: permutation invariant neural networks (PINNs), and convolutional neural networks (CNNs) designed for semantic segmentation and object detection in images.

### 2.2.1 Neural Networks

Neural networks were originally proposed in 1943 by Warren McCulloch and Walter Pitts [45]. At that time, it was seen as a potential way to model and better understand brain activity, by simulating neuronal behaviour. By 1958, Frank Rosenblatt had successfully built the *Mark 1 Perceptron*. With 400 photovoltaic cells as input, and mechanical engines to alter its weights, it could be used for simple image recognition tasks.

Unfortunately, research into neural networks stalled for many years, due to the massive computational requirements needed in order to calculate the gradients of each neural connection’s weight with respect to the error. A key breakthrough, published in 1986,



which reinvigorated research on neural networks, was backpropagation [46]. Backpropagation forms the basis of modern neural networks, by offering a method to calculate the gradients of multi-layer networks in linear time. Backpropagation will be returned to shortly, in Section 2.2.2.

At their most simple, neural networks consist of a set of successive matrix multiplication layers, interspersed with nonlinear activations. Each layer receives an input vector of length  $n_i$ , and outputs a vector of length  $n_{i+1}$ . Within each layer, there is an  $n_i \times n_{i+1}$  matrix of weights,  $W_i$ , and a vector of biases,  $b_i$ . The output of the layer,  $\mathbf{x}_{i+1}$ , is the matrix multiplication of  $W_i$  with the input  $\mathbf{x}_i$ , added to the biases, and then put through a nonlinear activation function,  $\sigma$ ,

$$\mathbf{x}_{i+1} = \sigma(W_i \mathbf{x}_i + b_i) \quad (2.1)$$

### Activation Functions

In equation 2.1, the activation function,  $\sigma$ , is vital. The nonlinear activation function is used so that successive layers can represent nonlinear mappings. Without such a nonlinear function, the stacked layers would be redundant, as the product of multiple linear transformations can always be expressed as a single linear transformation. For example, for two layers, we can see it is possible to reduce the terms back to a single matrix multiplication and bias addition

$$W_0(W_1 \mathbf{x} + b_1) + b_0 = W_0 W_1 \mathbf{x} + W_0 b_1 + b_0 = W' \mathbf{x} + b' \quad (2.2)$$

Significant research has gone into the possible functions one can use for the nonlinear activation,  $\sigma$ . The sigmoid function was a popular choice for many years, as it has many useful features. It is well defined for all inputs and its outputs are in the interval  $[0, 1]$ . The functional form of the sigmoid is

$$\text{sigmoid}(x) = \frac{1}{1 + e^{-x}} \quad (2.3)$$

Similar to the sigmoid, the hyperbolic tangent ( $\tanh$ ) is also often used as an activation function. Rather than outputs in the interval  $[0, 1]$ , the  $\tanh$  produces activations between  $[-1, 1]$ . Sigmoid and  $\tanh$  functions do have a number of issues, however. They are relatively computationally expensive, and they suffer from the *vanishing gradient problem*, which happens when input values have very large magnitudes, making

the derivatives of the activation function effectively zero. This means that weights are updated incredibly slowly, and networks can consequently fail to train.

More recently, Rectified Linear Unit (ReLU) layers have become commonly used [47]. The ReLU is a simple nonlinear function,  $y = \max\{0, x\}$ . There are several advantages to this over a sigmoid and tanh: it is very computationally efficient, and gradients do not vanish at  $x = +\infty$ . However, gradients are exactly 0 for  $x < 0$ , which can also lead to a vanishing gradient problem. To combat this, variants to the ReLU have been proposed. A popular choice is the LeakyReLU [48], which adds a small gradient to values below 0. This is done by defining the activation as  $y = \max\{\alpha x, x\}$  where  $\alpha$  tends to be a small positive value (e.g. 0.01). This retains the nonlinearity of the output, but guarantees a small non-zero gradient for all values of  $x$ .

For networks which aim to assign a confidence to different classes at the final layer of the network, it is common to use the “softmax” activation function [49]. This activation acts on a vector of inputs to produce a normalised output which always sums to 1,

$$\text{softmax}(\mathbf{x})_i = \frac{e^{x_i}}{\sum_j e^{x_j}} \quad (2.4)$$

and was designed so that predicted confidence values obey Luce’s choice axiom [50]

## Other Layers

As well as the neural layers, and nonlinear activations, there are additional layers which are seen widely in neural network design. For example, normalisation layers are commonly used in order to improve training, by adjusting the distribution of values at a given point. Batch Normalization (BN) layers do this by rescaling the distribution of values to have a mean of 0 and a standard deviation of 1 [51].

In a similar vein to BN layers, Group Normalization (GN) layers also act to make the distribution of values follow a Gaussian distribution [52]. However, the dimensions over which the distribution is computed is altered. Rather than looking across the batch at each feature, the GN layer enforces a Gaussian distribution within the features of each sample in a batch, or in multiple groups of features. When used on small batch sizes, BN layers can suffer because of the noisy distribution of each feature in a small batch, whereas GN layers, pooling the distributions of groups of features together, are more robust.

Dropout layers are commonly used in neural networks to reduce overfitting on data [53]. In training, the dropout layer randomly switches some percentage of the values in a ma-

trix to 0. By doing so, it adds randomness to the signal passed to later layers, which prevents them from specialising too much on the training data and overfitting.

### 2.2.2 Backpropagation & Loss Functions

The weights and biases within neural networks have the potential to express a huge range of highly complex, nonlinear functions, but some optimization strategy is needed to find a suitable set of weights for a given task. Iterative optimization of parameters traditionally calculates the gradient of the error with respect to each parameter, and adjusts each accordingly, by a small amount. This would also work for neural networks, however the number of partial derivatives that must be calculated explodes as the number of parameters in the neural network's layers increase. This is because partial derivatives of each variable with respect to every other variable are required for a full calculation of the gradient of the error with respect to a given variable. Backpropagation rests on a powerful observation, which massively simplifies these calculations. For any network with several layers, the gradients can be computed for each successive layer in reverse order [46], [54], making it computationally feasible to rapidly update weights in even very large networks.

Backpropagation allows for us to update the weights based on the gradient of the loss in weight space. The choice of loss function, therefore, is also important in the training of neural networks. Broadly, the loss function is some kind of measure of the distance between the model's prediction, and the desired output. In order for gradient descent to work, the loss function should be continuous, and differentiable.

**Mean Square Error (MSE)** is a common loss function, and is also used widely in optimization problems outside of machine learning. Within machine learning, it is very often used for regression tasks, where the output is not a categorical variable, but a continuous value. We define the MSE loss,  $\mathcal{L}$ , with respect to the true answer,  $x_i$ , and the prediction,  $\bar{x}_i$ , over all  $N$  elements of the output, as

$$\mathcal{L}(\bar{\mathbf{x}}, \mathbf{x}) = \frac{1}{N} \sum_i (\bar{x}_i - x_i)^2 \quad (2.5)$$

**Categorical cross-entropy** is popular for tasks with outputs that are discrete and class-based, rather than regression problems where the correct answer lies on a continuum [55]. It is defined for binary values,  $x_i$  and predicted confidence values,  $\bar{x}_i$ , lying between 0 and 1, as

$$\mathcal{L}(\bar{\mathbf{x}}, \mathbf{x}) = -\frac{1}{N} \sum_i \log(1 - |x_i - \bar{x}_i|) \quad (2.6)$$

**Focal loss** has recently been suggested to improve object detection performance for one-stage models (for more information about object detection models, see Section 2.2.5). For object detectors, there is a huge class imbalance between bounding boxes with no object (background), and bounding boxes with objects (foreground), because the vast majority of bounding boxes do not cover an object. Focal loss reduces the loss for high-confidence predictions, meaning the large numbers of background bounding boxes do not saturate the loss and cause the model to predict everything as background. This is achieved by setting the focusing parameter,  $\gamma$ , to a value above zero, where

$$\mathcal{L}(\bar{\mathbf{x}}, \mathbf{x}) = -\frac{1}{N} \sum_i |x_i - \bar{x}_i|^\gamma \log(1 - |x_i - \bar{x}_i|) \quad (2.7)$$

These are prominent examples of the countless loss functions proposed for models across machine learning, and are the ones relevant to the work in this thesis. Ultimately, though, loss functions are always a function comparing a desired output with the model’s predictions. These losses are used to update the weights of the model, layer by layer, through backpropagation.

### 2.2.3 Convolutional Neural Networks

Like traditional feature transformations, CNNs extract high-level semantic features from images. However, the transformation is a dynamic function that can be optimised during a training phase. This malleability is likely what has made CNNs so popular and successful in so many applications. For context, in 2012 (the year AlexNet achieved a record score on the ImageNet challenge [5]) the phrase “convolutional neural networks” was used 2,000 times in academic publications. Five years later in 2017, it was used 22,000 times<sup>1</sup>. This explosion in use is largely due to increasing computational power allowing for “deeper” architectures, being trained on more data. Depth, in the context of neural networks, refers to the number of layers in the model, which is directly related to the non-linearity of possible feature representations that the network can learn [56].

A convolutional layer works by measuring the presence of a set of translation-equivariant features across the whole input. This equivariance means that, if a certain feature appears in a specific part of an input, then its position is also localised to the corresponding

---

<sup>1</sup>Found using *semanticscholar.org*

area in the output space. This can be represented by augmenting Eqn 2.1, such that the weight matrix of the  $i^{th}$  layer,  $W_i$ , is convolved with the input to the layer,  $\mathbf{x}_i$ , as

$$\mathbf{x}_{i+1} = \sigma(W_i \otimes \mathbf{x}_i + b_i) \quad (2.8)$$

By convolving, rather than simply matrix multiplying as in a normal neural layer, we are limiting the neural connections to only those inputs which are in the vicinity of a given output node. This works well because of the observation that pixels close to one another in an image are more likely to have important relationships with one another, whereas pixels on the far sides of the image are less likely to be related. As well as limiting the possible connections to a small vicinity around each point, we are also enforcing the idea that features are translation invariant: if a feature is worth learning, then it is worth looking for it everywhere in the image, not at a specific location. This invariance is achieved by setting the convolutional kernel to have the same weight matrix for every position in the image.

Over the last few years, a huge range of extensions have been made to the core principle of CNNs. Many are aimed specifically at getting past two common problems for deep networks. (i) the vanishing gradient problem, which stems from the fact that as we backpropagate a loss through a network's layers, the gradients tend to go very close to zero, meaning the lower layers in the network remain untrained. (ii) over-fitting, which happens with networks that have a large number of parameters, but a small number of training examples. Over-fitted models learn to recognise the specific examples of the training set, rather than learning the general rules that allow for extrapolation to new data.

For problem (i), batch normalisation [51] and residual connections [57] have become standard. Batch normalisation ensures that the input variance at each layer is 1, which tends to amplify the loss as it is backpropagated, allowing for non-zero gradients to reach the lower layers. Residual connections provide direct skipping connections between layers, so that information travels from the first to the last layer very quickly, allowing for gradients to be passed backwards more efficiently. To solve problem (ii), it turns out batch normalisation also helps. Another popular technique (especially when dataset size is limited) is data augmentation (e.g. [5, 58]).

When discussing the architecture of a model, we use the term *hyperparameters* to describe the aspects of the model which are fixed, and not directly learned through training. For example, the number of features (channel depth) outputted by a given layer is a hyperparameter. For convolutional layers, three common hyperparameters are:

- **Stride** is the number of pixel positions moved between each individual convolution. The stride controls the size of the outputted feature map. A higher stride means the output will be smaller, whilst a stride=1 will lead to an output map that is identical in size to the input of the layer (discounting the geometry at the edge of the image).
- **Channel depth** is the number of features outputted from the layer. For example, RGB data has a channel depth of 3.
- **Filter size** is the width and height of the kernel used in convolution. Smaller filters look for features that are more localised (e.g. 1x1 or 3x3) whilst larger filters fuse information from larger areas (e.g. 5x5 or 7x7). Generally odd numbers are selected, so that there is a central pixel.

The next sections will turn to look at the kinds of problems that CNNs solve in image processing. Whilst there are a variety of tasks that CNNs can be used for, this review just considers segmentation and object detection, as they are the categories of problem which cloud masking and crater detection fall under.

## 2.2.4 Segmentation

Semantic segmentation of images consists of producing a map, showing which class each pixel in the original image belongs to. Individual objects are not separated, but each class is considered as a continuous function across the image. This is well-suited to problems like cloud masking, for which there is often no clear boundary between one cloud and another. Instead, we wish to know where in the image there is cloud. This section looks at the deep learning segmentation models used or relevant to this thesis.

### U-Net

U-Net was first proposed for segmentation of biomedical images [58]. Like many segmentation models, it has an encoder-decoder architecture. This means it has a CNN responsible for distilling spatial information in the image into a feature representation, and then a decoder which transforms these features into an output with the same spatial dimensions as the input image.

The decoder does this using transpose convolution layers. Whilst a standard convolution, or max pooling layer, computes a given output from a small area of inputs (determined by the size of the kernel), a transpose convolution instead computes an area

of outputs from the features in a single input location. In this way, transpose convolutions *expand* the spatial extent of the feature map, and can be used to go from the spatially small output of a typical set of convolutional layers, back to the original image's dimensions.

As well as the main encoder-decoder of U-Net, connections which skip parts of it are also made (see Figure 1 in Ronneberger et al. [58]). At each level of the encoder, the data is copied across to the corresponding layer of the decoder which shares the same spatial scale, and concatenated with the feature map there. By doing this, information about low-level features are passed straight to the last few layers of the network, and mix with the deeper features which come from a longer chain of convolutions.

U-Net was highly successful in its original application, and was especially impressive given the small training set that was available. Since then, it has been applied widely in computer vision (e.g. [59,60]). CloudFCN, the model proposed in Chapter 3, is based on this design.

## DeepLab family

**DeepLab** introduced atrous convolutions to the problem of image segmentation [61]. Atrous convolutions are like a standard convolution, but with a dilated kernel (or a kernel with holes—*à trous*). Atrous convolutions work by connecting the same sized convolutional kernel (e.g. a 3-by-3 grid) with a larger area, by spreading out the input locations. This creates layers that gather information from a wider area, but that use the same number of weights. It also means max pooling layers are not needed in the network, because spatial information can be taken from a much larger field of view without them. DeepLab takes existing pretrained CNNs such as VGG-16 or ResNet-101, and removes their pooling layers, and replaces standard convolutions with atrous ones.

Rather than applying transpose convolution layers to expand the feature map's resolution like in U-Net, DeepLab just used bilinear interpolation to increase the resolution of the output of the atrous CNN back to the size of the image. These bilinearly interpolated features are then fed into a softmax layer which output the class confidences at each pixel. To improve these final outputs, a Conditional Random Field (CRF) [62] is used, which reduced the noise in detections by making it more likely that neighbouring pixels agreed with one another.

**DeepLabv2** improves on the original by extracting features with a set of different atrous rates [63]. The feature maps extracted using these different atrous rates are combined using a new mechanism called atrous spatial pyramid pooling. This fuses

information from different scales so that predictions in each pixel can use information from a range of fields of view.

**DeepLabv3** [64] re-examines atrous convolutions by considering how high atrous rates can lead to many of the weights being applied outside of the valid image dimensions, and are being padded with zeros. This limits the ability of the model to share global information about the image effectively across all pixels. Therefore, as well as the existing atrous convolutions, the feature map is also globally pooled, producing a 1-by-1 feature vector, which is then concatenated to the features outputted by atrous spatial pyramid pooling.

DeepLabv3 also removes the CRF used in versions 1 and 2, finding faster computation and better results without it. Instead, they output the final softmax predictions at a resolution 8 times lower than the original image, and bilinearly resample them. This works well because generally boundary predictions are smooth and so the bilinear resampling imitates this natural smoothness.

**DeepLabv3+** is the most recent model in the DeepLab family [65]. It replaces the convolutional layers used in other DeepLab models with depthwise separable convolutions [66]. These split a single convolution into two, with the first acting only spatially (no mixing of different features) and the second as a 1-by-1 convolution which then mixes the features. This massively reduces the number of weights per layer, and speeds up computation. DeepLabv3+ also adds a decoder module. This takes low-level features from the original image and combines them with the outputs of the atrous spatial pyramid pooling, and then they are convolved together.

## 2.2.5 Object Detection

Whilst segmentation models map where different categories are found across an image, and how much of those categories there are, it cannot tell you how many of a given object there are. Object detection, instead, locates individual instances of the class of objects and describes their extent using a bounding box, treating them as a set of separate entities, rather than as a continuous mask.

A huge variety of object detection frameworks have been proposed. Broadly, they can be placed into two categories: multi-stage detection, and one-stage detection. Multi-stage detection first localises objects in the image, and then classifies them, in a two-step process. Meanwhile, one-stage detection both classifies and localises objects at the same time. In this thesis, one example of each is employed in Chapter 8, and so this section explores the development of both of these models, and their predecessors.



## R-CNN family

**R-CNN** (Regions with CNN features) [67] was introduced as a way of translating the recent successes of CNNs in image classification to object detection. R-CNN is a multi-stage approach, and so first finds proposal regions, and then classifies those regions as one of a range of object categories, or as background. R-CNN is not dependent on a specific region proposal, and used an existing technique which was popular at the time: selective search [68]. After 2000 region proposals are made with selective search, a 4096-dimensional feature vector for each proposal box is extracted, by resizing the image inside the proposal box to a standard size (227 pixels squared) and inputting it into a five-layer CNN. Those feature vectors can then be used to classify the proposal as one of the object categories used during training, and is done so by using a Support Vector Machine (SVM) [69]. R-CNN performed well, but computation time was an issue, because training was done in two stages (the CNN and the SVM), and because the CNN must be used on every one of the 2000 region proposals separately.

**Fast R-CNN** improved on this design in a number of ways, increasing performance and decreasing computation time significantly [70]. The region proposals are computed in the same way as R-CNN, using a method such as selective search. However, instead of computing a feature vector for each individual region proposal, Fast R-CNN reduces the amount of computation required by first processing the entire image through the CNN, and then sharing those computed features with all the region proposals. It does this with a Region of Interest (RoI) pooling layer, which uses max pooling to generate a fixed-size version of the CNN's output for every region proposal. So, instead of each region proposal leading to a 227-by-227 pixel image that is fed into a CNN, Fast R-CNN instead first computes the features of the whole image, and then pools those features within each region proposal.

Additionally, Fast R-CNN removed the SVM used for classification and bounding box fine-tuning in R-CNN, and replaced it instead with a neural network. This allowed the entire network to be trained simultaneously, using a multi-task loss (one loss for classification, the other for bounding box regression). These improvements allowed Fast R-CNN to train 9x faster than R-CNN, and, when used for predictions, is 213x faster [70].

**Faster R-CNN** is yet another improvement on the R-CNN model architecture [71]. Faster R-CNN is fully convolutional, and introduces a new module, called the Region Proposal Network (RPN). This module replaces the traditional region proposal methods like selective search, and instead uses a CNN to create region proposals. First a CNN extracts features over the entire image, as was done in Fast R-CNN. Possible regions are

then sampled uniformly across the feature map, using anchor boxes to generate candidate regions at different scales and aspect ratios. By default, there are 9 of these anchor boxes at each position, at 3 different scales (128, 256 and 512 pixels across) each with 3 different aspect ratios (2:1, 1:1 and 1:2 with respect to width and height), making 9 total anchors per location.

The RPN computes an objectness score (its confidence that an object exists in that location) for each of these anchor boxes, along with a bounding box regression (small shifts in the anchor box's geometry) using a fully connected neural network. Those with the highest objectness score are then sent to the RoI pooling and classifier as was done in Fast R-CNN. This means the entire pipeline is now a convolutional network, and the RPN, CNN, and classifier can all be trained simultaneously.

**Mask R-CNN** extends the ability of Faster R-CNN from object detection to *instance segmentation* [72]. This can be thought of as a hybrid between segmentation and object detection, where each object is given a bounding box and a mask. The mask denotes the precise shape of the object within the bounding box, as is similar to the output of a segmentation model, but just within the bounding box of an object.

Mask R-CNN achieves this with the addition of another output branch to the existing Faster R-CNN framework which it is based on. This additional branch takes the CNN features extracted from the image, taken from the bounding box region, and computes a segmentation mask from it. The RoI pooling layer proposed in Fast and Faster R-CNN has a drawback in this regard, because the positions in the CNN feature map are quantized, meaning it is difficult for the masking module to get fine-grained information about an object's boundaries. In place of RoI pooling, the RoI align layer instead uses bilinear interpolation of the CNN's feature map, so that information about the precise location of objects within the region are better retained<sup>2</sup>.

## YOLO family

**You Only Look Once (YOLO)** is a one-stage object detection framework, which creates bounding boxes and class labels in a single network [73]. The image is resized to 448-by-448 pixels, and inputted into a CNN which has 24 convolutional layers. The CNN outputs a 7-by-7-by-1024 feature map. Unlike R-CNN models, which would then use such a feature map to create bounding box predictions, YOLO directly predicts the bounding box values and class probabilities for a set of 7-by-7 grid points. The features are put

---

<sup>2</sup>For an intuitive, graphical summary of how RoI align works, see: <https://erdem.pl/2020/02/understanding-region-of-interest-part-2-ro-i-align>

through two fully connected neural layers, first converted into a single 4096-dimensional vector, and then into a 7-by-7-by-N output.

This is the final output of the model, and contains an N-dimensional vector corresponding to each of the 7-by-7 grid positions. The N features comprise the bounding box coordinates for a set of possible bounding boxes at the grid cell (dependent on the number of different anchor boxes one uses) the objectness scores for each of these bounding boxes, and the class predictions. The final object detections are found by discarding those bounding boxes with low objectness scores, and keeping those with high objectness scores.

YOLO is exceptionally fast compared to models like Faster R-CNN, however it does produce more localisation errors in its bounding box predictions, and can fail when objects are densely clustered, because each of the 7-by-7 grid positions can only predict one object, even if it has many anchor boxes. However, it does hold an advantage over the R-CNN models in that information for the entire image is fed into the predictions of each bounding box, meaning these predictions can benefit from the wider context of the image.

**YOLOv2** improves upon the simple convolutional design of YOLO with a set of additions and tweaks to the core model [74]. The largest change is the use of anchor boxes, each of which can contain an object, rather than the strategy of only allowing one object per grid cell in YOLO. This means objects close to one another can be more successfully detected. Rather than relying on hard-coded anchor box geometries like in R-CNN models, YOLOv2 instead performs an analysis on the training data before the supervised learning begins, and calculates an optimal set of anchors which will give the closest match to the objects in the training set.

**YOLOv3** again provides several beneficial updates to YOLOv2 [75]. A new convolutional model is used for the feature extraction, and this outputs features taken at 3 different scales, which correspond to features of different spatial size ranges in the input image. This improves the ability of YOLOv3 to classify and detect objects at a larger range of sizes. Given that crater detection is the application covered in this thesis, this is an important improvement, because craters exist over a huge range of sizes. Appendix C gives more information on development of later versions of YOLO.

### 2.2.6 Permutation Invariant Neural Networks

Returning to non-convolutional neural networks, a distinct style of architecture has arisen, in recent years, for tasks which do not have a fixed number of input variables. Permutation Invariant Neural Networks (PINNs) are a broad family of model designs, which

have the property that the output of the network does not change given any re-ordering (permutation) of the set of inputs. The PINN is therefore symmetric with respect to the order of the inputs. This section first introduces the theoretical basis for permutation invariance, and then gives some examples.

Permutation invariance can be achieved through any symmetric pooling operation (e.g. summation, maximisation, or mean averaging) over the outputs of some function acting on a set of inputs. Broadly, a PINN can be seen as applying a transformation (which in the case of a PINN is a neural network),  $\mathcal{F}$ , across the inputs at each index  $i$  (in this section, we use  $i$  to denote the index of the input data, rather than the layers of a neural network) of a set of  $N$  inputs  $X$ , then pooling those  $N$  transformations with an operation,  $\mathcal{P}$ , defined by

$$y = \mathcal{P}(\{\mathcal{F}(X, i) \mid i \in 1, 2, \dots, N\}) \quad (2.9)$$

This formula has the useful property that the size of the vector  $y$  is independent of the number of input elements,  $N$ , allowing for any arbitrary group of inputs to be represented in the same vector space, which a standard neural network can then use.

In simple cases,  $\mathcal{F}$  acts upon each element  $x_i$  individually. However, more complex operations can be performed which rely on other members of the set of inputs  $X$ , such that elements other than  $x_i$  affect the  $i^{th}$  output. Therefore, in the general case, if the output  $y$  is always to be symmetric with respect to permutations on the members of  $X$ , then  $\mathcal{F}$  must be *equivariant* with respect to any index swapping transformation,  $\mathcal{T}$ , on a pair on input indices  $i$  and  $j$ . Equivariance, then, means that any reordering of inputs results in exactly the same reordering of outputs

$$\mathcal{F}(\mathcal{T}_{ij}(X), i) = \mathcal{F}(X, j) \quad (2.10)$$

Several studies have proposed PINN models which satisfy this definition. PointNet [76] and PointNet++ [77] are prominent examples, designed for the classification or segmentation of point cloud data. They process each 3-dimensional point  $(x, y, z)$  into a 1024-dimensional feature vector. These are then combined across all  $N$  points using a max-pooling operation, to produce a single 1024-dimensional vector which holds information on all points in the point cloud simultaneously.

Permutational layers—which combine each pair of  $N$  inputs, and then pool them so as to return  $N$  outputs—are permutationally equivariant (satisfying (2.10)) and have been shown to work well in modelling dynamics [78]. Each permutational layer consists

of a neural network and allows for information about the interactions between inputs to be considered by the network, by looking at each pair. This allowed the model to successfully predict the velocities and positions of idealised discs moving and colliding in 2 dimensions.

Unfortunately, pairwise operations have  $\mathcal{O}(n^2)$  complexity, and so their application may not be practical in tasks with a large number of input points. To alleviate this, PointNet++ [77] limits pairwise processing to only those points within the vicinity of the target point, limiting the computational complexity whilst retaining the fusion of information from nearby points. When these ideas are applied in Chapter 5, this is not a concern, given that the maximum number of spectral bands in multispectral imagery is comparatively low. However, future work concerning sensor independence with hyperspectral data—which contains many more bands—may benefit from an approach which intelligently reduces the number of band pairings considered.

## 2.2.7 Performance Metrics

When comparing models against one another, machine learning studies use several metrics to quantify the validity of predictions against the test set’s labels. Inevitably, each metric emphasises and prioritises different qualities in the model results. This section provides definitions for a range of metrics used in this thesis. Before defining the metrics themselves, though, it is helpful to define a set of four terms which are used to calculate them:

- **True Positives** are the instances for which both the model’s prediction and label say is a given class (e.g. cloudy pixels, or successfully detected craters).
- **True Negatives** are the instances for which both the model’s prediction and label say is *not* a given class. In the case of cloud masking, these are the non-cloudy pixels correctly identified as such. For object detection, this isn’t possible to calculate, or rather, it is in some sense infinite, as it includes every position for which there isn’t a crater.
- **False Positives** are the instances in which the model says it is a given class but the label counts are negative (e.g. pixels predicted to be cloudy which are not, or predicted craters which were not in the labels).
- **False Negatives** are the instances which the model does *not* predict as a given class, but which are labelled as such (e.g. pixels predicted as non-cloudy when they are in fact cloudy, or craters missed by the model).

Overall Accuracy (OA) is the total percentage of correctly identified pixels. It cannot be used for object detection, because True Negatives, included in the formula, are not meaningful in object detection. It is an intuitive metric for segmentation, but can be unrepresentative of practical use when class prevalences are very imbalanced,

$$OA = 100 * \frac{TP + TN}{TP + TN + FP + FN} \quad (2.11)$$

Recall tells us how successful the model is in finding the population of cloudy pixels, or craters. It does not take into account the fact that the model might incorrectly label pixels as cloud, or falsely claim a crater exists where it does not (False Positive).

$$Recall = 100 * \frac{TP}{TP + FN} \quad (2.12)$$

Precision, as opposed to recall, determines the percentage of the predicted positives that were in fact positive. E.g. pixels which were identified as cloudy that are actually cloud, or the fraction of crater predictions that turn out to be real craters. It does not depend on the number of missed cloudy pixels or missed craters (False Negatives), and is defined as

$$Precision = 100 * \frac{TP}{TP + FP} \quad (2.13)$$

The  $F_1$  score is the harmonic average between precision and recall. This means that it always lies between their values, but it is heavily penalised if one is much lower than the other. Of the metrics commonly found in deep learning,  $F_1$  is one of the most widely used as a singular measure of a model's performance, given that it takes into account both precision and recall, calculated as

$$F_1 = 100 * \frac{2 * Precision * Recall}{Precision + Recall} \quad (2.14)$$

Balanced Accuracy (BA) is similar to OA, but it is weighted by the relative prevalence of each class. It can also be thought of as the average of each class' recall, both positive and negative. This is ill-defined for object detection, because the True Negative population is also not defined. BA is defined as

$$BA = 100 * 0.5 * \left( \frac{TP}{TP + FN} + \frac{TN}{TN + FP} \right) \quad (2.15)$$

So far, these metrics have assumed that the predictions are not ranked by confidence, or that some threshold has been applied by which binary classes are found. However, neural networks are able to give confidence values, which allow us to rank detections in order of the model’s certainty. Whilst not a metric *per se*, the Precision-Recall Curve (PR Curve) is a useful concept to consider. The PR Curve plots the model’s precision and recall, as the threshold applied to the outputs is altered (see Figure 2.2). If the confidence values range from 0 – 1, then setting a threshold to 0 would result in a high recall, but a low precision. As the confidence threshold applied is increased, the recall will decrease, and the precision increase, because the model becomes more “picky”.

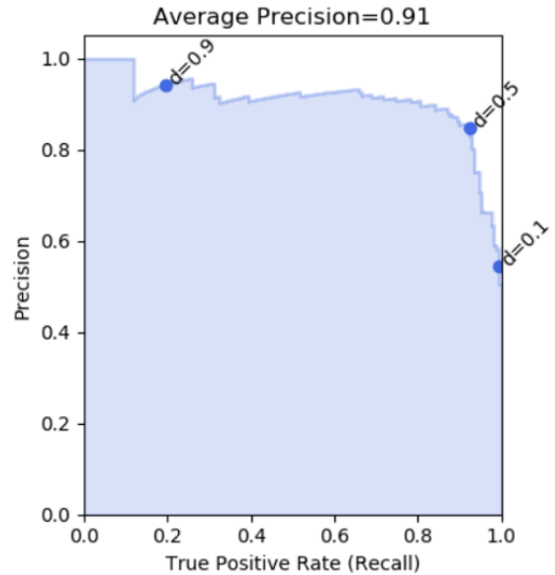


Figure 2.2: Precision-Recall curve. Area underneath the curve is the average precision. Figure reproduced from: <https://towardsdatascience.com/the-complete-guide-to-auc-and-average-precision-cf1d4647efc3>

The Average Precision (AP) takes the area underneath the PR Curve, and thus gives a measure for the model’s performance which is independent of the specific confidence threshold that a user may choose to use. Whilst this thesis deals with binary problems (two classes, positive or negative), the more general measure is the mean Average Precision (mAP). To calculate mAP, one calculates the AP for each individual class, and then takes the mean value of these APs.

In the case of object detection, a question must be considered before any of the above metrics are calculated. What defines a match between a prediction and a label? Clearly, the bounding boxes cannot be expected to be identical. Similarly, if there is some non-zero intersection between a prediction and a label, that does not necessarily mean it is a good enough fit to be counted. The most common method for matching predictions with labels is the Intersection over Union (IoU). The IoU gives a measure—between 0 and 1—of the similarity between two polygons.

For object detection, we define our metrics *at* a specific IoU threshold. This IoU threshold can be used to then apportion the detections and labels to the categories of True Positive (IoU>50% between label and prediction), False Positive (IoU<50% between prediction and all labels), and False Negative (IoU<50% between label and all predictions). One of the most common all-round object detection metrics is the “mean

Average Precision at an IoU threshold of 50%” (mAP@50%), which is used for judging performance in the Pattern Analysis, Statistical Modelling and Computational Learning - Visual Object Categories (PASCAL VOC) challenge [79], for example.

### 2.2.8 Other Machine Learning Techniques

#### Random Forests

Alongside neural networks, other predictive methods exist within machine learning. Random forests [80] are one such predictive model. Random forests are a collection of decision trees. Each decision tree is trained on a small amount of the total available training data, and is what’s called a “weak learner”, in that on its own, a decision tree cannot make particularly good predictions. When used in an ensemble (a random forest), however, where the predictions of many decision trees are averaged, the quality of predictions is much greater.

As input, a decision tree is given a 1-dimensional vector, in which each element corresponds to the value of some feature. These features could be the different spectral bands’ reflectance values in a pixel, for example. To construct a decision tree, one selects a feature at random from the input space. Then, a threshold is defined on that variable such that it maximises the information gain across it (as in, it splits the different categories of data as cleanly as possible, minimising the number of instances in the dataset which are on the wrong side of the threshold). These thresholds are found on randomly selected variables until all instances in the dataset can be correctly categorised, or until some maximum number of decisions has been reached in the tree.

When decision trees create many thresholds, they have a tendency to overfit the data. For this reason, each decision tree is trained only on a small sample of the training data (known as bootstrapped aggregating, or “bagging”). This bagged data creates many decision trees which have learnt to overfit on small, overlapping subsets of the dataset respectively, and when their predictions are pooled they become more robust. Random forests are incredibly quick to train and run, making them ideal for the semi-automated annotation conducted in Chapter 4.

#### Clustering

A key topic in *unsupervised* machine learning is clustering. This is the process of discovering categories (clusters) in a set of input points, without prior knowledge of which



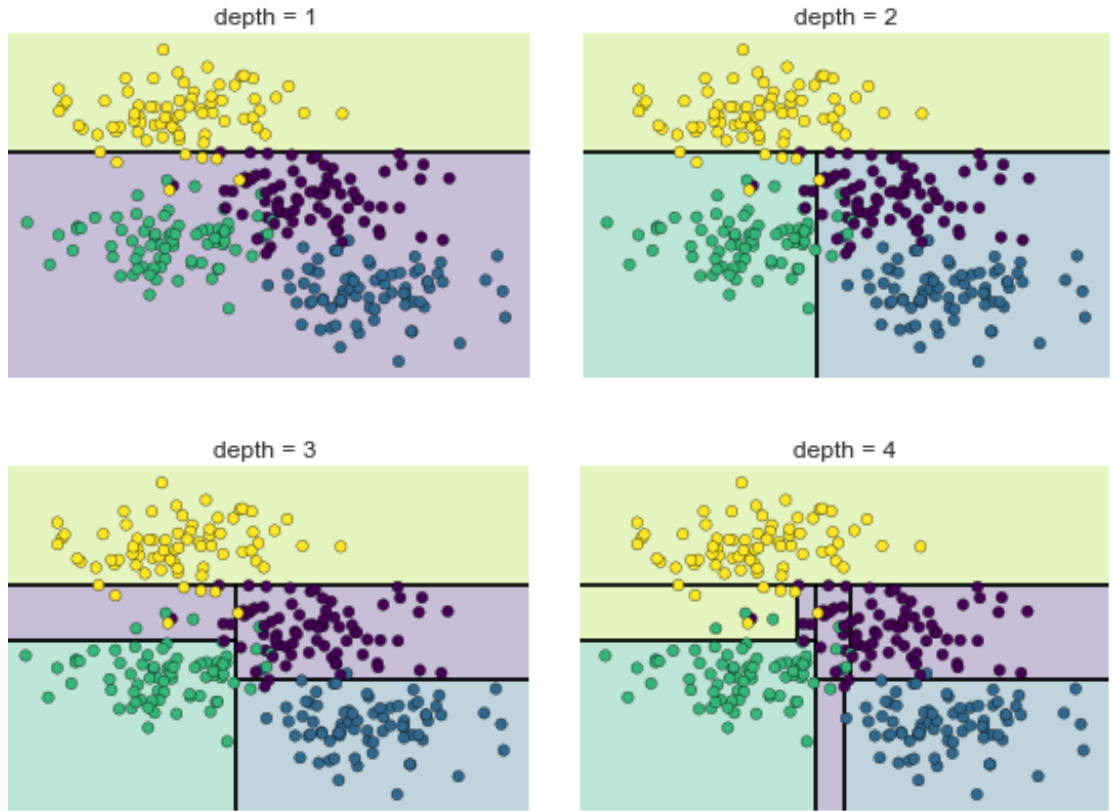


Figure 2.3: An illustrative example of a decision tree, learning to classify data in a 2-dimensional input space, across 4 classes. The tree creates more and more boundaries in the two input dimensions which minimise the number of points incorrectly classified. In most applications, the input space has many more dimensions, but is most easily visualised with two. Credit: <https://jakevdp.github.io/PythonDataScienceHandbook/06.00-figure-code.html>

categories any of those input points belong to. Some algorithms, like k-means, are explicitly given the number of clusters that exist, but given that clustering is applied to crater detection in Chapters 6 and 7, and the number of craters in an image is not known ahead of time, this section looks only at algorithms for which there is an undefined number of clusters.

[81] proposed the mean shift clustering method. The mean shift algorithm assumes that points are randomly sampled from some probability density function around the centres of several clusters. Most often, this probability density function (or kernel) is modelled as a Gaussian distribution. The expectation value of a cluster is then the sum of probability densities for the points in its vicinity, where points far from the centre have a low expectation value, and points close to the centre have a high expectation value. By maximising this expectation value, the most likely cluster centres are determined. Restarting the algorithm with different starting locations finds different cluster centres. By varying the standard deviation of the Gaussian probability kernel, the size (and thus number) of clusters can be altered.

Mean shift clustering has the advantage of being able to model arbitrary numbers of clusters, and can be tuned with just one parameter: the size of the kernel. However, it is highly sensitive to the kernel size, and clusters can be incorrectly merged or split if it is set badly.

Agglomerative clustering (e.g. [82]) is a kind of hierarchical clustering approach. First, a distance function must be defined, which is commonly Euclidean distance, but may vary based on the application (as in Chapter 7). Then, the distances between all points are calculated, giving a matrix of distances between each pair of points. A *dendrogram* is constructed by first merging the two closest points into a cluster, and then updating the other points' distances to it so that they are taken from the mean location of that new cluster. This is repeated until all points are in a single cluster. By defining a maximum distance that can be clustered, it is then possible to select the desired solution, which gives a reasonable number of clusters, dependent on the application.

## 2.3 Cloud Masking

*This section is adapted from the related work sections of [1], and a paper currently under review.*

At any given time, around two thirds of the Earth is obscured by clouds [83]. Hence, optical satellite instruments must contend with substantial cloud cover, obscuring the planet's surface. For applications focused on surface processes—e.g. vegetation monitoring [84], or surveillance [85]—clouds must be accounted for and removed. Meanwhile, for applications that are directly related to atmospheric processes, such as weather monitoring, cloud pixels must be retrieved. Crucially, both removal and retrieval of clouds require pixel-scale segmentation of a satellite image into cloudy vs. clear regions. The large size of high-resolution satellite images, as well as the tedious nature of pixel-scale manual annotations, have been strong motivators for the development of fully automatic algorithms which aim to accurately and reliably detect clouds in satellite imagery.

Automatic cloud detection is not a straightforward problem. The expected accuracy is high, whilst oftentimes the ground-truth is ambiguous (especially at the boundaries of clouds [86]). As well as high accuracy, a successful algorithm must have low enough computational cost to be applied across large datasets, be robust over different terrains, and have compatibility across different instruments and resolutions. Single-pixel techniques provide straightforward solutions which, if they use machine learning, are easy to implement as fully differentiable models (differentiable models are ones for which all

optimisation and parameter-tuning is governed by one loss function, as opposed to having modular subsections of an algorithm which are optimised independently). However, these single-pixel methods do not utilise relationships between nearby pixels. Techniques which utilise high-level features can use spatial correlations to arrive at nuanced predictions when pixel-level information is not enough, but—as we will see in Section 2.3.2—they often use separate preprocessing stages, which are not optimised at training and can increase computational complexity. These approaches appear to be complementary, in that they achieve overlapping but non-identical parts of the set of required properties. Therefore, an approach which carefully combines pixel-level with spatial features in a single differentiable architecture could have the beneficial properties of rich multi-scale features, without the need for optimising different stages in isolation.

Spectral bands offer different and complementary information with which to detect cloud. For example, visible bands like blue, green and red exhibit high albedo on cloudy regions, which can be used to distinguish them from surface features which often have lower albedo in some or all of these bands [87]. However, some surface features exhibit high albedo in all visible bands (e.g. roofs or exposed rocks), and so bands such as SWIR can be used [88] as they are very sensitive to surface water and water vapour [89]. SWIR bands can also be helpful in distinguishing icy terrain from cloud [90], which is an issue that has been studied extensively [91]. Similarly, thermal bands—whilst lacking the high resolution of other bands (see Table 2.2)—can measure an object’s temperature, which for clouds is generally much lower than unobstructed surface regions.

Relevant cloud detection techniques can be divided into two main classes: thresholding approaches derived from physical or empirical observations, and machine learning-based approaches which rely on statistical reasoning. Thresholding algorithms have a set of rules based on the relative values of various combinations of bands, in order to evaluate whether a given pixel is clear or cloudy. These methods are often informed by physical parameters such as reflectance and/or top-of-atmosphere temperature (e.g. [92], [93], [94]). On the other hand, machine learning techniques take these spectral bands (or some function of them) as inputs, and through some optimization process derive a model that maps the input space onto the desired classes, with little to no dependence on the underlying physics.

### 2.3.1 Thresholding Methods

In the early 1980s the need for automatic cloud masking was recognised, due to ever increasing quantities of Earth observation data [95]. It was found by Rossow et al. [96]

that a thresholding technique based on radiative transfer analysis was significantly better when compared with other thresholding techniques based on clear-sky radiances. The radiative transfer method was successful because it computed two successive thresholding operations, in both visible and IR bands, making it more robust than techniques which relied on only one threshold. Since then, thresholding techniques have often taken the form of decision trees in which *if-then* nodes are computed using relational operators, on band values or derived physical parameters like temperature (e.g. [92,97–99]). Some algorithms then use post-processing techniques to refine the results. For example, Zhu et al. [93,100] use the known illumination conditions of the scene to refine their estimates through geometric measures of similarity between detected clouds and shadows. Other techniques focus on classification of particular subcategories of cloud, such as cirrus clouds [101,102], or cumulonimbus clouds [103]. Spatial correlations between neighbouring pixels can also be used to improve classification accuracy and smooth the detected cloud boundaries, as exemplified by Hughes et al. [104]. Recently, [105] classified cloud and clear pixels adjacent to cloud, motivated by the importance of performance on the cloud boundaries.

As noted by Foga et al. [106], the physical basis behind thresholding methods means that they can be straightforwardly transferred between different instruments and datasets, assuming a similar resolution and spectral range. Another advantage of thresholding techniques is that no training dataset is required, which is time-consuming to produce. However, their simplicity is a significant limitation on their performance, and such approaches tend to fall short of techniques which embrace some kind of optimisation through labelled training data [107].

### 2.3.2 Machine Learning Methods

Machine learning algorithms differ fundamentally from “physical” cloud masking algorithms, in that they generate the function that maps input data to prediction by using statistical analysis of a training set. The number and arrangement of trainable parameters within the employed model dictates the space of possible solutions, whilst the training set used dictates the values of those weights and thus the specific solution found within that space [108]. Therefore, the variations seen in performance between cloud masking methods are affected by both the model architecture and training data.

Prior to the recent emergence of deep learning, most machine learning-based approaches focused on single pixels. Methods using individual pixel values as inputs to classification algorithms are well studied: with classifiers such as Support Vector Machines (SVM) [109], Principal Component Analysis (PCA) [110] and classical Bayesian

methods [102] being frequently tested with varying levels of success. Single-pixel neural networks are also common among cloud masking algorithms (e.g. [104, 111, 112]). These methods benefit from their computational speed, because more complex relationships between neighbouring pixel values are not calculated. However, the spatial correlations that these single-pixel techniques ignore are important in most natural images, including remote sensing data. In essence, these single-pixel level techniques explore an input space of limited dimensionality (e.g. the number of spectral bands of the instrument), and cannot express functions radically different from those of physically-derived thresholding techniques.

Expanding the dimensionality of the input space by including multiple pixels introduces the possibility for more complex relationships than those which can be mapped from a single pixel, and can be expected to increase potential performance. However, the implemented model must be capable of extracting meaningful information from this larger input space. Several methods for extracting features from larger regions exist; single-band textural features have been used as inputs to classifiers [111], or in conjunction with pixel-level features [113]. As discussed in Section 2.2.3, CNNs have had widespread success in image processing applications due to their ability to efficiently learn features across an extended scene (e.g. [5]).

If pixel-wise segmentation is not desired, and area-wise classification is acceptable, then standard classification CNNs can be employed to classify square regions of an image, as in Shendryk et al. [114]. Another method for approximating cloud segmentation as a classification task is through superpixel construction. Superpixels can be used to break scenes up into a set of regions that contain similar pixel values [115]. This allows regions to be treated like discrete objects, to be classified by a CNN [116, 117] or bespoke architectures like PCANet [118] as cloud or non-cloud. However, this can cause issues, as clouds are very often amorphous and merge with one another, meaning the performance is highly sensitive to the initial superpixel construction itself, which cannot be updated easily during training.

Ultimately, feature transforms like superpixel construction and textural transforms such as Gabor filters, are all non-differentiable pre-processing stages, and hence cannot be optimised for the specific task of cloud detection. This means they are unlikely to create the most suitable inputs for the vast majority of machine learning tools (e.g. a CNN, neural network, SVM or Random Forests).

Recently, Fully Convolutional Networks (FCNs) [119] have shown promise in image segmentation problems across a variety of fields (e.g. U-Net in biomedical imaging [58] and SegNet for natural image segmentation [120]), by outputting pixel-wise predictions

over extended scenes. They do this by applying both convolutions and their inverse—transpose convolutions—to learn a direct transformation between image and outputted mask. This offers a framework for a model that successfully fuses pixel-level and spatial features. A recent example of this can be found in Mohajerani et al. [121], which used an FCN for cloud masking. Their algorithm includes a pre-processing thresholding step for snow/ice retrieval, and uses an existing Landsat 8 cloud mask as a prior to improve performance. In a similar vein, [122] use rescaling to combine features from different depths in a U-Net.

Some deep learning approaches have demonstrated interoperability on multiple satellite sensors without retraining [123]. Often, they use the similarities between sensors to train models on specific, shared spectral band combinations. This approach still provides a model which is not sensor independent, in that it cannot then be used on any arbitrary set of spectral bands. This has two drawbacks, (i) the number of sensors a model can work with is limited to only those with all of the selected bands, and (ii) the information from other spectral bands not included in the specific combination of selected bands is discarded. Shendryk et al. [114] use PlanetScope data, and Sentinel-2 data restricted to the RGB and NIR bands, to simulate the same spectral bands as PlanetScope, allowing for a model which is interoperable with 4-band RGB+NIR satellite data. Similarly, Li et al. [124] shows a single model working across RGB images taken from 8 different sensors and a range of resolutions, from 4–50 m. Wieland et al. [125] notes that not only are RGB and NIR bands shared, but also that the two SWIR bands across Sentinel-2, and Landsat 5, 7, and 8 are similar, using this to train a 6-band model shared across the four sensors.

In contrast to these methods, which exploit a specific combination of bands shared across sensors, Chapter 5 proposes *sensor independent* models which work across different sensors without selecting a specific spectral band combination, and without retraining. The bands that a sensor independent model can use are still limited to those available to the model during training, but any combination of those bands can be taken as input. This affords greater flexibility in sensor choice, more training data, and a more comprehensive leveraging of the available information from the spectral bands those sensors’ data include, by not throwing away those bands which are not shared.

Regardless of the architecture, supervised machine learning approaches require sufficient training data in order to achieve maximum performance. The community has in recent years made several annotated datasets publicly available, such as the SPARCS dataset [126], comprising 80 1000-by-1000 pixel Landsat 8 patches, or the Landsat 8 Cloud Cover Assessment (CCA) dataset [106], comprising 96 roughly 7000-by-7000 pixel

Landsat 8 patches. For single-pixel approaches, [102] released a dataset of labelled spectra for Sentinel-2. With these datasets, among others, the potential for machine learning techniques in cloud detection can be explored more fully than was previously possible. A more comprehensive discussion on the current state of cloud masking datasets can be found in Chapter 4, and in Sections 3.4.1 and 5.2.4.

## 2.4 Craters on Mars

Craters are a ubiquitous feature on the surfaces of many terrestrial bodies in the solar system. Earth, the Moon, Mars, Mercury, as well as many of the Jovian and Saturnian moons, are covered in numerous craters ranging hugely in size. These craters can relay information about the body’s history and past conditions. Primarily, crater populations are used as a proxy for a surface’s age. This is because the longer a surface is exposed, the more meteoritic impacts it is exposed to. This can be used to make relative age estimates, however, absolute aging requires a detailed knowledge of both historic impactor rates and erosion rates. With both of these functions, we can infer the time taken for a given crater population to be made [127].

Impactor rates come from a fusion of solar system evolution modelling, and empirical observations of current meteoritic fluxes, and are seen to vary between the inner and outer solar systems [128]. The production rate of craters is observed as an inverse power-law between crater size and frequency [129]. This is due to the relative scarcity of large impactors in the inner Solar System. The result on the surfaces of planets is that the densities of smaller craters is several orders of magnitude larger than for bigger craters.

Erosion rates are less well understood. Unlike impactor flux, erosion rates are dictated by both exogenous and endogenous forces. The primary exogenous force is the resurfacing effect of new impacts, happily this has a direct correlation with the production rate at any given time, and—although stochastic and localised—can be easily identified and corrected for on the surface. For example, a large crater can be assumed to have covered whatever population of smaller craters was there before, and thus this area can be omitted in the calculation of crater density for sizes below the large crater [130]. Endogenous sources of erosion, however, are far less straightforward to account for. Complex processes such as impact of the solar wind, galactic cosmic radiation, thermal day/night cycles at the microscopic level, dust deposition, aeolian and fluvial erosion, alongside volcanism and (in the case of Earth) biological resurfacing, are only constrained by our somewhat speculative models of the body’s past climatic and volcanic history. It should be noted that these macroscopic erosion mechanisms are largely absent on the Moon and other

airless bodies. On Mars, significant disagreement exists regarding past climatic history, e.g. [131, 132].

On Mars, we have now been able to age a significant portion of the surface using Crater Size-Frequency Distributions (CSFDs) [133, 134]. As more analysis is done, a complex planetary history has emerged, with areas that underwent volcanic resurfacing events over the last ten million years (e.g. Elysium Planitia [135]), alongside ancient Noachian surfaces preserved for 3.9 billion years. This event is known as the Late Heavy Bombardment (LHB) and is thought to have caused dramatic resurfacing events across all inner solar system bodies, including the Moon and Mars [136]. The LHB prevents us from age-dating surfaces before it, given the overwhelming number of large impacts seen between 3.9-3.8 Ga [128]. The LHB is characterised by disproportionately high numbers of large impactors. Several theories have been proposed to understand how the LHB came about, and why it seems to have produced extraordinarily large craters [137–139].

So, around 4 Ga is the upper bound for age-dating studies, but there is also a lower bound. Conversely to the upper bound, which is the result of an overwhelming number of craters, the lower limit is dictated by a lack of crater measurements from which to draw our statistics. To be specific, the large craters that are normally used to date a surface are not present. We do, however, see a significant population of small craters on young surfaces, due to the inverse power law of the production rate with respect to size. As discussed previously, the erosion rates that effect these small craters are badly constrained, and conspire to make age-dating of these young surfaces imprecise [140]. Minton et al. [141] found small craters to be in equilibrium on the Lunar mare, with erosion and production processes balanced, and found through modelling that distal ejecta from new impacts was the primary cause for the obliteration of craters on the Moon.

## 2.5 Isidis Planitia

The work in Chapter 9 seeks to isolate some of the complex factors affecting small crater populations. It has long been understood that the size, shape and longevity of impact craters are dependent upon the properties of the surface on which they are made [140]. Therefore, in order to simplify the interpretation of crater detection results, it is desirable to select a site over which there is a relatively consistent geological structure, relief, and age. Isidis Planitia is an excellent candidate, given that it is a large, flat area, and has few geomorphological features other than the impact craters being studied.



One exception to this is the cone structures, found in curvilinear ridges, across Isidis [142]. These are thought to be a result of either mud volcanism [142], or pyroclastic surges [143]. Whilst the provenance of these features is beyond the scope of this work, their size distribution is within that of the crater detections used in Chapter 9 (mostly between 200-1000 m diameter [143]), and given that they look very similar to craters, it is likely that at least some will be counted falsely as craters.

Isidis Planitia is a large ancient impact basin, which formed early in Mars' history. Since then, there have been three major episodes in the history of Isidis [144]. First, an era characterised by large and frequent impactors, up to 3.8Ga. Second, an episode of extensive reworking through volcanism laying large deposits of volcanic materials over Isidis, and then fluvial erosion of those materials, lasting between 3.8-2.8 Ga. Finally, since then, Isidis seems to have been reworked through wind erosion and dust deposition, further modifying the surface until the present day.

At the edges of the basin, the heavily eroded slopes of the original impact crater form complex, undulating terrain, at the boundary between the higher altitude volcanic regions of areas such as Syrtis Major and the low altitude basin. To the north east, the edge of the basin reaches out and connects to Utopia Planitia, which forms part of the extensive lowlands over Mars' northern hemisphere. The central basin, however, is almost entirely flat.

## 2.6 Crater Detection

Crater detection is perhaps the most obvious application for object detection algorithms in planetary science, and yet successful implementation of a Crater Detection Algorithm (CDA) continues to elude the community. To date, no fully automatic system has ever been used to conduct a trusted crater survey [145]. This section will provide an overview of the important developments in crater detection. The primary focus of this section will be on CDAs that use visible imagery, rather than digital terrain models, given that for smaller craters (which we are more interested in detecting) the higher-resolution of visible imagery is favourable.

### 2.6.1 Template-Matching Methods

An important early development in crater detection was the development of shadow-highlight matching algorithms [146]. By progressively changing the contrast of both the original image and an inverted counterpart, both shadows and highlights could be found.

Then, using random forests [147], they were paired and in this way candidate craters were located. This method has several advantages: it is very computationally efficient, it finds any crater with the tell-tale shadow-highlight pair, and when it finds a crater it approximates its size and position very accurately. However, many non-crater features on the surface share this shadow-highlight relationship. Therefore, given its high Recall and low computational requirements, this step became a candidate-generation stage in many of the ensuing algorithms.

One such algorithm uses shape and texture filtering to further discern between craters and non-craters [148]. By taking the candidates generated with [146] shadow-highlight algorithm, and extracting texture features as was done for face detection [149], they improved performance dramatically. Several iterative improvements to this approach were made in the following years, including using a boosting approach, in which features that are seen to be good predictors of craters are given higher weightings, and thus a more reliable prediction can be made [150].

## 2.6.2 Machine Learning Methods

In 2016, the most successful classification stage used with the shadow-highlight candidate-generation was proposed. Using a CNN, Cohen et al. [151] increased the state-of-the-art performance dramatically once again, with an  $F_1$  of 89%. However, at this point, the majority of missed craters were now those which had not propagated through the candidate-generation stage, meaning the limiting factor on performance is now no longer the classification, but the pre-selection. Now, given the success of CNN-based architectures in other applications of object detection, it seems natural to seek to abandon the traditional image processing techniques that are currently used to generate candidate craters, and instead use purely convolutional techniques.

To this end, [152] proposed a deep learning technique based on image segmentation. The U-Net model is tasked with returning a map of where crater edges are, hopefully giving well defined rings around each crater. A simple template-matching method is then used to extract the locations and sizes of the craters. In their study, the global Martian crater catalogue [11] is used for training and testing, with craters between 2–32 km diameter. More recently, [153] also used crater edge segmentation, and additionally trained a random forest to classify the areas bounding those detected edges, rather than just using a template-matching method. This study used images taken from Google Mars, but does not detail the size of craters that were detected within those images.

## 3 | CloudFCN

*This chapter is based heavily on a published paper [1]. The design, development and testing of CloudFCN were all carried out by myself. Annotations of the Carbonite-2 data were courtesy of Cortexica Ltd and Earth-i Ltd. The co-authors of [1] provided oversight and help with managing the project. The chapter comprises the paper’s technical sections, with some minor rewording.*

### 3.1 Motivation

This chapter introduces a cloud detection pipeline that incorporates both low-level colour features and high-level spatial descriptors. To do this a bespoke transforming autoencoder architecture is used, inspired by the U-Net algorithm [58], which has recently demonstrated good performance in a variety of image segmentation tasks (e.g. [59, 60]). The residual connections within the model allow for fusion between the low-level information of individual pixels and high-level convolutional features from successively larger fields of view. Meanwhile, the weights within the model’s layers determine the specific features extracted, and their relative importance when fused, and are learned during a supervised training stage. Non-trainable stages (e.g. hard-coded preprocessing pipelines) are avoided by giving the model the pixel values themselves as input, and outputting the final pixel-wise cloud masks at the end.

The design of this model is guided by theoretical considerations and intuitive reasoning based on the specific problem of cloud masking. More specifically, the concept of a ‘receptive field’ is used as a rationale for network size, which is very often a parameter simply tuned based on computational limitations rather than domain-specific knowledge. Similarly, the use of InceptionNet-style modules [154] within the network is informed by the observation that cloud edges are often ambiguous, and thus for a model to produce smooth outputted edges, local information sharing should be encouraged between neighbouring pixels. Finally, the class-weighting scheme provides both stable performance on this problem, but also to other segmentation problems in remote sensing, in which class populations are very often imbalanced.

Aside from the development of a high-performance algorithm, the main contributions of this study are the principles employed in designing, training and testing the algorithm.

Specifically, our reasoning about the network’s receptive field as a key design driver, and the class weighting approach taken (which are both applicable to image segmentation tasks in general). This study is a more complete and thorough version of preliminary work presented at an ESA conference [155], adding an extensive set of experiments on datasets totalling over 700 scenes and around 20 billion pixels. These are taken from multiple instruments, resolutions, geographical locations and terrain types, and are tested with several evaluation metrics, each focusing on a specific use case or performance criteria. Pixel-wise evaluation metrics (accuracy, omission and commission rates) are used to compare performance with other cloud detection algorithms on well known datasets. Meanwhile, scene-level coverage accuracy (in which the total estimated cloud cover over an image is measured) is used to evaluate it’s potential for on-board use. In addition, simulated noise experiments infer the algorithm’s performance on instruments with a range of Signal-to-Noise Ratios (SNRs) and bit-depths, whilst also shedding light on how the model uses textural and pixel-level features to mask clouds.

Sensor independence is not considered explicitly in this chapter. Given that this work was done before the creation of our Sentinel-2 dataset (Chapter 4) the focus is instead on the performance on individual satellites—primarily Landsat 8. Later, in Chapter 5, sensor independence will be tackled, which will also involve the CloudFCN model developed here.

## 3.2 Methods

### 3.2.1 Overview

As mentioned in Section 2.3.2, pixel-level data necessarily has fewer dimensions than multi-pixel windows, which can make reliable detection of cloud more difficult. This is exacerbated in RGB when compared to multispectral data, because high albedo terrains (bright urban areas, for example) look the same in RGB as cloud pixels do, when they are more separable in NIR and TIR bands. Although individual pixels can be misleading over certain terrains, texture often varies greatly between cloud and non-cloud. Therefore, if the system is to perform robustly in high albedo terrains, we need to extract spatial features around each pixel, as well as the individual pixel values themselves, and combine their information content to help distinguish cloud and non-cloud by their texture and surrounding context. Fusion of these different features is vital, because using only high-level features may make it difficult to learn simpler relationships based on colour and brightness. Using an FCN with residual connections in the style of U-Net

(see Section 2.2.4) achieves these aims, whilst allowing the entire model to be optimised end-to-end during the training phase.

As input, CloudFCN takes extended scenes from satellite imagery with arbitrary spatial dimensions. For each given spectral band combination (e.g. RGB) a different instance of the model is needed, however the only alteration made to the design is the number of input channels—no internal parameters are altered. The output of the model has the same spatial size as the input, but can take two distinct formats. When a pixel-by-pixel mask is desired, the final layer uses a softmax activation [54] to classify each pixel as Clear or Cloudy. This is advantageous because it can be easily extended to multi-class problems (e.g. Clear vs. Cloud vs. Cloud Shadow). However, when cloud coverage estimation (as in the percentage of cloud cover over a whole scene) is the desired final product, the output is a mask where each pixel has a value between 0 (clear) and 1 (thick cloud), and the cloudiness of a pixel is treated as a regression problem. An average is then taken over the whole scene for the cloud coverage estimation. This format allows the model to make estimates for thin cloud as well as thick, and results in more accurate percentages through averaging.

The model comprises two components, the *encoder* and the *decoder* (Fig. 3.1). The convolutional structure of these components is described further in Section 3.2.2. The encoder serves to extract spatial features from the scene, reducing the spatial dimensions whilst increasing the number of channels. Meanwhile, the decoder takes these features and reprojects them to create an output mask. Residual connections (also described in Section 3.2.2) link intermediate points in the encoder and decoder, to allow for fusion between low- and high-level features.

The proposed model has several desirable traits, which alleviate the issues that other methods exhibit. First, our design allows for flexible input formats, having no specific spectral requirements, and is able to ingest input images of different sizes. Second, it combines the simplest brightness and colour information with more abstract and complex features from the surrounding areas, creating a complementary set of features that exhibit the strengths of both pixel-level techniques and convolutional ones. Lastly, the output format of the model, and attendant loss function, can be selected based on user needs, increasing the number of possible applications for the method, from cloud coverage estimation (in which a Root Mean Square Error—RMSE loss is used) to pixel-wise masking (with a categorical-crossentropy loss)—see Section 2.2.2 for further details on these loss functions.

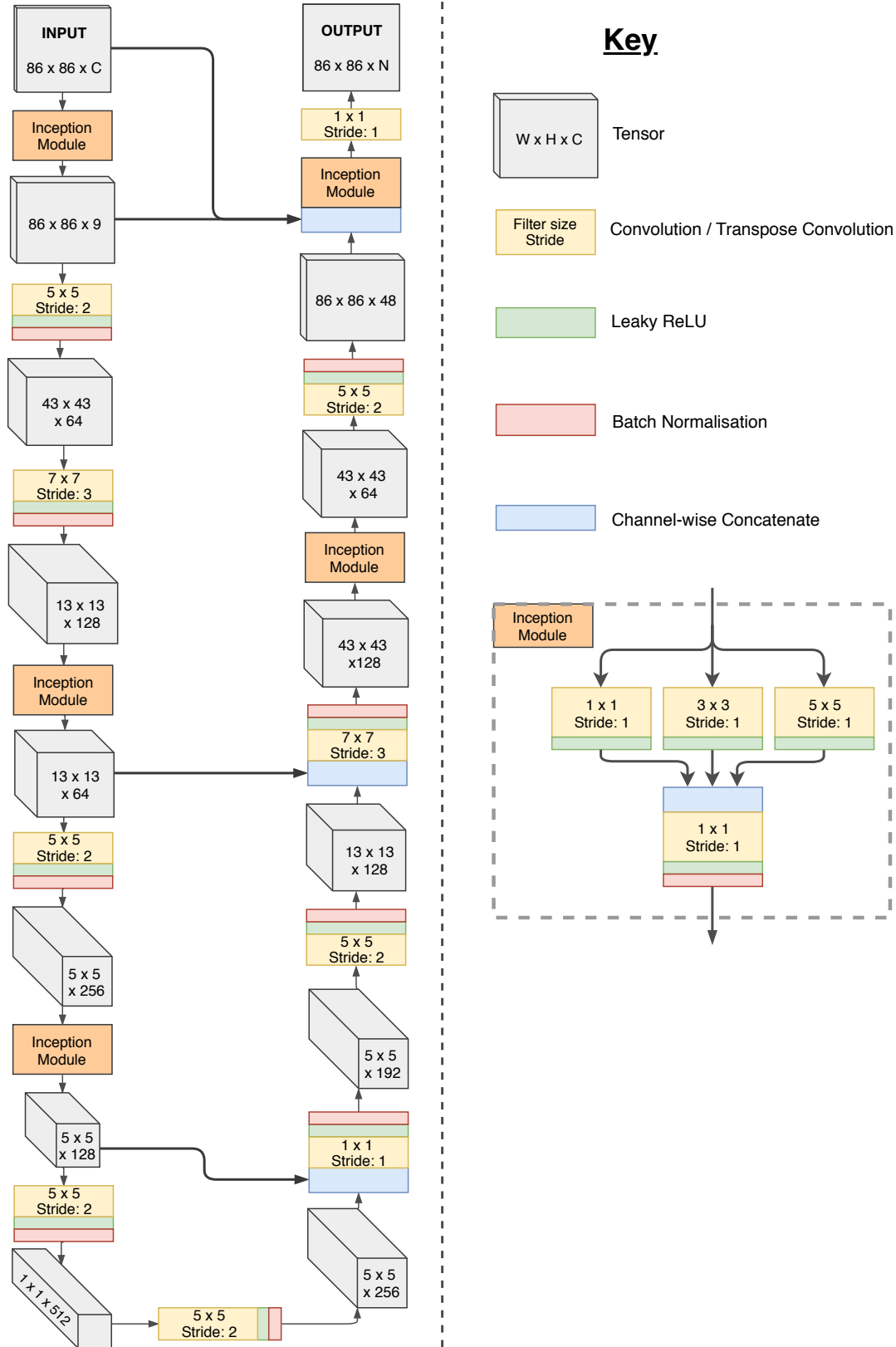


Figure 3.1: Flowchart of CNN used in cloud segmentation. Bold arrows represent residual connections, connecting different stages of the encoder and decoder arms of the model which run in a U shape. ‘Inception module’ denotes a set of parallel convolutions of stride 1, followed by a single dimensionality-reduction convolution on their concatenated outputs. Widths and heights of inputs and outputs are the *minimum* and could be any integer multiple of 24 above 86, with internal tensor dimensions changing accordingly. The first input layer is given  $C$  channels, as a placeholder for whatever kind of input being used (e.g. 3 for RGB). The output has channel depth  $N$ , to denote the number of output classes used.

### 3.2.2 Model Features

The convolutional layers of the model are designed to extract translation-invariant features across the image. Each filter within a layer is convolved with the input, producing a response map of where the filter pattern is similar to locations in the image. The advantage of convolutional layers is their shared weights—by using the same filters across the whole image, the number of weights is reduced, whilst enforcing translational invariance of the learnt features. Additionally, using exclusively convolutional layers is what allows us to create models that work on arbitrary image sizes, because no layer requires a specific input size, unlike fully connected layers commonly seen in classification networks. By using convolutions of stride greater than one, the layers’ outputs are spatially downsampled, leading to more compact feature map representations.

Between the standard convolutional layers, which reduce the spatial dimensions of the data whilst increasing the channel depth, there are InceptionNet-style modules [154] with several parallel convolutional layers of stride one, all with different filter sizes. These layers preserve the spatial dimensions of the feature map and fuse information from the surrounding area at each point (Fig. 3.1). As well as allowing for connections between neighbouring points, these modules also reduce the channel depth of the feature map, which increases the computational efficiency and reduces memory requirements. It was also found that using these layers (particularly the one connected closest to the output) reduced the amount of high frequency checkerboard artefacts in our output. These are seen commonly in transforming encoders, and are a result of uneven responses from the final transpose convolution layer [156], which are still seen in regions with high uncertainty (e.g. 2<sup>nd</sup> row, Fig. 3.3).

Residual connections are employed in our model to allow for different feature levels to be fused at various points in the decoder arm. By fusing the features, this ensures that the final output is informed by both simple intensity relationships between different spectral bands in the individual pixel, as well as by a range of complex spatial features. The residual connection concatenates the features outputted by one of the encoder’s convolutions onto the inputs of one of the decoder’s transpose convolutions.

Before three of the decoder’s transpose convolutions, there exists a residual connection. This effectively means there are 4 nested transforming encoder routes (3 across the residual connections and the last through the entire encoder and decoder) which are all used to arrive at an output value for a given pixel. Each of the 4 transforming encoder routes provides information about a different field of view, the first being pixel-level, the second a small window around each pixel, the third a larger window, and the last having

the maximum field of view of the network. The field of view of a given layer is termed the *receptive field*, and is discussed in more detail in Section 3.3.1.

After each convolution in the network, there is a leaky ReLU activation function, which introduces non-linearities [157]. A leaky ReLU is similar to a standard ReLU layer, in that it has a gradient of 1 for inputs above zero, but, unlike the flat response of a ReLU for input values below zero, it has a small gradient (Eqn 3.1). This allows negative values to still have a gradient in back-propagation, preventing them from getting ‘stuck’ at some value,

$$y = \begin{cases} x, & \text{if } x \geq 0 \\ ax, & \text{otherwise} \end{cases}, \quad \text{where } 0 < a \ll 1 \quad (3.1)$$

Batch normalisation layers [51] are used after each convolution or inception module. These rescale the response of a layer to have a mean of zero and a standard deviation of 1. Batch normalisation has several advantages. Primarily, it acts to regularise the network, thus reducing overfitting, which is especially important when using relatively small datasets like those available in cloud masking. It also helps to increase training speed, as gradients are re-scaled at each layer during back-propagation.

Max pooling layers [54] were not used in the model design. During max pooling, the feature map is downsampled by selecting the maximum value in each channel within a given window size, (typically 2-by-2, leading to a downsampling rate of 2). By doing so, one condenses the feature map and preserves the most important features, but information about where exactly those feature are present is lost. For classification tasks, this is not so important, however when the output is a pixel-wise mask, this loss of localisation could lead to inaccuracies in the output.

The number of layers within the model and their geometries were arrived at heuristically, by considering both the computational efficiency at inference and training time, as well as the model’s performance. With a larger number of channels in each layer, the complexity of information that the model is able to describe increases. However, at arbitrarily high channel depth, the information content is limited by the available dataset and training time, rather than by the model’s architecture. Therefore, it was first determined what would be a sensible number of layers (described further in Section 3.3.1). Then, the channel depth of each layer was selected by starting with a small number of weights, and gradually increasing them until performance was at a plateau.



### 3.2.3 Training

During training, the dataset was split into training and validation sets. training separate models per land cover type was avoided, in order to develop robust models that are invariant to terrain type. Therefore, both training and validation contains a representative sample from each terrain type. Although arbitrarily sized images can be ingested by the model, it is trained with image patches of a fixed size to allow for simpler implementation. Tests on the effect of cropped image size were inconclusive, suggesting that anything larger than around 150 pixels had similar performance, whilst crop sizes at the minimum (86 pixels) made performance suffer slightly. At larger sizes (roughly over 500 pixels), the memory requirements for back-propagation over large batches became unfeasible on our 6GB Graphical Processing Unit (GPU). Therefore, the models are mostly trained on patches of a few hundred pixels across. Importantly, although a training sample is a cropped patch from the original image, training and validation sets do not share parent images, in that they are split prior to any cropping operations.

Each batch contained 24 patches taken from the training set. The intensities are normalised such that across the entire dataset there is a mean=0 and standard deviation=1 in each spectral band. This is done so that—when training begins—the model’s layers do not receive very small or large gradients that could lead to unstable weight updates. Next, each image patch goes through several random transformations—listed below—in order to increase the variance seen in the training set, and to ensure the model is invariant to moderate noise and small changes in illumination condition. These transformations are done to each batch as training continues, rather than multiple augmented versions of each image being saved to disk.

1. **Rotation:** Rotated by a random integer multiple of 90°
2. **Flipping:** Flipped left-right with 50% probability
3. **Intensity shift:** All input values scaled by same random factor in range 0.9–1.1
4. **Chromatic shift:** Each input channel scaled by different random factor in range 0.95–1.05
5. **Salt and pepper noise:** Pixels set to hot (+3) or cold (-3) values, with per pixel probability of 0.005
6. **White noise:** Gaussian noise with sigma=0.05

The loss function is determined by the output format. Two cases exist, in the case where cloudiness is treated as a continuous variable where 0 is clear and 1 is thick cloud, then mean square error is used in order to train for pixel-wise regression. For pixel-wise classification, categorical cross-entropy is used as the loss function. This loss is used by the optimisation routine to update weights. Experiments were done with Stochastic Gradient Descent (SGD) [158], Adam [159], Adagrad [160] and Adadelata [56]. All three adaptive update methods outperformed SGD by similar amounts with regards to training time and final performance, so Adadelata was selected somewhat arbitrarily.

The initial learning rate of the Adadelata was set to 0.001, along with a decay rate of 0.95, which are their default values [56], given that no significant improvement was seen when tuning these parameters. Training was carried out until no notable improvement in performance on the validation set was seen over several epochs, where each training epoch was 1000 batches each of size 24. Generally, a few tens of epochs were used, taking under an hour on a desktop GPU (NVidia GTX 1060, 6GB) to reach a plateau. A sample of validation results were also visually displayed in each epoch, to gain a qualitative appreciation for the model's performance, and the characteristics of the output masks.

## 3.3 Theoretical Considerations

### 3.3.1 Receptive Field

The receptive field is defined as the region of inputs that can affect the value of a given node in the model, outside of which any variation in input would have no effect on that node's response. In designing the model architecture, a reasonable value for the receptive field of the output pixels was found intuitively. Ultimately, the optimal value is not exact, but lies in a range such that it is neither too small or too big. If it is too small, then the network will not have access to useful information about a pixel's surroundings, and if it is too big then the network's size becomes cumbersome and training time, computational power and dataset size limits performance. So, to find a sensible range, the ability of humans to recognise cloud in cropped Earth observation images was qualitatively assessed. At very small crop sizes (a few pixels) we as humans are often unable to discern between cloud pixels and other high-albedo regions, because contextual information about the scene is not given to us. At very large crop sizes, humans do not improve in their cloud masking predictions, because the limit of useful contextual information is reached. A field of view of a few hundred pixels was ample for humans to make predictions of cloud masks, based on our experience observing several

Landsat 8 scenes. Therefore, the model was designed such that the receptive field of its final convolutional layer was roughly 300-400 pixels across. Going beyond this value would lower the computational efficiency, making it difficult to process larger scenes quickly.

Receptive fields can be calculated recursively throughout the layers of the network, beginning at the input [161]. First, we consider the receptive field,  $r_{i+1}$ , of a neuron in  $layer_{i+1}$  given the geometry of  $layer_i$  (stride,  $s_i$  and kernel size,  $k_i$ ) (Eqn 3.2). In addition to the previous layer’s parameters, the jump,  $j_i$  is defined as the product of all strides up to that layer,

$$r_{i+1} = r_i + (k_i - 1) * j_i \quad (3.2)$$

$$j_{i+1} = j_i * s_i \quad (3.3)$$

and is used to keep track of the ‘distance’ in input space between two adjacent points in  $layer_i$  (Eqn 3.3).

Table 3.1 shows the receptive field and jump through successive layers in the network. Up until the *code layer*, which signifies the deepest part of the network (Fig. 3.1), the jump increases along with the receptive field. A neuron in the code layer of the network describes a feature that is sensitive to a 165-by-165 pixel area. The jump of 24 means that for every 24 pixels in the input image, one more feature vector is used in the code layer. After this point in the model the jump begins to decrease, whilst the receptive field continues to grow. This is because the stride of a transpose convolution has the inverse effect of a convolutional layer on the jump (transpose convolutions are also described in the description of U-Net in Section 2.2.4).

Although the total receptive field of an output pixel is 365, the outermost regions will only negligibly impact the model’s response. This is partly an inherent property of many-layered convolutions [162], but is in our case amplified by the use of residual connections. Each residual connection adds more importance to a central portion of the receptive field. For example, the first residual connection adds only pixel-level information, thus amplifying the importance of only the central pixel in the receptive field, whilst a residual connection deeper in the model will amplify the importance of a 21 or 69 pixel region at the centre of the full receptive field. This is a desired property, as it is clear that the importance of a feature 100 pixels away is less than that of a pixel in the output neuron’s proximity.

Layer	Layer Type	Stride	Filter Size	Jump	Receptive Field
1	Inception	1	5	1	1
2	Convolution	2	5	1	5
3	Convolution	3	7	2	9
4	Inception	1	5	6	21
5	Convolution	2	5	6	45
6	Inception	1	5	12	69
7	Convolution	2	5	12	117
8	Transpose	2	5	24	165
9	Convolution	1	1	12	261
10	Transpose	2	5	12	261
11	Transpose	3	7	6	309
12	Inception	1	5	2	345
13	Transpose	2	5	2	353
14	Inception	1	5	1	361
15	Convolution	1	1	1	365
16	Output	-	-	1	365

Table 3.1: Layer-by-layer calculation of receptive field through the network. This follows the encoder and decoder displayed in Fig. 3.1. The jump reaches its maximum before the first transpose convolution, where it then begins to decrease. Blue rows indicate that their outputs are sent through a residual connection, whilst red rows indicate the operation has a residual connection to its input.

### 3.3.2 Class weighting

Imbalanced class populations in a training set can lead to unwanted characteristics in the resulting model. In the extreme case, where one class represents a huge majority of training examples, the loss function drives the model to always predict the majority class, and never consider the rare minorities. Even when moderate class imbalances are present, the model can still develop strong biases against rarer classes. This *a priori* bias, although good for minimising loss on training data, may not be desirable in real-world use. For example, if OA is encouraged by the loss function, then the precision of a rare class will be prioritised over the recall, as the model will be more sceptical. For problems in which high recall is desired, this is an unwanted effect.

In many machine learning applications, imbalanced datasets can be addressed by preferentially sampling minority classes at training time. Alternatively, the loss associated with each class can be modified by a factor which prevents the model from becoming overly biased. In this work, the loss was modified by a factor that normalised for the relative abundances of the different classes. In the Landsat dataset used, cloudy and clear pixels took up relatively equal amounts of the data, so this was not as important.

However, when used for Carbonite-2 data, the number of cloudy pixels was too low and a strong bias against predicting cloud was found in the model. Therefore, loss weighting was applied to encourage the model to predict the presence of cloud.

A factor of  $1/abundance$  applied to the loss was too dramatic (where *abundance* is defined as the proportion of total pixels that are within that class), as for cloud this would mean a loss that was nearly an order of magnitude larger than for clear pixels, leading to unstable back-propagation. Instead, a factor of  $1/\sqrt{abundance}$  was used, which was a trade-off between balancing omission and commission errors and producing gradients small enough to lead to stable back-propagation. Formal experimentation on the amount of class-weighting was not conducted, because its purpose is not to fine-tune the omission and commission errors, but rather to ensure stable training that consistently finds a suitable local minimum. A more practical way to tweak the balance between omission and commission is to adjust the threshold confidence value at inference time, in order to optimise the performance for a given application.

## 3.4 Experimental Results

The performance of CloudFCN is measured in several experiments, with metrics that highlight different properties of the model. In Section 3.4.1, the two datasets used in this chapter’s experiments are described. Then, in Section 3.4.2 an experiment for cloud coverage estimation in RGB is conducted, which is relevant to possible on-board use of the algorithm. Next, in Section 3.4.3, pixel-wise cloud masking performance is measured on a Landsat 8 dataset, with only RGB bands taken from the 11-band instrument. Section 3.4.4 then tests the performance of the algorithm on full 11-band Landsat 8 images, and offers a comparison between several algorithms and our own. Finally, in Section 3.4.5, there is an analysis how noise affects the algorithm’s performance, by corrupting Landsat 8 data and re-evaluating the experiments performed previously. Evaluation of the algorithm’s performance in cloud shadow detection is omitted, because the available datasets did not have consistent shadow annotations (the Landsat 8 dataset does have some labelled cloud shadows, however it is not comprehensive, making training and validation on this class challenging).

### 3.4.1 Datasets

#### Carbonite-2

The Carbonite-2 satellite took high-resolution (80 cm per pixel) video in visible wavelengths. By gimbaling the platform as it orbits to achieve a “point-and-stare”, a continuous video over a single scene is taken, at a size of 5000-by-5000 pixels. The dataset comprises individual frames from several hundred videos from a variety of locations around the world (Figure 3.2). From each video used, 3 frames were selected randomly and marked both ‘thick’ and ‘thin’ cloud. ‘Thick’ cloud is defined as any cloud cover through which no surface features are visible, whilst ‘thin’ cloud is any cloud through which some surface content is visible.

In total, 1561 frames were annotated, however, many of these (698) contained large amounts of noise, and were unusable. This left 863 frames to be used in training and validation. The vast majority of frames selected were entirely clear, or entirely cloudy, as these were the simplest to annotate. However, 155 frames were manually annotated and contained both clear and cloudy conditions. The training set was made using all the mixed visibility frames along with some all cloud and all clear frames, totalling 185. For validation, tests on both all cloudy and all clear frames were conducted. 72 all cloud frames were left after making the training set, along with 606 all clear ones. This validation strategy focused on assessing the model’s reliability as a cloud coverage estimator, rather than as a masking algorithm. This means there was more interest in severe errors in which an all clear frame was marked as mostly cloudy, or vice versa. The analysis of pixel-wise predictions is left for the other dataset.

#### Landsat 8 CCA

The Landsat 8 CCA dataset [163] is a collection of level-1 Landsat 8 images. Comprising 96 individual scenes, this dataset was specifically designed for testing cloud masking performance over a variety of terrain types. In total, 8 categories of terrain are given, with 12 tiles each: Barren, Forest, Grass/Crops, Urban, Shrubland, Snow/Ice, Water, and Wetlands. The annotations have classes for clear, shadow, thin cloud and thick cloud. However, the creators note that shadow was inconsistently marked due to difficult topography, and it was too sparsely annotated to be useful in training. In addition, the distinction between thin and thick cloud classes is ambiguous and sometimes inconsistent, which could have caused issues for a model during training. Therefore, the original four class problem was condensed into a two class problem: Clear vs. cloud, in which clear

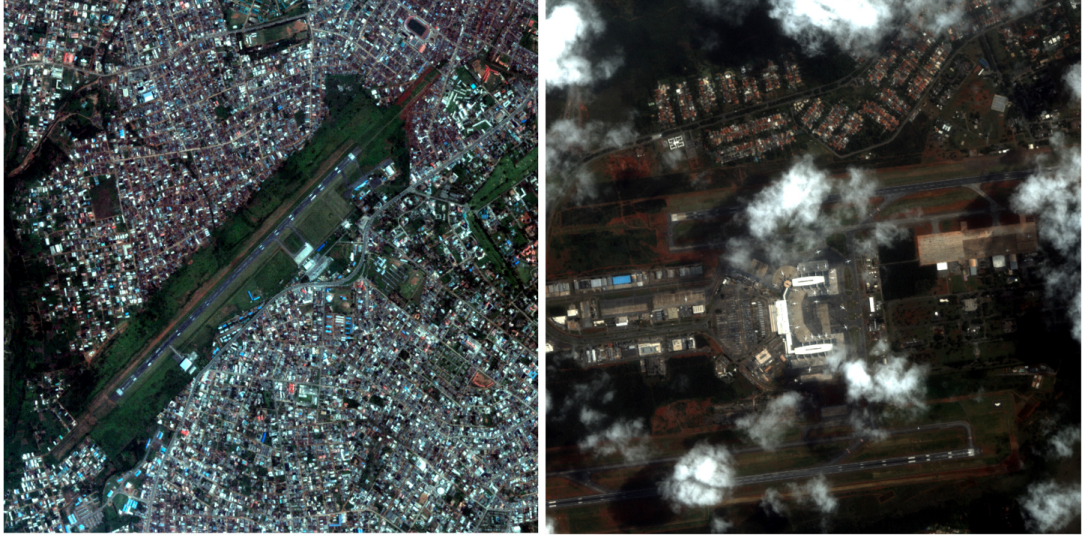


Figure 3.2: Two examples of data from the Carbonite-2 satellite, each 5 km across. The left-hand pane shows a scene from Benin, and the right-hand pane is over an area in Brazil. Data provided by Surrey Satellite Technology Ltd. and Earth-i Ltd.

includes shadows and cloud is the combination of thin and thick annotations. This classification scheme allows for direct comparison with the 9 algorithms tested with the dataset by Foga et al. [106] for cloud detection.

When training and validating on the Landsat 8 dataset, the dataset was split into two halves, each with 6 of the 12 scenes from each biome, and with similar average cloud percentages. For each experiment, the model is initialised randomly and trained on one half, then validated on the other. This is repeated with the halves switched, and the statistics are aggregated between the two.

### 3.4.2 Cloud Coverage Estimation

In this experiment, the aim was to show how the algorithm can be used as a simple coverage assessment tool, rather than as a pixel-wise masking technique. This is important to test for several reasons. First, by checking accuracy over extended images, we are judging whether the algorithm has strong systematic errors over certain terrain or cloud types, which would lead to dramatic biases over certain scenes. Second, on-board data quality assessment would likely be more reliant on cloud coverage, in order to make operational decisions as to which images to downlink.

Table 3.2 shows the results of the coverage estimation experiment. The subclasses of images were chosen not only to further detail the performance of CloudFCN in different situations, but also give insight into CloudFCN’s properties. Urban areas and airports are characterised by many small high albedo areas (rooftops and concrete areas) which

	Total	%Mean	%Median	>2%	>5%	>10%	>20%	>50%
<b>Clear</b>	606	1.85	0.28	10.40	4.46	3.47	2.48	0.66
<b>Clear + Airports</b>	129	1.17	0.34	8.53	4.65	1.55	0.78	0.00
<b>Clear + Roads</b>	322	0.87	0.29	8.70	4.04	0.93	0.31	0.00
<b>Clear + Waves</b>	42	3.71	0.69	30.95	7.14	7.14	7.14	0.00
<b>Clear + Urban</b>	541	1.01	0.29	8.50	2.77	1.66	0.74	0.00
<b>Clear + Crops</b>	379	0.98	0.28	7.39	1.85	1.32	1.06	0.00
<b>Clear + Water</b>	196	1.59	0.40	12.76	4.59	3.06	2.04	0.00
<b>Clear + Snow-Ice</b>	62	9.34	0.29	24.19	19.35	19.35	17.74	6.45
<b>Clear + Docks</b>	79	1.94	0.40	10.13	3.80	3.80	3.80	0.00
<b>Cloudy</b>	72	-	-	-	-	-	12.50	4.16

Table 3.2: Full results of the Carbonite-2 experiments. Mean and median columns give respective error percentages. ‘>  $x\%$ ’ columns give percentage of frames for which the error was higher than  $x$ . The results show a significantly skewed error population, with mean error consistently much larger than median, showing that for half the frames the accuracy is exceptionally good ( $< 0.28\%$ ).

CloudFCN could distinguish from cloud by their surrounding texture, rather than by their colour. Meanwhile, waves have both similar colour and texture to cloud, making them a challenging surface feature to discriminate, which was reflected in the results. Snow-Ice was also understandably challenging, however complete failures ( $>50\%$  error) were still rare.

The Cloudy class is somewhat difficult to interpret. In fact, very few frames were found that were entirely cloud. Most of the frames used had small regions without cloud, making predictions of high cloud cover less likely by the model. Therefore, only the most egregious error populations ( $>20\%$  and  $>50\%$ ) are relevant.

Overall, outlier rates are low, meaning there are only a few example scenes that cause complete failure. This is a desirable property for the algorithm if it is to be used for on-board for data reduction, as very few cloudy scenes will be kept, or clear scenes discarded. Based on this experiment, only 4.16% of cloudy frames would be transmitted, at the cost of 0.66% clear frames being discarded.

### 3.4.3 Pixel-Wise Cloud Detection

This experiment is conducted to ascertain the detection performance of CloudFCN with RGB bands as input. Using the Landsat 8 dataset, the performance in different kinds of terrain can be isolated. The metrics used in evaluating our algorithm’s performance are chosen to be the same as those used in previous studies [105, 106] and are derived with respect to pixel counts: *cloud\_as\_cloud* is the number of cloud pixels predicted



as cloud, whilst *cloud\_as\_visible* is the number of cloud pixels predicted as visible, and so on. *total\_cloud* and *total\_visible* refers to the total count of cloud and visible pixels respectively, and  $N$  is the total number of pixels. The term ‘visible’ is used to describe the union of both clear pixels and shadow pixels, given that the model is not trained to distinguish between them in this study. The three metrics used are defined as

$$\%Correct = \frac{cloud\_as\_cloud + visible\_as\_visible}{N} \quad (3.4)$$

$$\%Omission = \frac{total\_cloud - cloud\_as\_cloud}{total\_cloud} \quad (3.5)$$

$$\%Commission = \frac{visible\_as\_cloud}{total\_visible} \quad (3.6)$$

The final performance metric is given as the  $\%Quality = \%Correct - \%Omission - \%Commission$  and is used to judge the algorithm’s performance against others. The first rows of Table 3.3 documents these results for each of the 8 terrains in the Landsat 8 dataset, and the average over the dataset.  $\%Correct$  averaged 82.81%, although performance varied dramatically in different terrains. In Snow-Ice,  $\%Correct$  was at 50%, primarily due to a high commission rate which suggests it failed to learn features that separated snow from cloud. However, for 4 of the 8 biomes  $\%Correct$  was above 92%. Using only visible bands, CloudFCN outperforms several previous algorithms which use multiple infrared channels. This lends strong evidence to the assertion that using multi-scale features allows for better performance in cloud masking.

The performance of CloudFCN in RGB seems highly dependent on the surface texture. Two of the worst performing terrains: Barren and Snow-Ice, are both characterised by large regions with relatively little texture. In other terrains, such as Urban or Forest, high frequency spatial features make the surface more distinguishable from the cloudy regions. Record performance was measured on Shrubland terrain—although it is not clear why this is the case—this does suggest that still larger datasets are needed in order to reliably gauge the model’s performance.

#### 3.4.4 Multispectral Performance

Many instruments, including Landsat 8, collect data over a range of visible and infrared bands. This section will show how the inclusion of this multispectral data effects the performance of our algorithm. The experiments from Section 3.4.3 are repeated using all 11 bands of Landsat 8. Alongside these, there is a comparison between CloudFCN and several algorithms previously validated on the Landsat 8 CCA dataset [106]. Training and validation are done in an identical way to the previous experiments, and the

Model	Metric	Barren	Forest	Grass-Crops	Shrubland	Snow-Ice	Urban	Water	Wetlands	Mean
<b>CloudFCN (RGB)</b>	Correct	78.92	82.69	94.69	93.41	49.65	93.41	92.34	77.36	82.81
	Omis.	12.24	23.20	4.08	8.82	7.25	2.88	6.34	24.32	11.14
	Commis.	29.89	11.00	7.36	4.47	76.96	8.64	8.18	19.90	20.80
	Quality	36.79	48.48	83.25	<b>80.11</b>	−34.56	81.89	77.82	33.14	50.87
<b>CloudFCN (Multispectral)</b>	Correct	92.95	95.12	96.12	88.68	72.93	95.56	95.43	91.24	91.00
	Omis.	4.70	7.21	6.27	19.66	17.60	2.21	4.62	13.16	9.43
	Commis.	8.95	1.92	1.79	3.23	27.53	5.75	4.50	3.89	7.19
	Quality	<b>79.30</b>	<b>85.99</b>	<b>88.06</b>	65.79	27.80	<b>87.61</b>	<b>86.31</b>	74.19	<b>74.38</b>
<b>ACCA</b>	Quality	63.02	68.69	62	60.47	<b>36.25</b>	68.33	71.43	62.48	61.56
<b>AT-ACCA</b>	Quality	66.67	73.83	74.09	70.65	35.86	74.06	70.51	<b>76.25</b>	67.72
<b>cfmask</b>	Quality	77.1	67.27	85.74	75.53	26.37	74.72	50.98	65.97	65.69
<b>cfmask-conf</b>	Quality	66.78	66.72	83.59	72.3	20.75	76.54	51.11	67.45	63.63
<b>cfmask-nt-cirrus</b>	Quality	54.23	57.2	70.71	71.58	−15.87	74.37	50.23	47.16	51.62
<b>cfmask-nt-cirrus-conf</b>	Quality	54.44	38.79	60.01	66.38	−43.68	73.2	49.04	35.14	41.66
<b>cfmask-t-cirrus</b>	Quality	69.82	64.78	77.98	72.75	−24.1	72.42	57.21	53.27	49.01
<b>cfmask-t-cirrus-conf</b>	Quality	69.37	43.99	77.76	72.34	−52.76	74.72	57.24	52.14	49.63
<b>See5</b>	Quality	54.19	51.88	42.15	42.46	35.48	57.4	39.35	68.17	49.17

Table 3.3: Cloud detection results for Landsat 8 CCA dataset [163]. RGB gives performance of algorithm with Landsat 8 bands 4,3,2 as input. Multispectral uses all 11 Landsat 8 bands. Comparison with 9 other algorithms validated by Foga et al. [106] are given at the bottom, with Quality values derived from the values given in the study. For each biome, the best-performing algorithm is in bold. The multispectral performs best overall, with an average quality metric 6.7% greater than the next highest, AT-ACCA.

same performance metrics used. Visualisation of CloudFCN’s results shows impressive performance in difficult conditions (Figure 3.3)

The performance of the algorithm improved almost universally when all bands were used (see Table 3.3). In 7 of the 8 terrain types within the Landsat 8 dataset, performance improved, often by a very large percentage. The most dramatic improvements were seen in Snow-Ice, and Barren terrains. These terrains also saw the worst performance in RGB, perhaps owing to their high reflectivity and relatively smooth textures, making them difficult backdrops for cloud detection. On average, the multispectral algorithm performed significantly better than all other tested techniques. It is worth noting, however, that the RGB CloudFCN still outperformed several of the previous algorithms.

It is not straightforward to posit which multispectral bands helped the model most in its predictions, given the black box nature of neural networks. However, future work could explore more spectral combinations than those tested here, to better constrain which bands are useful for an architecture like CloudFCN. Nonetheless, the importance of spectral band selection in instrument design is underlined by this experiment. Not only does average performance improve substantially, but the variance in performance over different terrains is reduced by including more spectral data (from a range of 45%

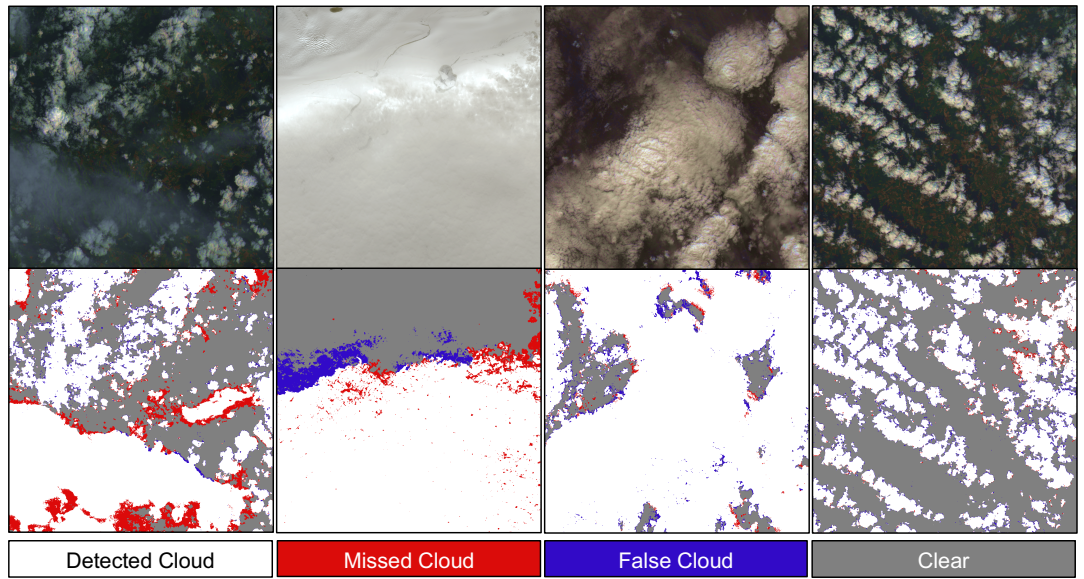


Figure 3.3: Four examples of cloud detection on the Landsat 8 dataset from the multispectral model. Very few mistakes are seen in the clouds’ interiors, but the edges are more error-prone. The first example shows what may be a cirrus cloud over a vegetated region, the boundaries of which show noticeably more errors of omission than neighbouring cumulus clouds. The cloud over snow in the second pane is successfully detected, although large errors are seen at its edge. The final two panes show highly successful detections of different kinds of cloud over varied terrain, despite some disagreement at the very edges of clouds, which are often not well defined and rather ambiguous. All scenes are shown as RGB composites using bands 4, 3 and 2. Note that the key below relates to the four colours in the masks, and not the columns of images.

in RGB accuracies to a range of 23% in multispectral accuracies across the different terrains).

### 3.4.5 Noise Tolerance

By testing on both Carbonite-2 and Landsat 8 data in several configurations, evidence has been provided to show that CloudFCN is capable of high performance across a range of sensors. Remote sensing cameras have a range of different sensitivities, bit depths and noise characteristics, which can adversely effect the quality of cloud masking algorithms. This section further proves the algorithm’s generality by showing the effect on performance of adding white noise and quantization to our validation data. By testing whether CloudFCN is sensitive to these parameters, further evidence can be found for its general applicability as a cloud masking algorithm for Earth observation.

Importantly, directly assessing the absolute performance on noisy Landsat scenes is not of interest here, as those scenes that are degraded badly by noise will not be used in

applications anyway. Rather, these experiments provide us with a reasonable assessment of what level of noise we can expect CloudFCN to overcome when used with other sensors. Comparison between the noise types also indirectly provides insight into the importance of pixel-level vs. textural features for CloudFCN’s predictions, because quantization primarily affects texture whilst leaving pixel intensities relatively unchanged, whilst white noise affects pixel intensities more strongly.

### White Noise

For the white noise experiment, Gaussian noise of varying amplitudes is added to the Landsat 8 dataset. The model is not retrained with more noise applied to it’s training examples, instead just reusing the exact models trained for Table 3.3. For the purposes of this experiment, only the white noise synthetically added to the data is measured, and not the noise already within Landsat 8 images, as it is small in comparison to the added noise [164]. The SNR (see Equation (3.7)) is used to measure the relative power of noise to the original image’s signal, and is defined with respect to the image’s mean ( $\mu_{signal}$ ) and the standard deviation of the noise ( $\sigma_{noise}$ ),

$$SNR(dB) = 10\log_{10}(\mu_{signal}^2/\sigma_{noise}^2) \quad (3.7)$$

CloudFCN is tested at SNR values ranging from 20 dB down to 7 dB, examples of images at different points in this range can be seen in Figure 3.4a. The omission and commission rates for both RGB and multispectral models on the Landsat 8 dataset can be seen in Figure 3.5a. In general, omission increased strongly with noise level, whilst commission was less affected, even going down somewhat for the RGB model. RGB omission was more sensitive, increasing from 11.0% by 42.2%, when compared to multispectral omission increasing from 9.4% by 18.2%.

### Quantisation

Quantisation—the reduction in bit depth—of an image, simulates a common compression technique in image processing, or a limitation in the sensor’s precision. Landsat 8 imagery is delivered in 16-bit precision (although it is derived from 12-bit raw data). In this experiment, Landsat 8 imagery is re-quantised between 16- and 4-bit precision, and track omission and commission rates over the Landsat 8 dataset for RGB and multispectral models. Examples of images at different bit depths show a loss of local texture information at lower precision (Figure 3.4b).

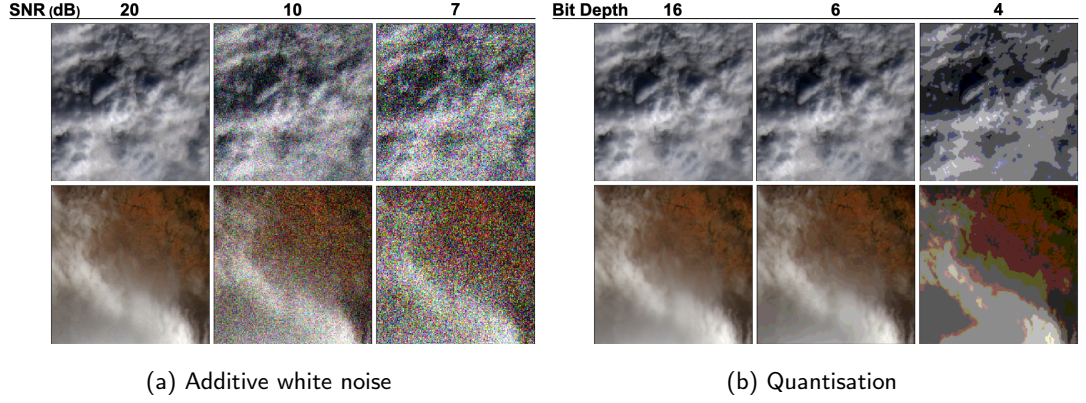


Figure 3.4: Examples of the effect of white noise and quantisation throughout the range of levels applied to the Landsat 8 dataset for the validation in Section 3.4.5. For white noise, an SNR of 7 dB represents a significant distortion of the data, leading to most small features being overwhelmed by white noise. Quantisation also leads to a loss of textural information, acting to smooth large areas with similar intensities.

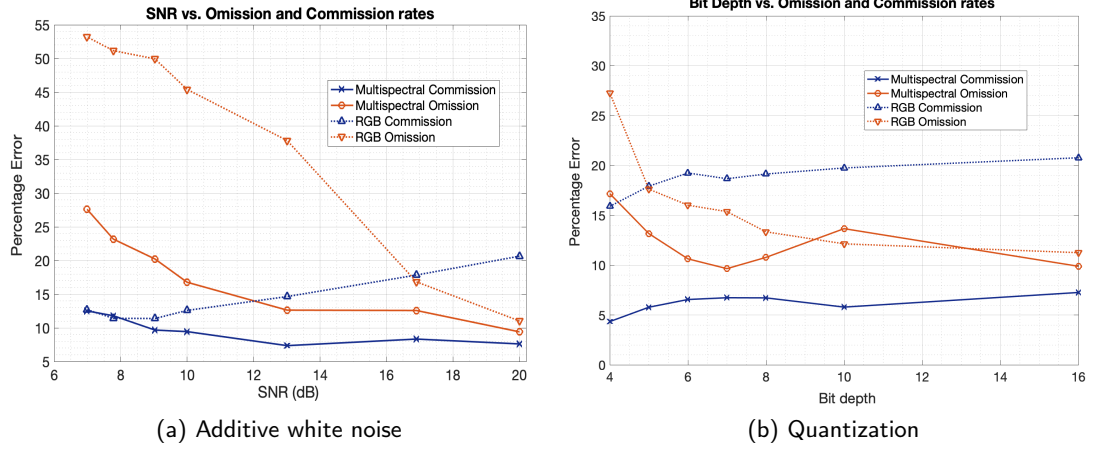


Figure 3.5: Omission and commission rates of both RGB and multispectral models against the SNR (a) and the bit depth of the quantization (b). For both white noise and quantization errors of omission increase with the level of noise. Meanwhile, the effect on commission is less strong, suggesting the added noise has more of an impact on cloudy pixels than non-cloudy ones.

The effect of bit depth on error rates can be seen in Figure 3.5b, and exhibit similar trends for both RGB and multispectral models. In both cases, omission rates increase at low bit rates, whilst commission trends downwards very slightly. The multispectral results suggest a higher tolerance for quantization, with no noticeable increase in omission until the bit depth goes below around 6-bits, whereas the RGB model begins to suffer at around 8-bits. RGB omission increased 16.2% from 11.0% to 27.2%, significantly more than multispectral which increased 7.4% from 9.8% to 17.2%. These results suggest that performance of CloudFCN is not dependent on high precision data, requiring only around 8-bits or more to perform optimally in RGB, or less in multispectral. Additionally,

this experiment supports the claim that CloudFCN learns to place high importance on textural and spatial information, because as the texture of clouds is removed and they become smoother whilst leaving pixel intensities roughly the same, the model begins misclassifying them.

## 3.5 Discussion

These experiments have shown that the proposed deep learning-based model performs well in a range of settings, and is immediately applicable to RGB and multispectral data as a state-of-the-art solution to the problem of cloud masking. In addition, it can be reliably used for scene-wide cloud coverage assessment, showing the potential of this algorithm for on-board use in future missions. On-board applications for convolutional algorithms are now realistic, thanks to the proliferation of several deep learning-specialised chips with low power requirements and small form factors. The RGB mode was used because it provided evidence for performance on the large number of RGB satellites currently being flown. Meanwhile, the multispectral mode makes CloudFCN’s results directly comparable in a fair way with other methods that have access to these Landsat 8 bands. Of course, many satellites have different spectral band combinations, which will be the subject of further study.

Multispectral bands—when available—led to better overall results. This is to be expected, given that many surface features (e.g., urban environments, exposed rock and snow) exhibit high albedo in visible bands, similar to cloud. However, the inclusion of infrared bands gives the model a larger parameter space in which to separate cloud from surface. If some of these bands are unhelpful (in the sense that they provide no correlative power for prediction of output class), then the model’s weights will organise in a way such that the unhelpful bands have negligible impact on the output class.

Performance over snow and ice was reasonable in RGB Carbonite-2 data, with only 6.45% of clear images over snow or ice being marked as >50% cloud. However, on the Landsat 8 dataset performance in RGB was low over this terrain. This suggests that textural information that is useful for discrimination of cloud against snow or ice is exclusively available at higher resolutions (1 m) and is not as present at 30 m, as in Landsat 8 imagery.

The application of cloud masking over snow therefore requires further research and refinement, for which several possible directions exist. For example, simply producing more labelled cloud masks over snow may help future models perform better. Or, a preprocessing step could be used over snowy regions, either with a deep learning approach,

or by geographical correlation with a snow-cover map, which would then be used as an extra input to the model. However, this increases the algorithm’s complexity and adds dependencies on external geographical information.

By applying noise to the data, it is shown that the algorithm can still perform reasonably well with low SNR sensors or with low bit depth data (especially for multispectral images). Although these simulated sources of noise do not allow us to concretely extrapolate the algorithm’s performance to other instruments, it lends strong evidence to support the claim that this architecture can produce a robust solution for a wide range of Earth observation cameras, assuming a relevant dataset can be created.

Ultimately, for deep learning to succeed in providing high-performance and practical tools to the Earth observation community, greater efforts must be made by the community in creating, validating and distributing high-quality, high-volume labelled datasets. Landsat 8 has the benefit of a large, openly available cloud masking dataset which made this study (and many others) possible. However, for many satellite platforms this approach is simply not feasible, given the lack of data. In this regard, traditional, physically-informed thresholding models continue to hold a significant advantage over deep learning, in that they exhibit interoperability on many satellites. Future model development work should therefore focus on creating deep learning models that do have some form of physical information handling capabilities, in order to create interoperable models that take advantage of the high performance of deep learning alongside the flexibility of traditional approaches. This reasoning was the motivation for later work on dataset creation and sensor independent cloud masking (Chapters 4 and 5).

## 3.6 Interim Conclusions

In this chapter, the development of a state-of-the-art cloud masking algorithm, CloudFCN, has been details. This model is a U-Net architecture whose design includes features such as Inception modules, batch normalisation layers and Leaky ReLU layers. These are selected based on evidence of their general success in computer vision tasks, and the specific challenges of cloud masking in remote sensing data. Further, performance testing confirms that FCN architectures are indeed a powerful tool in cloud masking in a range of settings, and can perform better than previous techniques, given high-quality labelled datasets. The success of CloudFCN in cloud masking suggests the architecture is generally applicable to all segmentation tasks in remote sensing. Re-purposing the algorithm for new use cases is straightforward, assuming the existence of sufficient labelled data. A direct extension to this work is the masking of cloud shadows, but more

tangential possibilities also exist. For example—land-use segmentation, sea ice mapping and vegetation studies.

The experimental results of this study underline the high accuracy of CloudFCN, but also demonstrate its robustness to noisy data and a wide range of terrains in Sections 3.4.4 and 3.4.5. This shows the algorithm’s potential as a practical tool for researchers currently using Landsat 8 data, and can inform practitioners when it is most suitable to use CloudFCN, and when its results should be treated with suspicion. This study not only establishes deep learning models as the best-performing method for cloud masking, but that it provides incentive for those working with other satellites to create large, hand-labelled datasets for segmentation problems (if they do not yet exist). This will allow deep learning techniques to be better exploited on a wider range of sensors, and is the focus of the next chapter.



## 4 | Cloud Masking Dataset

*Some sections of this chapter are based on the documentation of the dataset, published and available to download on Zenodo [3]. John Mrziglod and I collaborated in designing the IRIS annotation tool, the code for which was written largely by John. We performed roughly 50% of the dataset's annotations each. I validated and organised the final dataset separately, and wrote the documentation on which this chapter is based.*

### 4.1 Motivation

Early in the development of cloud masking algorithms, the need for validation data by which to assess the performance of model outputs was recognised. For example, in 1982, [95] discussed the use of ground-based observations to verify the accuracy of cloud masking algorithms. As well as validation through comparison with some source of trusted data, there is now also a need for data to use in the training of models, especially now that machine learning approaches have become more widespread and commonplace.

Currently, several datasets exist for cloud masking, which all have different strengths and weaknesses. Multiple labelled datasets have been produced by the USGS for the Landsat series of satellites, in particular, one for Landsat 7 [165], and two for Landsat 8 [126, 163] (the first of which is used in Chapter 3). For Sentinel-2, some data exists, but the size of those datasets is relatively small compared with those for the Landsat missions. The largest Sentinel-2 dataset [166] contains 38 scenes, annotated in full. For a breakdown of prominent existing datasets, see Table 4.1.

In order to experiment with sensor independence, data from a diverse set of satellites is preferable. Moreover, satellites with many bands provide greater variety when training sensor independent models (as we will see in Chapter 5). Given that Sentinel-2 has a relatively small amount of labelled data, it was determined that creating a dataset of Sentinel-2 images would have the greatest impact for both this work and others'.

In planning to create this dataset, some observations about cloud masking were made which informed the structure and methods used:

- (i) model performance over an entire image or product is highly correlated, which means using smaller sections of scenes has more value per pixel than using whole scenes.

- (ii) current cloud masking datasets often focus on specific regions, or hand-select the products used, which introduces a bias into the dataset that is not representative of the real-world data it is sampled from.
- (iii) cloud mask performance appears to be highly correlated to surface type and cloud structure, so a dataset should allow users to easily test models in different scenarios.
- (iv) labelling each pixel with different cloud types or land cover classes is very time consuming and often ambiguous. However, categorising each image does not take long and can provide some value (e.g. the 8 biomes in the Landsat 8 dataset used in Chapter 3).

These factors have led to the conclusion that a dataset which prioritises the number of products used over the number of pixels, and samples scenes randomly, would be of massive value to a study on sensor independence and to the wider research community, especially for Sentinel-2. In order to complete the annotations in a timely fashion whilst maintaining high accuracy, we opted to develop a semi-automated annotation tool which is described further in Section 4.2. The simple model employed during semi-automatic annotation (a random forest) works well because of observation (i) mentioned previously. Whilst a simple pixel-based approach like this would struggle to work across a diverse range of scenes, the fact that it is retrained from scratch on each image makes its task much more straightforward. The details of how the dataset was created is then discussed in Section 4.3, including the selection of scenes, and the labelling strategy. Then, a summary of the completed dataset is given in Section 4.4. Section 4.5 and 4.6 then look at the statistics of the dataset, and the steps taken to validate the results.

Dataset	Satellite	Resolution (m)	No. of images	No. of megapixels	Area (km <sup>2</sup> )	No. of classes	Cloud cover	Labelling	Sampling
L7 CCA [92]	Landsat 7	30	206	7'560	6.8e6	4	33.2%	Manual	Evenly across 9 latitude zones
SPARCS [126]	Landsat 8	30	80	80	7.2e4	7	19.4%	Manual	By discretion
L8 CCA [163]	Landsat 8	30	96	4'000	3.6e6	4	47.9%	Manual	Evenly across 8 biomes
95-Cloud [167]	Landsat 8 (RGB+NIR only)	30	95	5'100	4.6e6	2	50% [121]	Manual	Mostly over North America
CESBIO [166]	Sentinel-2	60	38	105	3.8e5	5	19.6%	Semi-automated	Multiple scenes over a few sites
CloudPeru2 [168]	PerúSat-1	2.8	153	5'900	4.6e4	2	48.9%	Manual	Over Peru
OURS [3]	Sentinel-2	20	513	536	2.1e5	3	53.1%	Semi-automated	Random across 2018 catalogue

Table 4.1: A non-exhaustive list of publicly available datasets for cloud masking. All of these datasets are labelled as complete segmentation masks, where every pixel in the input image is assigned a class.

## 4.2 IRIS Annotation Tool

John Mrziglod (previously at European Space Agency, and World Food Programme) and I collaborated to develop an annotation tool for the dataset, Intelligent Reinforcement for Image Segmentation (IRIS), which is designed to make image segmentation for multispectral imagery as efficient as practicable. The key feature of IRIS is an iterative random forest model which aids the user in annotating the image. The basic workflow for the user starts with labelling some pixels of different classes (e.g. *clear*, *cloud*, and *cloud\_shadow*). These are then used to train a random forest, which subsequently predicts the rest of the pixels in the image. The user can then inspect these predictions and correct some errors, and retrain the model. This can continue for an arbitrary number of iterations, until the user is satisfied with the accuracy of the mask. Around this core feature, several design elements are included to make the task easier and more reliable. This section first describes the random forest in more detail, and then documents the other software features and how they can be used to aid in annotation.

### 4.2.1 Semi-Automated Labelling in IRIS

The Random Forest (RF) built into IRIS is vital for reducing the time spent annotating images. As outlined previously, the RF learns from the pixels manually labelled by the user, and then extrapolates what it has learnt to the rest of the image. For the images we were using (1022-by-1022 pixels), training and prediction took a few seconds, depending on the complexity of the RF being used. This can be activated using a hotkey or a button on the IRIS interface.

As input, the RF receives the reflectances of the Sentinel-2 bands at each pixel position. As well as these band values, IRIS also allows users to provide the RF with some other features at each pixel, which can help its predictions. Edge filters can be added to the input (which can help it close to class boundaries or on other complex structures). These edge filters are simply the gradient in reflectances along the x- and y-axes, and so give the RF extra information about whether the pixel is in an area with high structure, or not.

Another possible input for the model is known as the “meshgrid”. The meshgrid splits the image along x- and y-axes into a set number of grid tiles (e.g. 3-by-3). Then, the RF is given as input which grid cell that the pixel is within, as an integer index value. For example, a pixel in the top left grid cell might have a cell index of 1, whilst a pixel in the bottom right might have a cell index of 9. By giving the model this as

an input, different styles of prediction can be learnt for different sections of the image. This can be useful in extremely difficult circumstances, where different sections of the image have very different conditions or features, making classification without knowledge of the location very challenging. In practice, we found that it was not advisable to use the meshgrid unless absolutely necessary, as many annotations of each class would have to be provided for every grid cell, otherwise the model would overfit and mark all pixels in a given grid cell as a certain class.

Preferences	
Model Parameters	
Number of estimators:	100
Maximal depth:	30
Number of leaves:	30
Train ratio:	80%
Max. number of training pixels per class:	20000
Model Inputs	
Use context metrics?	<input type="checkbox"/>
Use edge filter?	<input type="checkbox"/>
Use meshgrid?	<input type="checkbox"/>
Meshgrid cells	3x3
Use superpixels?	<input type="checkbox"/>
Inputs bands	<div> <div>Bands to include</div> <div> <div>\$Sentinel2.B:</div> <div>\$Sentinel2.B:</div> <div>\$Sentinel2.B:</div> <div>\$Sentinel2.B:</div> <div>\$Sentinel2.B:</div> <div>\$Sentinel2.B:</div> <div>\$Sentinel2.B:</div> <div>\$Sentinel2.B:</div> <div>\$Sentinel2.B:</div> <div>\$Sentinel2.B:</div> </div> <div> <div>&lt;</div> <div>&gt;</div> </div> <div>Bands to exclude</div> </div>
Postprocessing	
Suppression filter size:	5
Suppression filter threshold:	0%
Suppression filter background class:	Clear

Figure 4.1: Configuration pane for model in IRIS. Values shown here are the defaults, set at the beginning of annotation of each image.

image, the labelling errors are assumed to be at least somewhat uncorrelated between images, reducing systematic errors and increasing variance.

When an image is loaded and annotation begins, the RF model is initialised in a fairly simple configuration, with 100 trees, each with a maximum depth of 30, and a maximum of 30 leaves, with only the Sentinel-2 band reflectances as input (see Figure

The RF is retrained from scratch for every image. This is done for several reasons. First, a pixel-by-pixel technique such as an RF does not have the capacity to reliably learn how to detect clouds over a diverse set of images (otherwise the whole exercise of training a convolutional model would be redundant). Second, even if the model was able to learn from and improve its performance over many images, then those annotated at the start would be unfairly disadvantaged. Third, and perhaps most importantly, it is impossible to eliminate all errors during annotation for every image, but by using the same RF for all images, it would turn that random variance of labelling styles between different images into a fixed bias. This would then lead any model trained on this dataset to simply learn the biased style of annotations that the RF had learnt. Instead, by resetting the model for each

4.1). This simple model tends to avoid overfitting, which is useful at the beginning of annotation, given the relatively few pixels the user will have labelled. As the number of corrections and additional labels made by the user grows, they may choose to increase the model’s complexity as they see fit, especially if they notice that the model’s performance has plateaued over several iterations. This can be done easily during annotation from IRIS’s Preferences tab (see Figure 4.1). After updating the model’s parameters in the Preferences tab, the model must then be retrained for those changes to take effect.

As well as being able to alter the inputs of the RF, there is also a post-processing stage, which can help to denoise the results. This was implemented because we noticed that in areas close to class boundaries, or where uncertainty in the model was high, there would be scattered, high frequency changes in the class predictions, because each pixel gave slightly different results. To mitigate this, the suppression filter eliminates predictions which do not match a certain percentage of its neighbours. The filter is turned off by default (see Figure 4.1). Both the size of the window around each pixel, and the percentage of that window which must be the same class for it to pass through the filter, can be configured. By default, pixels which fail to meet the criteria are turned to clear, as this is often the “background” class in an image. However, this can also be changed in situations where the image is predominantly cloudy, with small non-cloudy areas. The net result of the suppression filter is to smooth the edges of predictions, making them look more coherent and less noisy.

### 4.2.2 Visualisation in IRIS

IRIS uses multispectral data, and so decisions must be made as to how the data is displayed, usually as false colour composites in RGB on a computer monitor, especially when many bands are present, as is the case for Sentinel-2. For this reason, when setting up IRIS, the user can define a set of “*views*”; a collection of different visualisations that can be flipped between and compared during labelling. The ones used in the project are defined in Table 4.2, and cover several common ways to visualise Sentinel-2 data. It was found that certain views were much more effective than others over certain surface types and cloud environments, and having a large selection that could be easily switched between improved the user experience dramatically.

As well as changing the band combinations being visualised in IRIS, the appearance of the image could be altered on the fly with hotkeys for increasing and decreasing the saturation, contrast and brightness of the images. This was often helpful in areas of very high reflectance, like snow, where decreasing brightness and increasing contrast could

Name	Bands
RGB	B4, B3, B2
Colour Infrared	B8, B3, B2
Red Edge	B6, B5, B4
Land vs. Water	B8, B11, B4
Urban	B12, B11, B4
Agriculture	B11, B8, B2
Atmospheric Penetration	B12, B11, B8A
Snow	B1, B11, B12
NDVI	$(B8-B4) / (B8+B4)$
NDWI	$(B8A-B12) / (B8A+B12)$
NDSI	$(B8-B4) / (B8+B4)$
Cirrus	B10

Table 4.2: Views defined in IRIS for the Sentinel-2 cloud mask catalogue. Whilst many of the views' names are based on features that they visualise clearly, all views were used interchangeably during annotation, with no fixed use-cases. RGB, Colour Infrared, Snow, and Cirrus were particularly useful.

reveal more detail, or over sea where reflectance was low and the image was too dark without additional brightening. Panning and zooming were also easy to perform by dragging across the screen and using the mouse wheel. This zooming was essential when annotating small areas close to the edges of small clouds, for example.

We used IRIS to annotate images of a fixed size (1022-by-1022 pixels at 20m resolution). However, the edges of those images are more difficult to annotate if the areas outside the boundary cannot be seen, because adjacent areas inform how we label a given point. To eliminate this, we designed IRIS to display a slightly extended image, 64 pixels more per side, than we actually annotate. When labelling, the extent of the actual area being masked is bounded by a dashed line, and marking cannot be made outside of it.

### 4.2.3 Painting in IRIS

IRIS uses a paintbrush tool that could be quickly altered in size using hotkeys. This enables the user to rapidly fill large areas, and then switch to a smaller brush-size to add detailed labels at the edges if needed. Often, after training and predicting with the model, small errors are made by the RF at the edges of clouds, and so a very fine brushstroke can be helpful in correcting these. As well as the paintbrush tool, there is

also an eraser tool, which can be used to delete the user’s annotations, or to clear the image of annotations completely.

The segmentation mask can be overlaid on top of the image using a set of semi-transparent colours (see Figure 4.2, one for each class. The hand-drawn labels (known as the *user mask*) are visualised separately to the *final mask*—the combined mask of user and model output—so that the user can see where they have already marked, and not get confused between their own markings and those of the random forest.

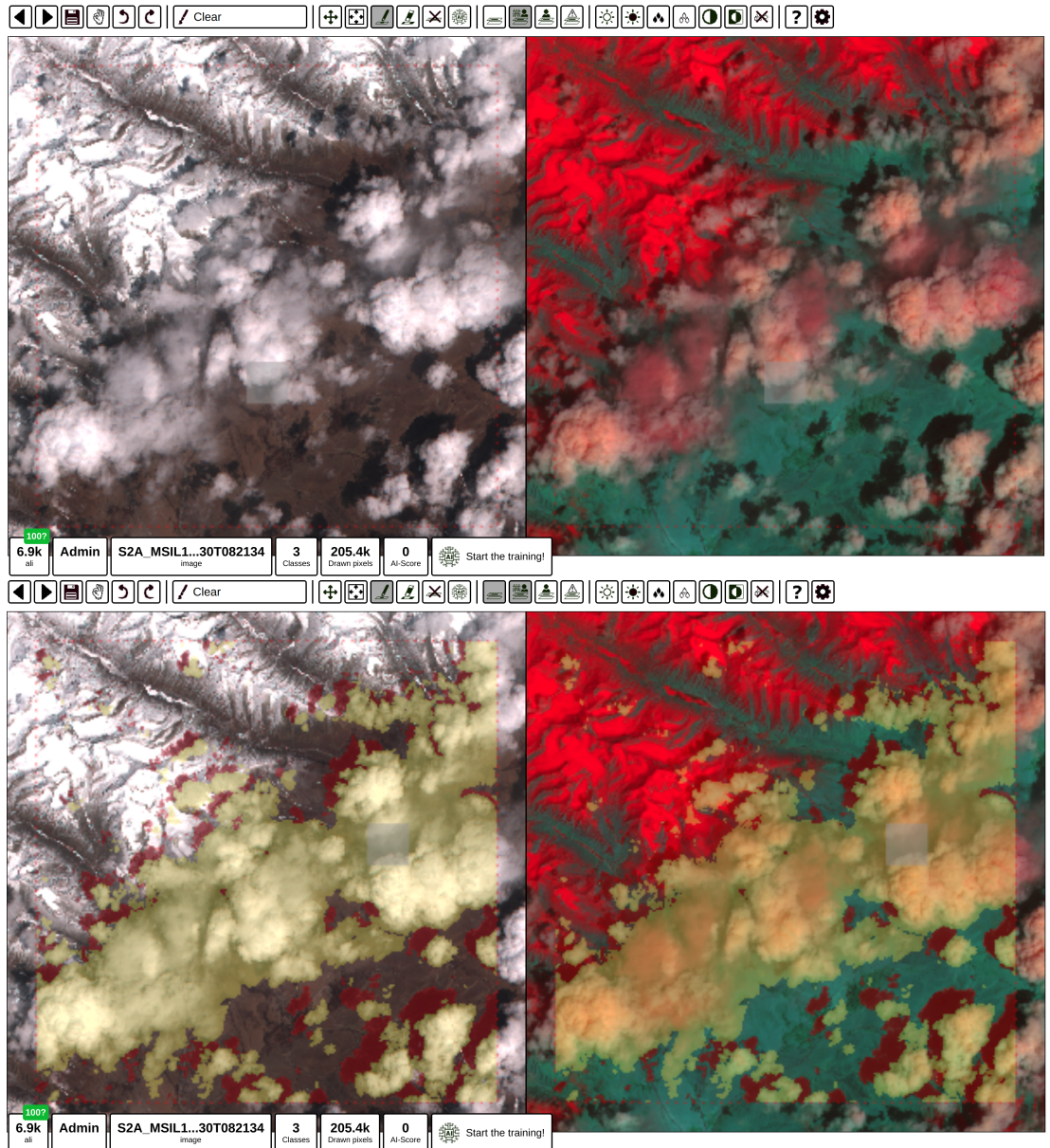


Figure 4.2: User interface in the IRIS software when annotating. Sentinel-2 subscene shown in “RGB” view (left) and “Snow” view (right)—see Table 4.2 for their band combinations. The bottom pane is an example of when the mask visualisation is active. Note the bright red colour of the snow covered areas, making it easier to delineate cloud from surface over the mountainous region. This screenshot shows the mask close to completion, with few errors left for the user to resolve.

#### 4.2.4 Other Features

When annotating many hundreds of images, it is inevitable that difficulties arise which could not have been foreseen during the development of IRIS. To allow greater flexibility during the annotation process, each user has the option to leave notes for each image they annotate, describing any problems that arose. For example, we used these to indicate when an image was entirely clear, as it could be mistaken for an unlabelled scene that had been accidentally marked as complete, rather than one which was inspected and found to contain no cloud. We also used the notes to indicate where it was not possible, due to stratigraphic shadow (shadows cast by landforms, such as mountains) or unclear boundaries, to mark cloud shadow. The notes could also be used to indicate when images show an interesting or surprising feature.

For some images, none of the views we defined were particularly helpful in discerning what it was we were actually looking at. In these instances, the hyperlink to the coordinates of the image in *Google Maps* was incredibly useful, allowing us to quickly see where on the planet we were looking, and see it using the data available on the *Google Maps* satellite view, which is almost always cloud-free and higher resolution than Sentinel-2. Often this was used when the entire scene had a near-constant high reflectance, and it was difficult to tell whether it was either entirely cloudy, or entirely clear over an extended icy terrain, like Antarctica.

Undo and redo buttons were extremely useful, especially when training and retraining the RF based on new annotations. If the user is unhappy with the RF's new predictions, or has made a mistake during labelling, they can quickly undo the changes, or experiment and compare by repeatedly undoing and redoing. This requires IRIS to save a history of changes so that it can revert to previous states.

After each image was annotated, the software also asks users to confirm whether they are happy with the mask, or whether they would like to continue working on it later. Additionally, the user is asked to rate on a subjective scale from 1–5 how difficult the image was to annotate:

1. Near perfect
2. Very confident
3. Mostly confident
4. Possibly some errors
5. Possibly large errors

Besides the annotation interface, which is where users actually create the segmentation masks, there is also an *admin page*, which gives those users with administrator



privileges more information about the project. In the admin page, the work done by users can be viewed in tabular form. Each mask that a user creates forms a row in the table, with the image id, user id, date of creation, whether the mask is confirmed as final by the user, and the difficulty rating. Alternatively, rather than per mask, the table can also be viewed as one row per image, showing the number of users who have annotated it, and a measure of agreement (the  $F_1$  score averaged across classes) between them. A progress bar also indicates the total percentage of the images which have been labelled.

In real-world use, all of the features described above can be used to massively increase the efficiency of the labeller, especially when hotkeys are used to eliminate mouse movements. Whilst saving a small amount of time per individual use, the net time saved in mouse movement is huge when annotating many images. When combined with the time saved in not needing to paint every pixel, because the random forest infills the majority for you with its predictions, annotation becomes incredibly rapid. We estimate that annotation of all 513 images, 60 of which were annotated twice, took roughly 100 hours of time between the two of us. This means, on average, each scene took roughly 5min 45s to label. Whilst we did not measure the time taken for each individual scene, we both noted that the vast majority of scenes took far less time, whilst a few with complex structures that the RF failed to label correctly took far longer (see details about some of these in Section 4.6), sometimes requiring over an hour. Based on some initial testing involving manual annotations on a few images, IRIS led roughly to a fivefold increase in speed, meaning that over the entire dataset, around 400 hours of labelling time were saved.

## 4.3 Dataset Creation

We first randomly sampled images from the Sentinel-2 2018 archive. By “random”, we mean that every product in the 2018 archive had an equal chance of being selected. For each selected product, we then extracted random areas of 1152-by-1152 pixels at 20m (note the difference between this and the final 1022-by-1022 subscenes).

Once our subscenes were extracted, we began annotating some test images, and configured IRIS. Several different views were defined based on these tests, as described in Section 4.2.2. During annotation, we padded the subscenes by 64 pixels on each side (resulting in the 1152-by-1152 image used in IRIS), so that there was no disadvantage to annotations close to the edge of the final 1022-by-1022 window. Originally, the intention was to export 1024-by-1024 pixel masks, however we noticed too late a bug in IRIS which

caused all edge pixels of the mask to be unfilled, hence we cropped a pixel from each side of the images and masks, leaving the final 1022-by-1022 window size.

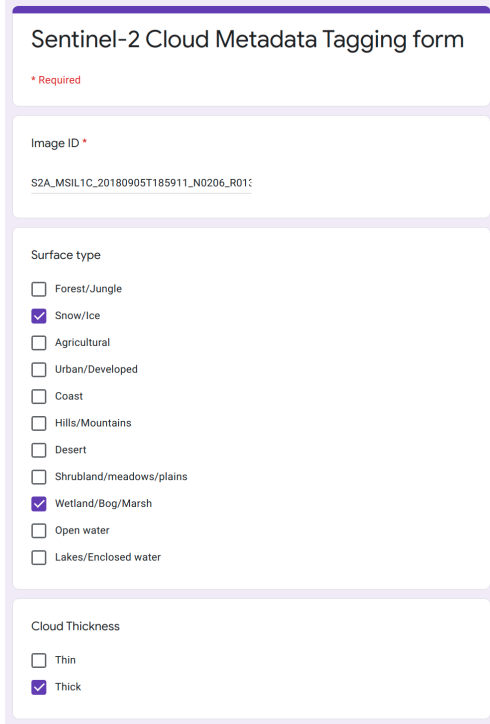


Figure 4.3: Example of the Google Form used to add classification labels to each scene. Cloud Height and Cloud Type were also selected in this interface. The checkboxes allowed us to select multiple non-mutually exclusive options within each category. The results from this Google Form were then automatically tabulated into a Comma Separated Values (CSV) file, with a row for each entry.

As mentioned previously, the RF could be configured dynamically during annotation. As a general rule, we began annotating a subscene with a simple, small RF that just used the reflectance values. If the model was seen to be struggling, then edge filters could be added as input, and the size of the RF could be increased in terms of number of trees, branches and leaves. A suppression filter could also be activated, which smoothed the mask by removing cloudy/shadowy pixels which were not in the vicinity of others. We always checked carefully over the image to check that the RF had done a reasonable job, and often found that significant numbers of iterations and manual corrections were required.

We annotated with high recall in mind. If we felt split between annotating an area as cloudy or clear, we defaulted to marking as cloudy. There is no perfect solution to this, as cloud masking is an inherently ambiguous task, but we felt that most users of cloud masks de-

sired high recall, and thus our cloud masks should be cautious and err on the side of cloud when in doubt.

Our annotation style was formalised using 10 hand-picked images (which contained difficult or instructive examples of the kinds of decisions that would need to be made). These 10 images constitute the *calibration* set, on which we refined our annotations in an attempt to get as high an agreement between us as possible, within a session of a few hours.

Once *calibration* was completed, we began annotating subscenes individually, served to us randomly by IRIS. This comprised the scenes with the *main* dataset, totalling 453 annotated subscenes. After this, we then decided to both annotate another 50 subscenes in parallel (without consulting one another like we had in *calibration*). This *validation*

set can then be used to measure the level of inter-annotator agreement, and thus gives users a rough guide of human-level performance (discussed further in Section 4.6).

After the segmentation masks were completed, we then went through every image again and added image-wise classification tags (see Section 4.4 for the definitions of each tag used). These were gathered externally to IRIS, and were entered into a spreadsheet using a Google Form, in which the product ID of each scene was entered, along with checkboxes for each of the classification tags (see Figure 4.3). Some logical checks were then carried out on the spreadsheet to flag and correct human errors in this process (e.g. no image can contain a *cloud type*, but not a *cloud thickness*, *relative cloud height* and *cloud extent* tag). Undoubtedly, some errors remain in the classification tags, and many of the images were inherently ambiguous, with their classification very much a subjective judgement.

## 4.4 Dataset Description

The final dataset comprises cloud masks for 513 1022-by-1022 pixel subscenes, at 20m resolution, sampled randomly from the 2018 Level-1C Sentinel-2 archive. In addition to the pixel-wise, 3 class (*clear*, *cloud*, *cloud\_shadow*) segmentation masks, we also provide users with binary classification “tags” for each subscene that can be used in testing to determine performance in specific circumstances. These include:

- **SURFACE TYPE:** 11 non-mutually exclusive categories
  - **forest/jungle:** dense tree cover
  - **snow/ice:** any snow or ice covering either land or water
  - **agricultural:** farmlands or pasture
  - **urban/developed:** human structures, conurbations, villages, industrial sites (does not include solitary roads)
  - **coastal:** coastlines, specifically (not just water near to land, but the division between land and water itself)
  - **hills/mountains:** noticeably hilly terrain
  - **desert/barren:** completely arid, or very sparsely vegetated drylands
  - **shrublands/plains:** somewhat sparsely vegetated areas, or non-forested areas without obvious signs of agriculture
  - **wetland/bog/marsh:** areas with large amounts of standing water (although not extended, large bodies of water, as this is dealt with separately)

- **open\_water:** extended bodies of water, such as large lakes that continue beyond the image bounds, or the sea
- **enclosed\_water:** lakes, large rivers, or contained areas of seawater (e.g. fjords)
- **CLOUD TYPE:** 7 non-mutually exclusive categories
  - **cumulus:** rounded, clumpy and separate clouds
  - **cumulonimbus:** clouds with very large, steep, vertical profiles, reaching up very high
  - **altocumulus/stratocumulus:** most general cloud type, denoting any clouds that form periodic cells. Could be high or low, and thick or thin
  - **cirrus:** thin, extended clouds that have high reflectance in the Cirrus (B10) band. Sometimes almost invisible or very faint in any other band
  - **haze/fog:** low-lying, textureless, thin cloud. Often distinguishable because it looks similar to cirrus, but with no reflectance in B10
  - **ice\_clouds:** any cloud that has a noticeable icy signature, seen as reddish-orange in the false-colour B1 (red), B11 (green), B12 (blue)
  - **contrails:** exhaust trails left by planes
- **RELATIVE CLOUD HEIGHT:** 2 non-mutually exclusive categories
  - **low:** any cloud which has no obvious sign of high-altitude (high reflectance in the Cirrus (B10) band)
  - **high:** any cloud with noticeable reflectance in the Cirrus (B10) band
- **CLOUD THICKNESS:** 2 non-mutually exclusive categories
  - **thin:** any cloud where some surface colour/signal can be seen through it (not including at the edges of thick clouds, to avoid all thick tags being accompanied by thin ones)
  - **thick:** any cloud where no surface can be seen through it (except possibly at its edges)
- **CLOUD EXTENT:** 2 categories
  - **isolated:** clouds which are separated from one another, such that there are no areas of continuous cloud across the majority of the subscene
  - **extended:** clouds which span continuously across a majority of a subscene

By including subscenes from so many Sentinel-2 products, selected completely at random, it was possible to produce a statistically representative, stratified sampling of the whole Sentinel-2 archive. Without having several hundred scenes, it would be difficult to be certain that the dataset was representative of the wider Sentinel-2 archive, especially with regard to rarer cloud formations or surface types.

Wherever practical, cloud shadows were also annotated, however this was sometimes not possible due to high-relief terrain, or large ambiguities. In total, 424 were marked with shadows (if present), and 89 have shadows that were not annotatable due to very ambiguous shadow boundaries, or terrain that cast significant shadows. If users wish to train an algorithm specifically for cloud shadow masks, we advise them to remove those 89 images for which shadow was not possible, however, they need to bear in mind that this will systematically reduce the difficulty of the shadow class compared to real-world use, as these contain the most difficult shadow examples.

We also provide users with shapefiles that define the boundary of the mask on the original Sentinel-2 scene in latitudinal and longitudinal coordinates. If users wish to retrieve the L1C bands at their original resolutions, or fuse the scenes with other data from that geographical area, they can use these to do so.

## 4.5 Dataset Statistics

This section uses statistics to help describe the dataset, and validate our labelling strategy. The prevalence of the three classes (*clear*, *cloud*, *cloud\_shadow*) is shown in Figure 4.4. As we can see, *clear* and *cloud* classes are roughly equal, with around 50% each, whereas *cloud\_shadow* is much rarer. This is an underestimate for the true percentage of shadow, though, given that some scenes contained shadow that could not be annotated with this class. The difficulty, rated between 1 and 5, was a subjective measure used to note where we thought substantial errors may be found in the segmentation masks. Happily, the majority of images were of difficulty 1, with only around 17% at difficulty 4 or 5 (Figure 4.5).

We can also look at the percentage of scenes with each classification tag. As they are not mutually exclusive, the tags of a given type (e.g. cloud height) do not add up to 100%. Some tags, such as contrails, are exceedingly rare, whilst cumulus clouds are observed in the majority of subscenes. The categories that are traditionally of most concern when using cloud masks (thin cloud, cirrus cloud, and snow/ice) are all well represented (Figure 4.6).



Figure 4.4: Pixel-wise class percentages across the dataset. *Clear* and *cloud* are found in roughly equal amounts, whilst *cloud\_shadow* is much rarer.



Figure 4.5: Distribution of user-rated difficulty scores across the images. Slightly more than half of images were marked as very easy (these were often subscenes which were completely cloudy or clear).

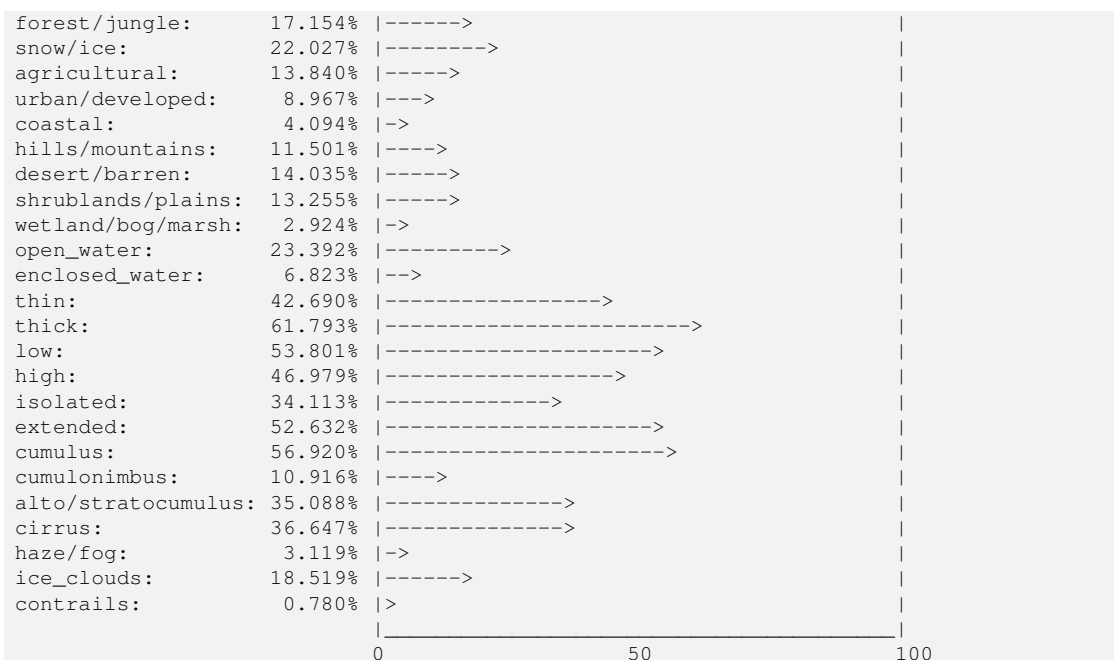


Figure 4.6: Prevalence of each classification tag across the 513 images.

## 4.6 Validation and Uncertainties

The labelling of clouds is an inherently ambiguous task, given that the optical depth of cloud lies on a continuum from almost entirely opaque to vanishingly thin. Inevitably, the human labeller must define a thinness at which to delineate between cloudy and clear, and this must be done with only visual cues over varied terrain. Often, even if a cloud is thick enough to be marked, its boundaries can be ill-defined and impossible to segment with certainty.

By using multiple annotators, we inevitably introduce some additional variance into the segmentation masks, beyond the variance produced by a single annotator. We can measure the level of inter-annotator agreement on the validation set, to give us an idea of the total variance in annotations. We opted to use the class-wise  $F_1$  score, meaning each class was treated as the “positive” class against all others, and the  $F_1$  score then calculated between the two masks in the usual way as defined in 2.2.7. This has the advantage of being symmetrical with respect to which mask is treated as the “prediction” and which as the “truth” (the recall of one annotator is the precision of the other, and vice versa), and so is a good metric for comparing the agreement between the two labellers. We also calculate the total percentage of pixels in agreement with one another, which we refer to as the accuracy. However, this does not tell us about the agreement with respect to specific classes like the class-wise  $F_1$  score.

First, we can measure these metrics for the *calibration* set of 10 images, labelled at the beginning of the dataset’s creation (Figure 4.7). At the end of the labelling exercise, the 50 subscene validation set was also annotated by both of us (Figure 4.8). We were surprised to find a higher overall agreement for *validation*, given that we worked on the masks separately, without consultation. However, this may be because the *calibration* images were hand picked because of their usefulness in demonstrating certain features that may be difficult to annotate consistently. We encourage users to split the dataset such that the test set includes all calibration and validation images, so that an approximate comparison can be made between model performance and inter-annotator agreement.



Figure 4.7:  $F_1$  scores and accuracy between the labels of the two annotators across the 10 images in the *calibration* set.



Figure 4.8:  $F_1$  scores and accuracy between the labels of the two annotators across the 50 images in the *validation* set.

### 4.6.1 Known Issues

Here we review the issues with the dataset that affect its reliability.

**Mask boundaries:** As discussed previously, we discovered a small bug that affected the border pixels of each mask, making them all clear. To resolve this issue, we have cropped all masks and subscenes to be 1022 pixels across in both dimensions.

**Cloud pixels surrounding shadows:** We noticed in many scenes that, at the boundary between cloud shadow and clear pixels, a thin line of cloud pixels could sometimes erroneously appear in the Random Forest prediction. We corrected these errors manually when they were spotted during annotation, however, because it affected very small areas, some were missed. It is likely that this occurred because the edge of cloud shadows resembled the darker, shaded areas of clouds to the Random Forest. We considered attempting to use a post-processing technique to remove these artifacts, however we determined that it would likely cause more errors than it would solve. Overall, these are most often a few isolated pixels at the edges of some shadows, and so should not effect an algorithm’s training (or test accuracy) too dramatically.

**Large, thin cloud boundaries:** If thin clouds (often cirrus) appeared over a large section of the image, it was often incredibly difficult to tell exactly where the boundary lay between clear and cloud. Ultimately this is a subjective exercise, and we approached



it with the understanding that, if in doubt, we would mark it as cloud. This is because most end-users of cloud masks desire high cloud recall, though we appreciate it may not suit users who prefer high precision.

**High-reflectance icy subscenes:** Some subscenes, often over the Antarctic, were essentially completely white in all bands, except the SWIR bands, indicating that the entire subscene was dominated by ice or snow. It was often difficult to tell if this was either clear, high-altitude ice shelves, or smooth icy clouds. We therefore usually marked these with high difficulty (4 or 5), as we were never completely certain that we hadn't marked the entire scene incorrectly.

**Classification tag ambiguities:** The classification tag scheme was developed to be a quick-and-easy way to understand how an algorithm performs in a given context. They are based on visual cues (described in Section 4.5), rather than physical measurements. As a result, the categories used will not be suitable for every user, and some human errors will be more likely to persist. In particular, *cloud type* should be seen as much more of a subjective classification, that delineates clouds by certain visual characteristics, but without any formal meteorological rigour. We recommend that if users have specific needs of their own they should develop their own classification scheme to compliment the one provided here.

## 4.7 Interim Conclusions

Existing cloud masking datasets are heavily focused on the Landsat series of satellites. We have designed this dataset to complement those prior efforts by focusing instead on Sentinel-2, with the philosophy of maximising the diversity of scenes used, whilst sampling them at random. We have produced the largest (with respect to number of scenes) cloud masking dataset currently available. In pursuing this, we have also designed and released a highly practical, intelligent, and adaptable tool, IRIS, that can be used for a huge variety of image segmentation labelling with any image data held in a multi-channel array, not limited to multispectral satellite data.

Research into sensor independent deep learning requires—by definition—datasets from many sensors. With this Sentinel-2 dataset, we have opened up the possibility for training and testing sensor independent models, which Chapter 5 will deal with. Future labelling work can further extend this to other sensors, applying the same framework for semi-automated annotation using IRIS.



## 5 | SEnSel

*This chapter is based heavily on a paper accepted for publication in IEEE Transactions on Geoscience and Remote Sensing. The technical work presented here was carried out by myself, but was supported by the activities of my co-authors as described in Chapter 4.*

### 5.1 Motivation

This chapter proposes a versatile framework, Spectral ENcoder for SEnsor Independence (SEnSeI), which enables deep learning architectures to become sensor independent. SEnSeI is a pre-processing module, which acts as the translator between the “languages” of individual sensors (their individual channels’ reflectances and wavelengths) and a shared feature space that can then be inputted into an existing deep learning model (or indeed, any other machine learning model). SEnSeI’s design does not depend on the model it is used with, allowing it to be bolted onto any pre-existing machine learning or deep learning architecture, which can then be trained to be sensor independent. SEnSeI is designed as a permutation invariant neural network, the theoretical background and practical development of which is set out in Section 2.2.6.

Ideally, the addition of SEnSeI to a model will have no deleterious effects on its performance, whilst also making it sensor independent. In order to distil down the implications of SEnSeI, all the experimental results are tied back to the following hypotheses:

**Hyp. (A):** If SEnSeI is added to a deep learning model, there is no reduction in performance when trained and tested on data from a single sensor.

**Hyp. (B):** If SEnSeI is added to a deep learning model, the model can be trained on multiple sensors without a drop in performance relative to when it is trained on an individual sensor.

**Hyp. (C):** SEnSeI can enable a model to perform adequately on a previously unseen sensor, given training data that provide close equivalents to the unseen sensor’s spectral bands.

Both a single-pixel neural network, and a large convolutional model (DeepLabv3+ [65]) are combined with SEnSeI, using various training set configurations. These two models

show how SEnSeI interacts with as wide a range of models as possible, from the very simple to the very complex. The key findings in this chapter include:

- A novel model architecture, SEnSeI, is proposed. SEnSeI is a permutation invariant neural network, with unique features specifically designed for remote sensing data.
- DeepLabv3+ is shown to work extremely well as a cloud masking model, exhibiting higher performance than other existing techniques for several multispectral satellites.
- By combining SEnSeI with DeepLabv3+, training datasets can be combined to make models which work across many instruments, with a range of resolutions and spectral band combinations.
- Whilst in some instances, sensor independence leads to higher numerical performance (e.g. in the tests on Landsat 8 data), the primary benefit of adopting sensor independence with SEnSeI is the increase in a model’s usability. With SEnSeI, it is now possible to create reliable cloud masks for a huge family of multispectral instruments, even those which do not currently have labelled datasets.

After this section, the experimental setup, including model design, datasets and their preprocessing, and model training is detailed in Section 5.2. Then, quantitative test results on four satellites’ data, Sentinel-2, Landsat 8, Landsat 7 and PerúSat-1 are given in 5.3, along with visual results on Sentinel-3 SLSTR. Sections 5.4 and 5.5 discuss how these results fit with the aforementioned hypotheses, the possible implications this work has on deep learning in remote sensing, some of the future avenues of research this entails, and its limitations. The code associated with this work is publicly available [169].

## 5.2 Methods

### 5.2.1 SEnSeI Design

SEnSeI is an example of a PINN. This simply means that the output of SEnSeI is independent of the order of the inputs. In practical terms, this invariance is achieved through a simple averaging over the outputted features corresponding to each input. Imagine a neural network ingesting  $N$  input vectors, and thus outputting  $N$  feature vectors, which are then pooled to make a single feature vector. This averaged feature space is the shared representation that SEnSeI passes to a standard deep learning model. Importantly, this

can only be done because of the fixed size of the representation outputted by SEnSeI; an output which changed shape based on the inputs would not be compatible with standard deep learning models.

The structure of satellite data is the primary driver for the design of SEnSeI. In essence, satellite data can be considered as a set of bands, each with an extended spatial map of reflectances, and a spectral profile. This set of bands is the set of input points to SEnSeI, and its output is a matrix of the same spatial dimensions as the original bands, but with a fixed number of channels which does not depend on the number of input bands. The precise nature of this representation is, of course, non-linear and determined through stochastic training of the network, meaning the output channels do not necessarily have a well-defined physical analogue, or strong associations with any one input band.

Essentially then, SEnSeI’s task is to take each channel, and represent them all in a way that preserves the information about their reflectance and their wavelength, and also ensures that when pooled, that information is not degraded or confused with other channels. If trained on a fixed set of input channels, it is easy to see how this architecture could overfit, given that the spectral profiles it fits to would always be identical. To avoid this, random subsets of the available channels are used to increase variation, as described further in Section 5.2.2.

The spectral profile of each band is parameterised as a ‘*descriptor vector*’. This consists of a vector with three values: The lower, central and upper wavelengths of the spectral band in question. The central wavelength is taken as the peak of the spectral response curve if known, or simply the halfway point between lower and upper wavelengths. The wavelength values are normalised, with further details of the procedure found in Section 5.2.5. Figure 5.1 visualises the spectral bands of the sensors used in this experiment.

The architecture of SEnSeI can be split into 4 sub-modules, detailed in Figure 5.2. First, the ‘*DESCRIPTOR BLOCK*’ comprises a neural network with several fully connected layers, with ReLU and Group Normalization layers between them. 9 layers were used, with a final output vector of 64 features. At this point, there are  $N$  vectors being outputted from the ‘*DESCRIPTOR BLOCK*’, each corresponding to one of the  $N$  bands at input. No information from the reflectance values, nor other bands’ descriptor vectors, are used in these calculations. Simply, each of the  $N$  descriptor vectors (of length 3) enter a 9-layer neural network, which transforms them each into vectors of length 64.

Next, the ‘*PERMUTATION BLOCK*’ takes each possible pair of the  $N$  outputs from the previous module, and concatenates them together into  $N^2$  128-dimensional feature vectors, where each band is paired with all  $N$  bands, including itself. The 128 dimensions

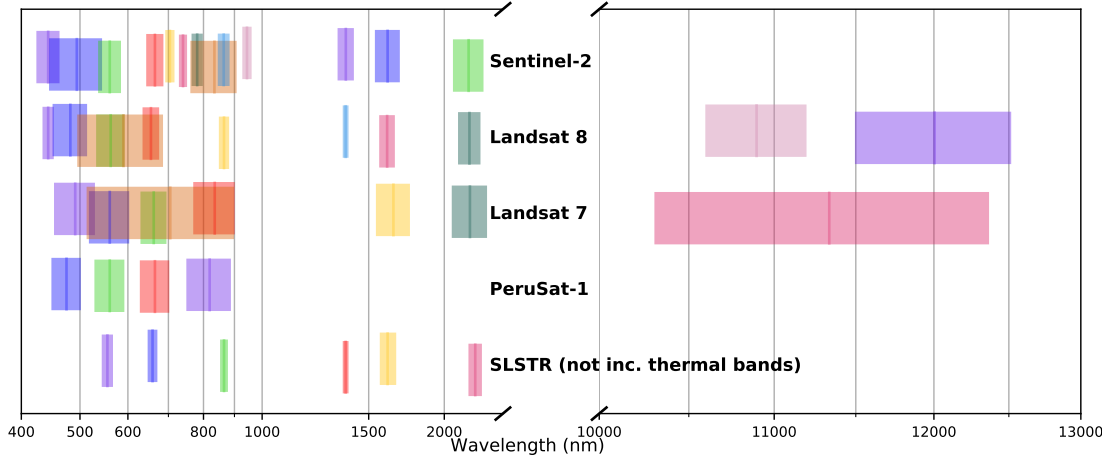


Figure 5.1: The band spectra of the instruments used in this chapter. This shows the diversity of sensors used, and underlines the need for sensor independence. The values here represent the FWHM of the bands, but should not be taken as exact. The descriptor vector corresponding to each band comprises its minimum, central, and maximum wavelengths. SLSTR’s thermal bands are not shown as they are not used in this study’s experiments. Colours and small vertical offsets are used arbitrarily, to help make neighbouring and overlapping bands more distinct.

simply come from the two 64-dimensional vectors concatenated together. These are then fed through 5 fully connected layers, and outputted as 128-dimensional vectors. To reduce the number of vectors from  $N^2$  back down to  $N$ , a pooling operation must be performed. By averaging across the  $N$  pairs associated with each of the bands, the number of feature vectors is reduced back down to  $N$ .

Third, a ‘*COMBINED DESCRIPTOR BLOCK*’ does essentially the same as the ‘*DESCRIPTOR BLOCK*’ module, but this time acts on the outputs of the previous module, again producing a 64-dimensional feature vector for each of the  $N$  inputs. This block is the final set of neural network layers in SEnSel.

At this stage, SEnSel has not used any of the reflectance band information. To do so within a neural network would require us to compute the outputs of each layer across each pixel of the input image, as the pixels do not all have the same reflectances. This could become very computationally intensive for large images. To get around this, the reflectance information is introduced with the ‘*BAND MULTIPLICATION BLOCK*’. This module takes the 64-dimensional output feature vectors of the ‘*COMBINED DESCRIPTOR BLOCK*’ and multiplies each by the bands’ corresponding reflectance values with an added offset of 0.5 (so that information from low reflectance values are still included in the output), producing an  $N$ -by- $X$ -by- $Y$ -by-64 feature map. This multiplication is computationally cheap and so can be carried out easily over each pixel.

The entire operation, up to this point, can be thought of as the transformation  $\mathcal{F}$  from (Eqn 2.9). If we were to swap any two inputs, say, Red and Green, in Figure 5.2, then we

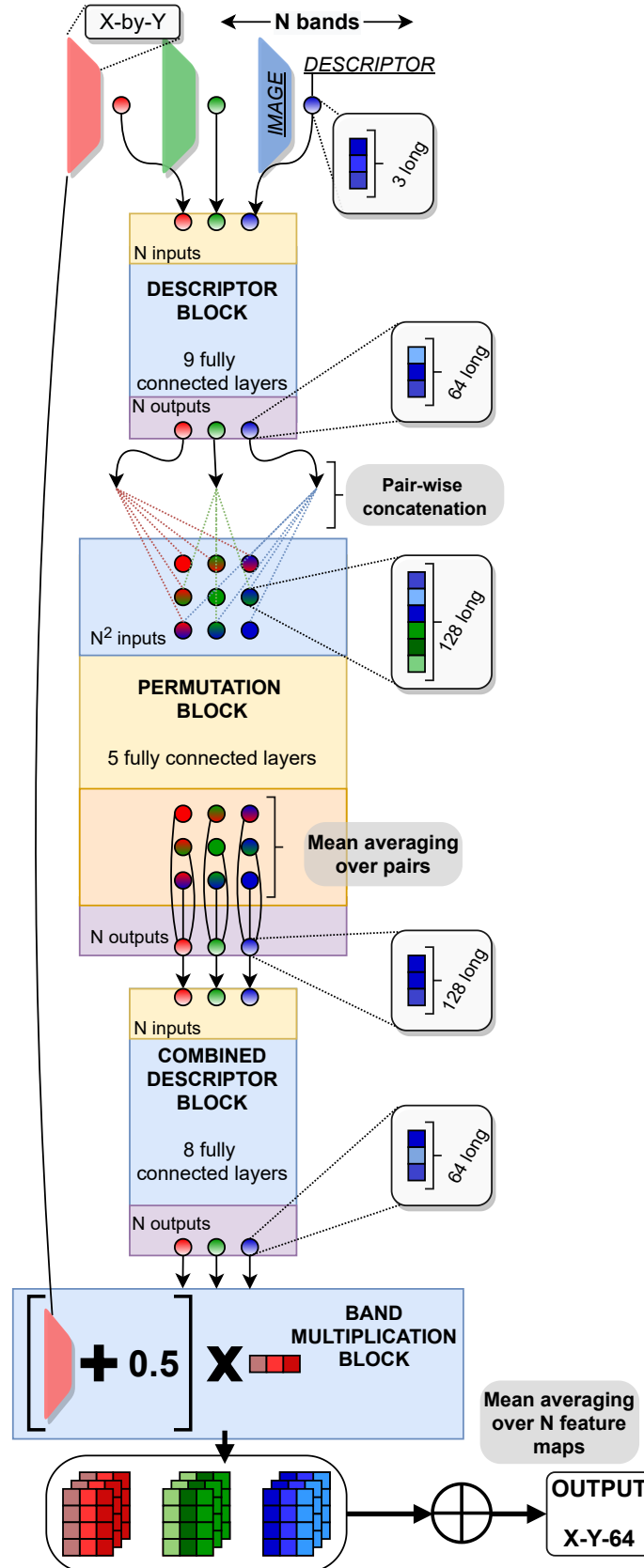


Figure 5.2: Schematic of SEnSel. Inputs are  $N$  images with  $N$  corresponding descriptor vectors (circles), with only 3 (red, green and blue) shown here for simplicity. Band multiplication creates feature maps which have the same extent as the input image, wherein each pixel is represented by a 64-dimension vector, scaled by the band's reflectance in that position, plus an offset of 0.5 such that low reflectances are still represented. The entire operation is permutation invariant, given that the final mean averaging over the feature maps does not depend on the inputs' order.

can see that the order of the outputs would be changed in the same way, hence satisfying (Eqn 2.10). Finally, there is a pooling operation,  $\mathcal{P}$ , applied to the set of feature maps: A mean averaging over the  $N$  feature maps, producing an X-by-Y-by-64 feature space that can be directly inputted into any other model, in this case DeepLabv3+. Whilst SEnSeI does not fuse information between pixels, the spatial features one finds in the input image are also present in the outputted feature map, because each pixel's feature vector is affected by its input reflectances. This means that models attached to SEnSeI still have access to that spatial information.

Although this approach is applied specifically to multispectral data, SEnSeI could easily be adapted to work with a range of different data types. Essentially, any data structure for which there is a set of images, each with a corresponding group of descriptors (in this case wavelengths) that can be parameterised.

Whilst SEnSeI has many neural network layers, computation is incredibly cheap, because the neural networks act only on the descriptor vectors, rather than the much larger image data. The final band multiplication operation is the only one which scales with the spatial dimensions of the image data. During supervised training there was only a 5% slow down per iteration compared to the standard DeepLabv3+ model. When using the models to make predictions after training, SEnSeI did lower the speed by a small amount, but only when many bands were used (see Appendix A).

### 5.2.2 SEnSeI Pre-training

The weights for SEnSeI were pre-trained before the supervised training of cloud masks, as part of an autoencoder architecture. There were two primary aims in this exercise: (i) optimise SEnSeI's design, by testing many different architectures, and (ii) pre-train the weights of SEnSeI for use later in the supervised training of SEnSeI with DeepLabv3+.

Autoencoders are commonly used to train models in an unsupervised manner, as is done here. The autoencoder setup involves *encoding* the inputs with SEnSeI, and then *decoding* those encoded features with a decoder. In this case, there were two, parallel decoder modules. At this point, SEnSeI is not being trained for cloud masking, but simply its ability to represent the data from different spectral bands in a coherent manner. The autoencoder training setup is outlined in Figure 5.3.

The first decoder module, known as the *discriminator*, predicts whether or not a given reflectance band was included in the input. It outputs a confidence that a band was in the original input, doing so for twice as many candidate descriptors as were in the original, such that half its answers should be true (for those which were in the inputs),



and half false (for those which were not). The loss for the discriminator,  $\mathcal{L}_{discriminator}$ , is the mean squared error of the predictions. The  $i^{th}$  truth value is 1 if that descriptor was present in the original input, making its contribution to the loss  $(1 - p_i)^2$  where  $p_i$  is the prediction. For fake descriptors which weren't in the original, the truth value is 0, making the term in the loss  $p_i^2$ . These two cases lead to the function for  $\mathcal{L}_{discriminator}$  seen in Figure 5.3.

Second, the *estimator* takes the inputted descriptors, and determines what the reflectance value of each band was, based on the output of SEnSeI. These two different decoders lead SEnSeI to be optimised both in its ability to represent many spectral bands simultaneously without overlap and confusion, and to retain information about the reflectance values in each of those bands. The loss for this branch,  $\mathcal{L}_{estimator}$ , is also the mean square error. This is calculated as the square of the difference between each band's true reflectance, and the predicted reflectance outputted by the estimator (Figure 5.3).

Both of these losses take the form of a mean square error function. For the estimator this is an intuitive choice, because the mean square error heavily penalises outliers due to the quadratic term, which encourages the model to make many small inaccuracies across the reflectance values, rather than a few very large errors. For the discriminator, given the unusual form of the loss (wherein half the values are expected to be 1, and the other half 0) the mean square error offered a straightforward formula for the desired behaviour, and was thus the most practical loss to use.

As input, SEnSeI was given a set of between 3 and 14 randomly generated descriptor vectors, alongside randomly generated reflectance values associated to each one. The descriptor vectors were created so that they represented realistic spectral bands: central wavelengths between 400nm–20 $\mu$ m, covering the range of typical multispectral instrument bands (see Figure 5.1), and bandwidths also tuned to cover typical cases. Having outputted a combined feature vector, the two decoder modules (simple neural networks with several fully connected layers) performed their discrimination and estimation. Once trained, the discriminator and estimator are no longer needed and are discarded, leaving only SEnSeI to be used in the supervised training of the cloud masking model.

The specific architecture of SEnSeI was chosen during this unsupervised training, by minimizing the loss with respect to the number of layers in each block, number of hidden units in each layer, types of pooling operation, and so on. However, like many neural network architectures, performance was not hugely sensitive to changes in these hyperparameters, and so the final version of SEnSeI was chosen from a wide set of possible configurations with similar performance. Figure 5.2 shows that architecture, as selected by this unsupervised training.

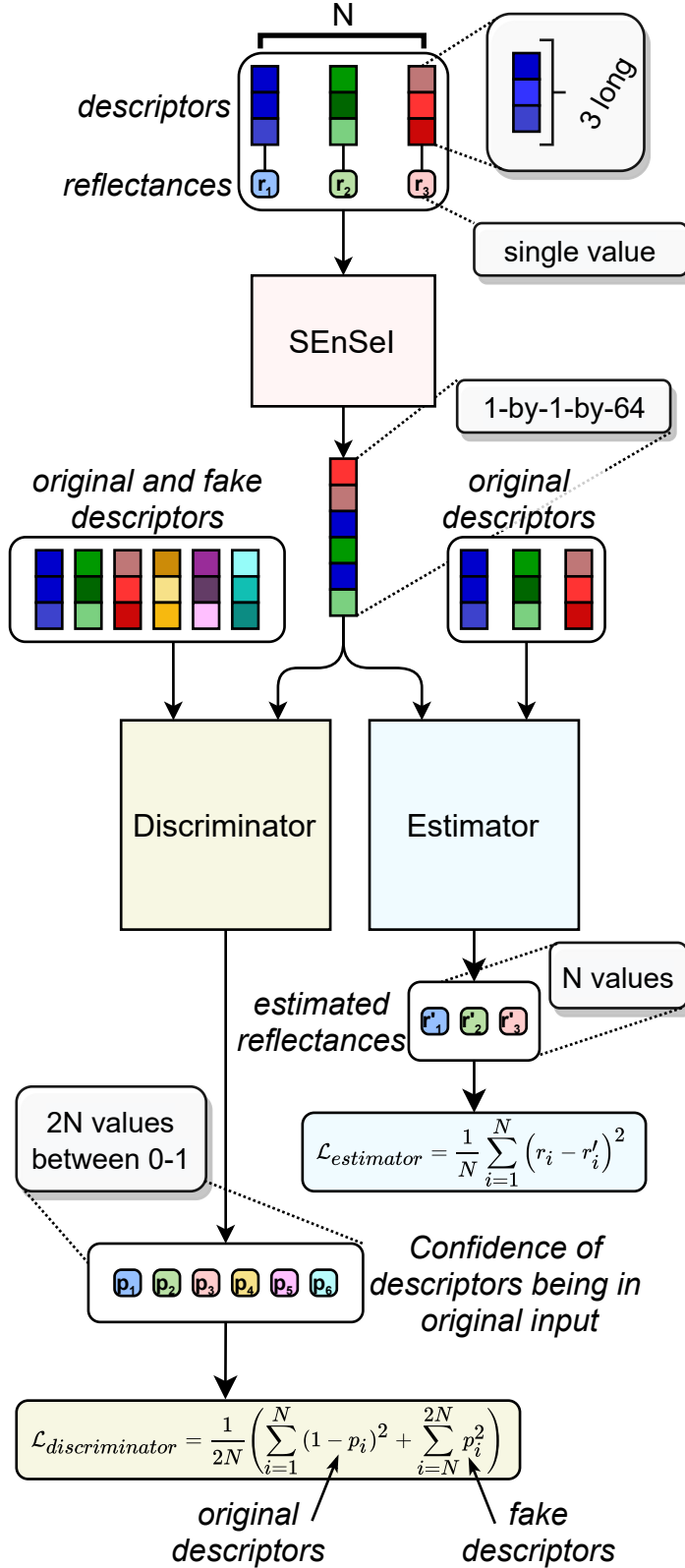


Figure 5.3: Flowchart for unsupervised autoencoder training of SEnSel. A set of  $N$  inputs comprising pairs of random descriptor vectors and reflectance values are processed by SEnSel. Then, two decoder branches, the discriminator and estimator, attempt to recover information about the inputs based on SEnSel's output. The discriminator learns to identify which descriptor vectors were present in the input, whilst the estimator predicts the reflectance values associated with each descriptor. Both branches are trained using the mean square error function. When used with a convolutional neural network, SEnSel takes images, rather than single reflectance values, however during unsupervised training single values are preferable, as they allow for larger batch sizes.

### 5.2.3 DeepLabv3+

Section 2.2.4 gives a technical overview of the DeepLab models. The implementation of DeepLabv3+ used here is in *TensorFlow 2* [170] and is based heavily on [171]. The first convolutional layer of the model is modified to accept inputs with more than the standard 3 RGB channels—64 when used with SEnSeI, or the number of bands on a satellite if used without SEnSeI.

Supervised training of DeepLabv3+ models was identical across experiments, other than whether or not SEnSeI was included as a preprocessing module, and which datasets were used. For the experiments in this chapter, a desktop machine with an NVidia RTX 2080 Super (8GB) GPU. The models were initialised with pre-trained weights in SEnSeI, derived using the unsupervised training described in Section 5.2.2, and MobileNetv2 with weights from a pre-trained version of DeepLabv3+ on the PASCAL VOC training set [172]. All images were preprocessed according to Section 5.2.5, and entered the model at 257x257 pixels across, with a batch size of 8. The model was trained using SGD with an initial learning rate of  $1e-3$  and a momentum of 0.99. The initial learning rate was reduced by a factor of 4 when the loss converged, down to a minimum learning rate of  $2e-5$ . Each epoch was 1000 steps, with a maximum of 200 epochs (although most models were stopped after around 100 epochs as they had converged on a stable loss rate). Training of the models took several hours, with each epoch taking roughly 5 minutes to complete. Models with SEnSeI took around 5% longer per epoch, but also tended to take 10–20 more epochs to fully converge.

### 5.2.4 Datasets

The experiments in this chapter use several labelled datasets, some of which have already been described in Chapters 3 and 4. The Landsat 8 CCA described in Section 3.4.1 and the Sentinel-2 cloud mask catalogue introduced in Chapter 4 are both used. Some other datasets are also used, and so are described here.

#### Landsat 8 SPARCS

The Spatial Procedures for Automated Removal of Clouds and Shadows (SPARCS) validation dataset [126] consists of 80 manually created masks, each over a 1000 pixel squared subscene taken from a Landsat 8 product. As well as cloud and cloud shadow, the masks differentiate between land, water, snow/ice, and flooded surfaces. We combine these different surface types in the mask, to make the labels simply clear vs. cloud.

SPARCS is distributed in GeoTIFF format, with all bands and the mask sampled at 30m. The dataset does not contain Landsat’s Panchromatic band, but includes all other 10 bands from Landsat 8. Of the 80 scenes, many show complex and varied cloud structures, suggesting the images were hand-picked. The lack of entirely cloudy or clear scenes means the dataset is not representative of real-world use on the entire Landsat 8 catalogue, but provides a challenging and diverse benchmark for models to be tested on.

### Landsat 7 Cloud Cover Assessment

This dataset [165], also produced by the USGS, follows a similar format to the Landsat 8 CCA dataset. 206 scenes are included, but are grouped by latitude, rather than by surface type. The classes used in annotation are also the same as those used for the Landsat 8 CCA, however the percentage of scenes annotated with cloud shadow is somewhat lower than the Landsat 8 dataset. Additionally, through visual inspection it was found that some thin clouds were not marked, compared to similarly thin clouds that were marked in other datasets. During preprocessing of the dataset, it was not possible to recover 9 of the 206 scenes, due to corrupt files, leaving a total of 197 for our experiments.

### CloudPeru2

PerúSat-1 is a cubesat mission launched by the Peruvian space agency (CONIDA). The sensor samples data at 2.8 m resolution, in 4 spectral bands; red, green, blue and NIR. The CloudPeru2 dataset [168] comprises 22,000 images, each 512x512 pixels across, taken from 153 scenes over Peru. Alongside each image is a hand-drawn mask with cloud vs. non-cloud pixels. This dataset has been used to train and test cloud masking algorithms before [168], although in these experiments it is treated only as a test set, and is not used for training. This dataset—and those discussed previously—are also listed in Table 4.1.

## 5.2.5 Preprocessing and Formatting

For training, each image is split into 263-by-263 cropped patches and saved. Whilst DeepLabv3+ was configured to use 257-by-257 images as input, the crops were made slightly larger to introduce some variation to the samples through random translation, whilst still being small enough to be read in quickly during training. All reflectance values were converted from their native ranges to double precision floating point values between 0–1, although some pixels have reflectances higher than 1, if there is specular reflection, or if the surface is at an angle which reflects more than the normal plane could.

For the thermal bands of Landsat 7 and 8, the radiances were converted to Brightness Temperatures (BTs), using (Eqn 5.1) and the parameters  $K_1$  and  $K_2$  in their product specifications. Then, these BTs were normalised between 0 and 1, such that a value of 1 corresponded to the 95th percentile of the total available data (Eqn 5.2). This was done in order to create a similar distribution of values between thermal and non-thermal bands,

$$BT = \frac{K_2}{\ln\left(\frac{K_1}{\text{Radiance}} + 1\right)} \quad (5.1)$$

$$BT_{normalised} = \frac{BT - BT_{min}}{BT_{95\%} - BT_{min}} \quad (5.2)$$

Alongside each image and mask pair, the wavelengths of the spectral bands held in the image were also saved, to be used as the descriptor vectors which SEnSeI takes as inputs. This was simply an array with the lower, central, and upper wavelengths of each band in nanometres, as defined in their product specifications. For input into SEnSeI, the wavelengths in nanometres were normalised to a logarithmic scale, as the linear differences between e.g. red and green are far smaller than those in the SWIR or TIR range, therefore

$$\lambda_{normalised} = \log_{10}(\lambda - 300) - 2 \quad (5.3)$$

During training, data augmentation was applied as it has been shown to help prevent over-fitting, and improve model performance [173]. This included random flipping, rotation, translation, addition of white noise, salt-and-pepper noise, and small random shifts in overall reflectances across the image's bands.

## 5.3 Experimental Results

### 5.3.1 Sentinel-2

This experiment was conducted to measure the performance of both the standard DeepLabv3+ model, and the sensor independent variants with SEnSeI prepending DeepLabv3+, alongside some other cloud masking options that are widely used. The three main goals of this experiment were:

1. Demonstrate a state-of-the-art cloud masking model for Sentinel-2.

2. Investigate whether Hyp. (A) and Hyp. (B) from Section 5.1 would hold, or if the addition of SEnSeI, and an increasingly generalised training configuration, would cause performance to suffer against the specialised, non-sensor independent model.
3. Determine how a deep learning-based approach compared to other popular Sentinel-2 cloud masking algorithms.

The Sentinel-2 dataset was split randomly into training, validation, and testing sets, such that 40% (205 subscenes) were in training, 10% (51 subscenes) in validation, and 50% (257 subscenes) in testing. The training for each model was then carried out as Section 5.2.5 details.

Two versions of DeepLabv3+ were trained without SEnSeI. First, a version simply trained with Sentinel-2 data, using all bands (referred to as ‘*DLv3 - S2*’ in the experiments). Second, a version trained on both the Sentinel-2 training data, and the Landsat 8 CCA dataset, using only those bands which are found on both Sentinel-2 and Landsat 8, similar to [125]), as mentioned in Section 2.3. This model is referred to as ‘*DLv3 - S2&L8 [common bands]*’. The common bands—in the naming scheme of Sentinel-2—were bands B1, B2, B3, B4, B8, B10, B11, and B12.

Four different configurations of DeepLabv3+ were used for SEnSeI. All four of these had the same architecture. Three began training with the same pre-trained weights in both SEnSeI and DeepLabv3+. These three configurations are increasingly sensor independent, adding to the ability of the model to generalise to other sensors. Another configuration used a SEnSeI with randomly initialised weights, to see what advantage pre-training offered:

1. ‘*SEnSeI+DLv3 - S2 [fixed bands]*’ was given all Sentinel-2 bands at every training step. This means it was only ever given Sentinel-2 data with all bands present, thus producing a model that could not be generalised to work on any other satellite or subset of Sentinel-2 bands, but containing the SEnSeI architecture.
2. ‘*SEnSeI+DLv3 - S2*’, was trained using random combinations of the available Sentinel-2 bands, from a minimum of 3, up to a maximum of all 13. This additional complexity in training produces a model which is theoretically capable of predicting clouds from sensors with any combination of the existing Sentinel-2 bands (this assumption of sensor independence will be tested further in Section 5.3.3).
3. ‘*SEnSeI+DLv3 - S2&L8*’, was also given the Landsat 8 CCA dataset as training data. It then received random combinations of the available bands on each satellite, creating what is shown in later experiments to be a truly sensor independent model.

4. ‘*SEnSeI+DLv3 - S2CL8 [no pre-training of SEnSeI]*’, was the same as the previous but the weights in SEnSeI were randomly initialised.

As well as DeepLabv3+, SEnSeI was combined with a 2-layer neural network, which has no convolutions. This neural network has only a few hundred parameters, and represents an extremely simple machine learning solution. By combining SEnSeI with both this small neural network, and the very complex DeepLabv3+, it can be shown that SEnSeI works with a wide range of model architectures. The naming conventions for the neural network model configurations are the same as for DeepLabv3+, with “NN” replacing “DLv3”.

In addition to DeepLabv3+ and the neural network, an attempt to train CloudFCN [1] (Chapter 3 with SEnSeI was made, to further show SEnSeI’s general applicability; however the training became unstable when they were used together, and it was not possible to get the models to reliably converge. The reason for this is not clear, but it may be that, when used with SEnSeI, CloudFCN would need to be trained with a larger batch size than was possible on the 8GB of video memory used in training. This is an avenue for future research, as it may be that SEnSeI adds some instability to certain model architectures (perhaps those with an architecture similar to U-Net [58], like CloudFCN) during training. However, results from CloudFCN without SEnSeI in this experiment are included, trained on the Sentinel-2 dataset.

Results from *s2cloudless* [174] are also given, using a confidence threshold of 0.5. This algorithm is a pixel-based machine learning model, which uses the XGBoost algorithm [175].

Lastly, the *Standard L1C product mask* is the cloud mask generated during L1C processing [176]. A rasterized version of the masks were made, derived from the polygonal information in the Geography Markup Language (GML) file provided with each L1C product. All “OPAQUE” and “CIRRUS” polygons were included. The full product masks were then cropped to the extent of the subscene, and visually inspected to confirm they had been correctly processed and were aligned with the image. Comparison with this mask allows us to judge the relative performance of a deep learning-based model to a thresholding model.

To assess each model’s performance in this experiment, five metrics are calculated. OA, BA, Precision, Recall, and  $F_1$ , as defined in Section 2.2.7. These are calculated across the entire Sentinel-2 test set. Table 5.1 provides numerical values for these metrics over the dataset. Figure 5.4 gives some visual examples of the masks produced by the models.

Model	OA	BA	Prec.	Rec.	F <sub>1</sub>
DLv3 - S2	94.04	94.03	94.57	94.17	94.37
DLv3 - S2&L8 [common bands]	<b>95.19</b>	<b>95.18</b>	95.61	<b>95.31</b>	<b>95.46</b>
SEnSel+DLv3 - S2 [fixed bands]	94.46	94.42	94.58	94.99	94.79
SEnSel+DLv3 - S2	93.08	93.10	94.00	92.89	93.44
SEnSel+DLv3 - S2&L8	93.73	93.73	94.42	93.73	94.07
SEnSel+DLv3 - S2&L8 [no pre-training of SEnSel]	93.11	93.16	94.47	92.43	93.44
NN - S2	90.04	90.04	91.06	90.08	90.57
NN - S2&L8 [common bands]	89.86	89.98	92.50	88.02	90.20
SEnSel+NN - S2	88.90	88.75	88.19	91.31	89.72
SEnSel+NN - S2&L8	89.04	89.16	91.70	87.23	89.41
CloudFCN - S2 [1]	91.58	91.67	93.58	90.34	91.93
s2cloudless [174]	92.42	92.61	<b>95.90</b>	89.54	92.61
Standard L1C product mask [176]	83.11	83.29	86.88	80.27	83.45

Table 5.1: The five performance metrics for each model, across the 257 Sentinel-2 subscenes used as a test set. The highest performance for each metric is highlighted in bold. All variants of DeepLabv3+ outperform the other models in OA, BA, Recall and  $F_1$ , however s2cloudless does achieve the highest precision. The red arrows show model comparisons relevant to Hyp. (A), whilst the blue arrows indicate those for Hyp. (B).

Appendix B gives a further breakdown of results for the models across different image types.

Overall, *DLv3 - S2&L8 [common bands]* performs better than the other models. This shows that transfer learning techniques (as used by others, e.g. [114, 124, 125]) can result in boosts to performance, despite the removal of the bands which Landsat 8 does not also include. Besides this model, *SEnSel+DLv3 - S2 [fixed bands]* also exceeds the performance of *DLv3 - S2* in all metrics, showing that prepending SEnSel to a deep learning model does not produce a significant drop in performance, supporting Hyp. (A). When a more challenging training is used, in which band combinations were sampled randomly, performance does drop somewhat (*SEnSel+DLv3 - S2*). Promisingly, however, performance of *SEnSel+DLv3 - S2&L8* is higher than that of *SEnSel+DLv3 - S2* in every metric, making it clear that adding data from more than one satellite boosts



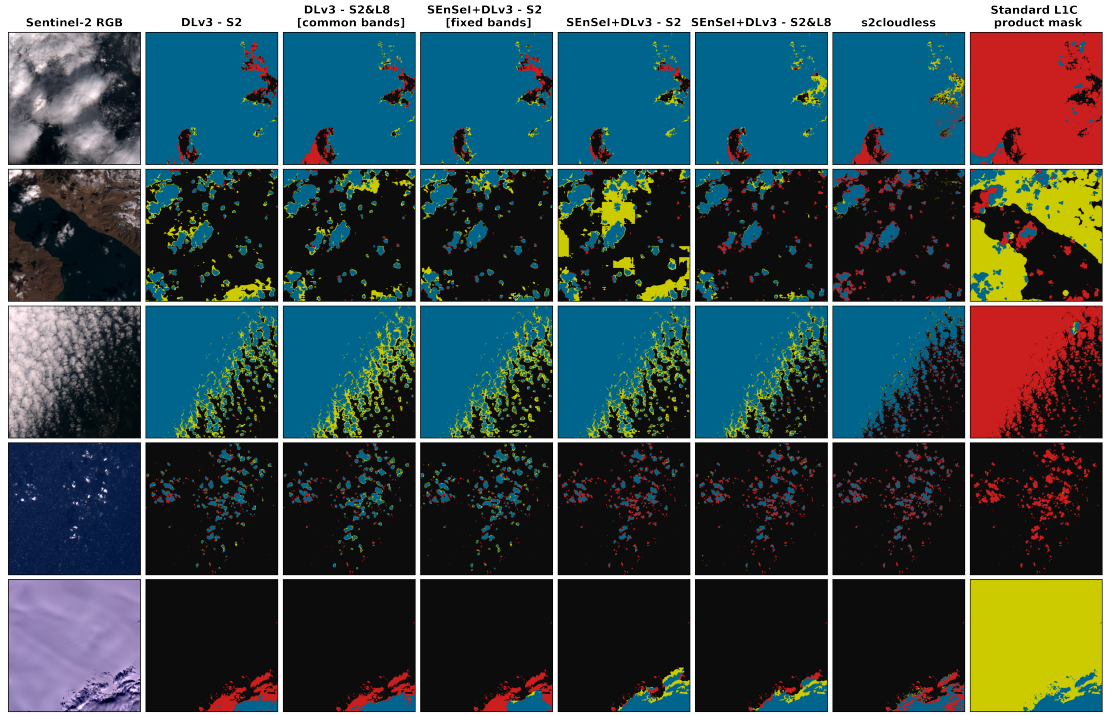


Figure 5.4: Visualisation of seven models’ predictions tested on five scenes from the Sentinel-2 test set, each 1022 pixels (20.4 km) across. Blue is correctly identified cloud, red is missed cloud, and yellow is false cloud detection. These scenes were selected as they contained interesting, diverse and challenging cloud structures.

the model’s reliability. There is a small decrease in performance of roughly 0.6% in OA, BA, and  $F_1$  when using a randomly initialised version of SEnSel, rather than the pre-trained weights. This suggests pre-training SEnSel is worthwhile, and has a moderate but noticeable impact on performance.

SEnSel’s effect on DeepLabv3+ is mirrored closely by its effect on the 2-layer neural network. The creation of a sensor independent model leads to a drop of about 1% from  $NN - S2$  to  $SEnSel+NN - S2$  in OA, BA and  $F_1$ . This 1% drop in performance (on test data from Sentinel-2) seems to be the cost of making a model which works across many sensors when using SEnSel. Interestingly, the addition of data from Landsat 8 did not boost the neural network’s performance, like it did for DeepLabv3+, suggesting that such a small, simple model as this cannot benefit from an expanded training set, whereas DeepLabv3+ can.

In order to determine whether the differences in performance seen in Table 5.1 are statistically significant or not, a cross-validation and bootstrapping experiment to investigate the uncertainties in the results was done. Due to computational limitations in training many models, it was only possible to analyse the uncertainty for just two of the model configurations ( $DLv3 - S2$  and  $SEnSel+DLv3 - S2\&L8$ ). Although not all models were included, it seems reasonable that given their architectures, training data, and the

test set are identical to those not included, this analysis can be safely extrapolated to approximate the uncertainty for all our model results. Each model configuration was five-fold cross-validated, to train five different models, each with a different 40% of the data as training, and 10% as validation. The same 257 scenes are always held as the test set for all models. The test set is then bootstrapped, with 1 million iterations per model, to measure the data-driven uncertainty in our performance. With this method, both models tested were found to have a range of  $\pm 0.5\%$  in OA, BA and  $F_1$ . This range is taken as a rough limit for statistical significance in the analysis of the results.

### 5.3.2 Landsat 8

This section presents results on the Landsat 8 SPARCS dataset (described in Section 5.2.4). In testing these models, true sensor independence is demonstrated (a model working on multiple satellite sensors at once), and show strong evidence for Hyp. (B) and Hyp. (C). In addition, this experiment is an opportunity to investigate the effects of different input bands, by testing the same models with different spectral band combinations. The first model, *DLv3 - L8*, is trained using the 96 scenes from the Landsat 8 CCA dataset, but without the panchromatic band (to make the model compatible with the SPARCS dataset). *DLv3 - S2&L8 [common bands]* is also used in this experiment, and is the same model as that used in Section 5.3.1. Then, in a similar fashion to Section 5.3.1, there are two configurations of DeepLabv3+ with SEnSeI:

1. ‘**SEnSeI+DLv3 - S2**’ is the same model (with the exact same weights) as used in Section 5.3.1 with mixed band combinations as input in training. No retraining takes place to prepare it for use with Landsat 8, however, bands that Sentinel-2 does not have (thermal bands) are not used in testing.
2. ‘**SEnSeI+DLv3 - S2&L8**’ is also the identical model to that which is used in Section 5.3.1, with no retraining.

For both of these SEnSeI configurations, tests are run on SPARCS using several different band combinations. For 10 bands, there are 968 possible combinations of 3 or more bands. Therefore, only a few of interest are selected:

1. ‘**ALL**’: All Landsat 8 bands (excluding the panchromatic band, which is not distributed with the SPARCS dataset).
2. ‘**COMMON**’: All bands shared by Sentinel-2 and Landsat 8. SPARCS already excludes Landsat 8’s panchromatic band, so this band combination is just all available bands except the two thermal bands.

3. ‘**RGB**’: Bands 2, 3, and 4.
4. ‘**AeroRGB**’: Bands 1, 2, 3, and 4.
5. ‘**AeroRGBNIR**’: Bands 1, 2, 3, 4 and 5.
6. ‘**No-RGB**’: All possible bands except 2, 3 and 4.

As well as our own models’ results, this experiment also includes Fmask, the standard Landsat 8 cloud masking software, and RS-Net [177], another state-of-the-art deep learning model discussed in Section 2.3, which was trained on several different band combinations of Landsat 8. Additionally, the results of Cloud-Net+ is compared, when trained on their 38-Cloud and 95-Cloud datasets and tested on SPARCS. Rather than computing predictions for these three models, they are simply recovered from prior studies ( [177] for Fmask and RS-Net, [167] for Cloud-Net+). For Fmask, there are differing results on the SPARCS dataset in different studies, possibly because different versions of the code of were used, and so the best published result for the SPARCS dataset is taken [177].

The results of this experiment are shown in Table 5.2, and some of the models can be compared visually in Figure 5.5. Out of the models not trained on SPARCS data, *SEnSeI+DLv3 - S2&L8 (COMMON)* achieves the highest OA and  $F_1$  score, although it is beaten by models trained using 5-fold cross-validation on SPARCS. Perhaps a more important result for Hyp. (C), though, is the performance of *SEnSeI+DLv3 - S2* across the different band combinations, which shows a model performing with  $> 90\%$  OA on data from a sensor it has not seen in training. Falling just short of Fmask in OA, this shows adequate performance for a model working on a different sensor. Finally, the significant increase in performance between *DLv3 - L8* and *SEnSeI+DLv3 - S2&L8*, shows that adding data from another satellite (Sentinel-2) *increases* performance on Landsat 8, agreeing with Hyp. (B). Not only is SENSEI allowing for models to become more generalised and work across different sensors, but, in this instance, it is allowing models to enhance their predictive capabilities on one sensor with data from another.

The change in performance of the same models with different band combinations is relatively small. For example, OA for most models varies by around 1-2%, whilst the  $F_1$  score is slightly more sensitive, varying by around 2-3%. Generally, models with limited band combinations, e.g. RGB and RGBNIR, suffered in comparison to more extensive band combinations, which makes sense given that they do not have access to as much spectral information from their inputs. Surprisingly, for all three models which were tested both with and without the thermal infrared bands (*SEnSeI+DLv3 - S2&L8*, and

Model	Bands	OA	BA	Precision	Recall	F <sub>1</sub>
DLv3-L8	ALL	93.03	83.32	95.17	67.46	78.96
DLv3-S2&L8	COMMON	93.70	87.04	89.82	76.14	82.42
SEnSel+DLv3 S2	COMMON	91.86	<b>93.05</b>	71.96	<b>94.99</b>	81.89
	RGB	90.91	91.77	69.92	93.18	79.89
	RGBNIR	91.11	92.23	70.18	94.06	80.38
	AeroRGBNIR	91.11	92.45	70.02	94.64	80.49
	No-RGB	91.52	92.45	71.34	93.96	81.10
SEnSel+DLv3 S2&L8	ALL	93.91	86.31	93.24	73.90	82.45
	COMMON	<b>94.24</b>	90.98	84.77	85.65	<b>85.21</b>
	RGB	92.45	92.34	74.73	92.18	82.54
	RGBNIR	93.51	92.29	79.12	90.31	84.34
	AeroRGBNIR	93.63	91.67	80.54	88.47	84.32
	No-RGB	93.83	86.19	93.00	73.71	82.24
RS-Net trained on Biome [177]	ALL	92.53	–	88.57	70.53	78.35
	COMMON	93.26	–	91.04	72.34	80.62
	RGBNIR	92.53	–	<b>95.35</b>	64.56	76.99
	RGB	92.38	–	86.54	71.83	78.50
RS-Net trained on SPARCS (5-fold C.V.) [177]	ALL	94.54	–	87.62	83.66	85.59
	COMMON	<b>95.60</b>	–	89.47	87.58	85.59
	RGBNIR	94.85	–	88.92	83.89	86.33
	RGB	94.86	–	88.58	84.34	<b>86.41</b>
Cloud-Net+ trained on 38-Cloud [167]	RGBNIR	91.30	86.47	76.94	78.60	77.76
Cloud-Net+ trained on 95-Cloud [167]	RGBNIR	90.45	85.69	74.14	77.90	75.97
Fmask [177]	–	92.47	–	77.47	86.21	81.61

Table 5.2: Results of models across the Landsat 8 SPARCS dataset. The versions of RS-Net which are highlighted were tested on SPARCS using 5-fold cross-validation, and thus have an advantage over other models in that they have been exposed to the style and distribution of SPARCS at training. Therefore, in metrics for which this version of RS-Net is best, the best model *not* trained on SPARCS is also highlighted. A version of Cloud-Net+ was trained using 64 SPARCS images and tested on 16, but is omitted here as the results do not cover all 80 scenes. The red arrow denotes a model comparison relevant to Hyp. (A) and the blue arrow to Hyp. (B). Meanwhile, the results next to the green line are relevant to Hyp. (C).

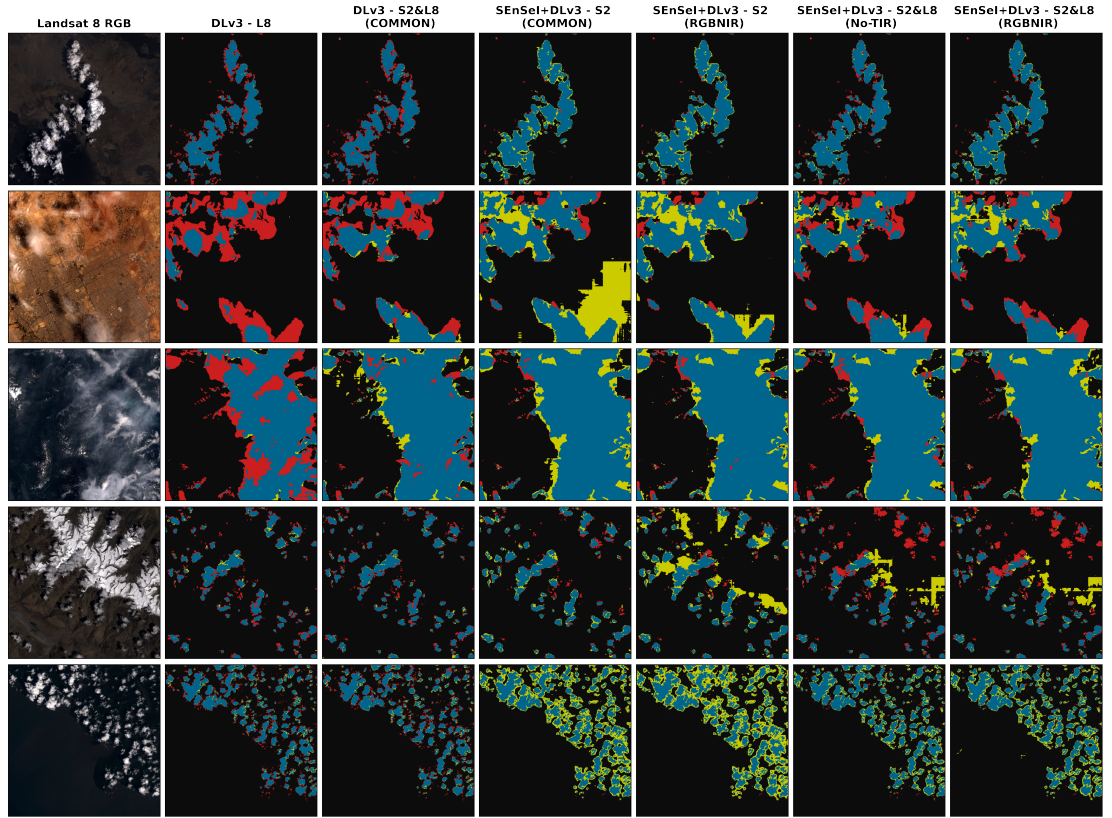


Figure 5.5: Visualisation of six models’ predictions tested on five scenes from the Landsat 8 SPARCS test set, each 1000 pixels (30 km) across. Blue is correctly identified cloud, red is missed cloud, and yellow is false cloud detection. *DLv3 - L8* shows much higher precision and lower recall than other models. The boundaries of the thin cloud in the 2nd row are understandably error-prone, given the smooth transition between cloudy and clear. Some models misclassify parts the snow-covered mountains in the 4th row as cloud, but successfully reject the snow on the left of the 3rd row’s scene.

both versions of RS-Net), performance actually *decreased* when they were included. This is surprising, given that thermal bands provide quite different information to the others, which would ordinarily be a good thing for a model to have.

### 5.3.3 Other Satellites

The experiments in this section seeks to test the limits of SEnSel’s sensor independence, by predicting cloud masks for satellites not included in the model’s training, and different resolutions to the 20–30 m/pixel range of Sentinel-2 and Landsat 8. Three sensors are included in the following experiments, two of which (Landsat 7 and PerúSat-1) have datasets that allowed us to measure our models’ performance, and another (Sentinel-3 SLSTR) for which only visual inspection is done, as no suitable labelled dataset was available.

Landsat 7’s non-thermal bands all have close equivalents in Sentinel-2 and Landsat 8. It’s thermal band, whilst not directly shared by Landsat 8, spans roughly the equivalent

Model	Bands	OA	BA	Precision	Recall	F1
SEnSel+DLv3 S2	RGB	76.23	73.17	64.27	64.05	64.16
	COMMON	85.24	86.25	72.61	89.26	80.08
	No-RGB	85.25	86.63	72.09	<b>90.73</b>	80.34
SEnSel+DLv3 S2&L8	RGB	74.62	71.72	61.52	63.06	62.28
	ALL	89.08	87.77	83.35	83.88	83.62
	No-RGB	<b>89.90</b>	<b>88.90</b>	<b>84.03</b>	85.92	<b>84.96</b>
C5 CCA [178]	-	88.5	-	-	-	-
AT-ACCA [178]	-	76.6	-	-	-	-
Expanded AT-ACCA [178]	-	89.7	-	-	-	-

Table 5.3: Results for models over the 197 scenes from the Landsat 7 CCA dataset. Models using only RGB made substantial mistakes, perhaps due to saturation in Landsat 7 images. The combination of all bands other than RGB (No-RGB) were generally the best, and the model trained on both Sentinel-2 and Landsat 8 data is generally better than the Sentinel-2-only model.

of both of Landsat 8 TIR bands, and so can be approximately simulated by averaging them together. Whilst the spectral profiles and resolution of Landsat 7 are familiar to our model, the sensor design and image quality differ between Landsat 7 and the other satellites. An 8-bit digitisation depth was used for Landsat 7, unlike the larger bit depths on more recent missions, which meant that some of the higher reflectances are saturated, in order to allow for more radiometric precision at lower reflectances. This leads some high reflectance surfaces such as snow, ice and clouds, to have a textureless appearance, as many pixels are at the maximum of the range.

The large Landsat 7 CCA dataset (described further in Section 5.2.4) is used to test SEnSel’s performance. As before, there are versions of SEnSel trained either on Sentinel-2, or Sentinel-2 and Landsat 8, and compare the test set results with different band combinations as input. These are compared to the results published in 2012 [178], which present accuracy scores for three models over 103 of the Landsat 7 CCA images. As well as the numerical findings in Table 5.3, some of the failures and successes are visualised in Figure 5.6.

Overall, the models are reasonably successful on Landsat 7, achieving accuracies close to 90%, with the best version just out-performing the highest reported accuracy from [178]. However, there is a noticeable increase in error rate when compared to our results on Sentinel-2 and Landsat 8. In particular, the model appeared to struggle in regions of the image which were uniformly high reflectance, and when it was given only



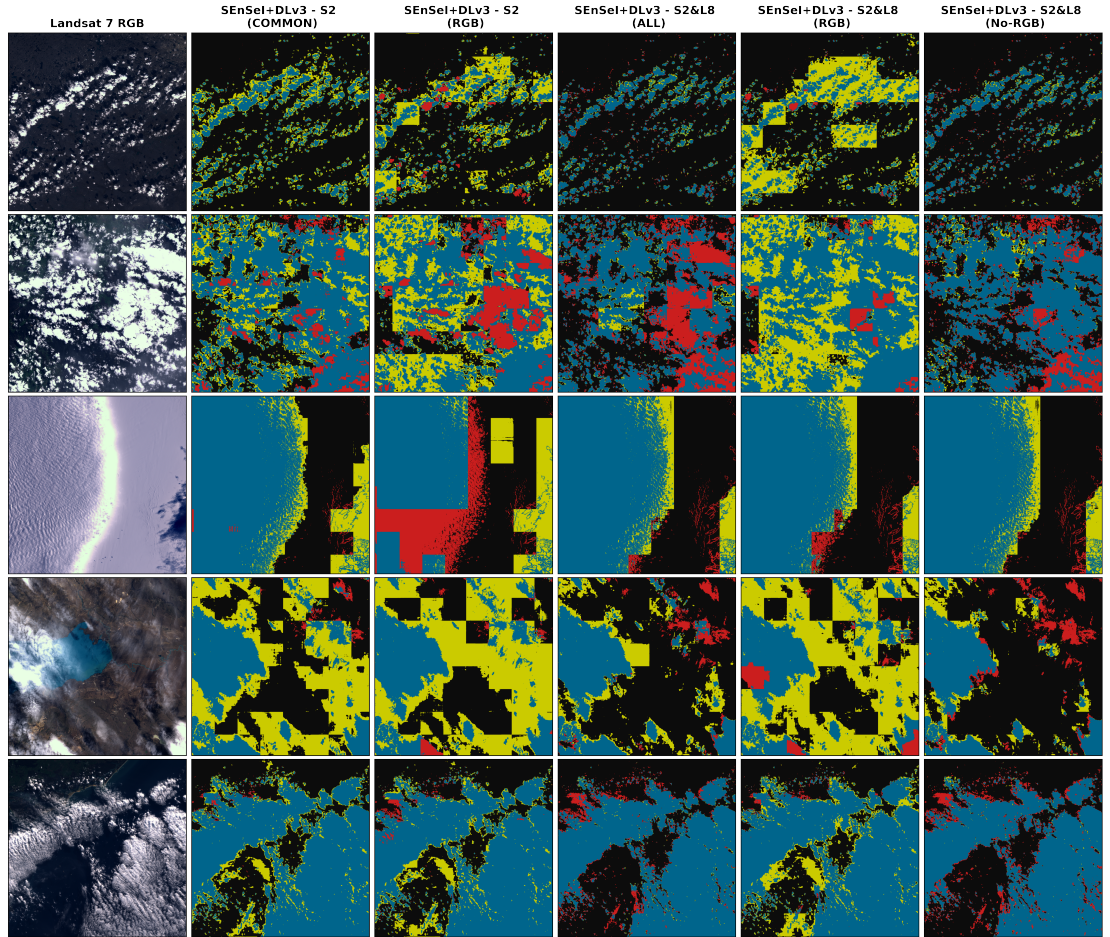


Figure 5.6: Visualisation of five models' predictions tested on 6 scenes from the Landsat 7 CCA test set. 2000 pixel (60 km) squares are taken from the scenes, as the full scenes are too detailed to display in this manner. Blue is correctly identified cloud, red is missed cloud, and yellow is false cloud detection. Models using RGB bands are inconsistent, while those using other bands are more successful. The fourth row shows an image with very thin cloud which is picked up by some of the models, but is not annotated as cloud.

RGB bands. This may have been caused by Landsat 7’s tendency to saturate pixels with high reflectances, leading to a lack of texture which perhaps confused the model. It may be possible to mitigate these errors by adding artificial saturation to images in training. Many of the errors in the Landsat 7 experiment appear as in square blocks. These are a sign that the model has low confidence (as in, predictions are close to the confidence threshold) and that within each 256-by-256 block, the model is predicting roughly the same value. Because of this, errors often appear in large blocks of 256-by-256 pixels.

Another likely reason for reduced accuracy on Landsat 7 is the discrepancy in annotation styles between datasets, with the Landsat 7 CCA dataset classifying as clear some areas that would be considered thin cloud in other datasets, which was also noted by Scaramuzza et al. [178]. This means that a model trained on the Sentinel-2 and Landsat 8 datasets is biased in relation to the Landsat 7 CCA labels. This experiment was still partially successful, given the challenges, but more work is needed on understanding why the model failed where it did, and why in this instance Hyp. (C) is not well supported.

PerúSat-1’s resolution is an order of magnitude finer than that of Sentinel-2 and Landsat 8, at 2.8 m/pixel. The CloudPeru2 dataset (Section 5.2.4) was used to see how a sensor independent model copes with a significant change in scale. The model results are given in Table 5.4. In order to investigate how a change in scale affects results, cloud masks are made at both the original resolution (512-by-512) and at 50% downsampling (256-by-256), with an effective resolution of 5.6 m/pixel.

Direct comparison with the other models included in the table is difficult, as those considered by Morales et al. [168] were all trained on 90% of the dataset, and tested only on 5%, giving their models a significant advantage in having been exposed to the same distribution of data as the test set. Given this difference in approach, our results—especially at 5.6 m resolution—are excellent, out-performing some of the models trained specifically on the dataset. The drop in performance seen at 2.8 m is perhaps a sign that the limit of the model’s resolution range is being reached, having only been trained on a resolution of 20 m/pixel or more.

Finally, DeepLabv3+ was used over a scene from Sentinel-3’s SLSTR. The six bands S1–S6 were taken as input, as these have close equivalents in Sentinel-2 and Landsat 8 from Green into the SWIR. They are all at 500 m resolution, and can be easily converted into reflectance values using the Sentinel Application Platform (SNAP) rad2refl preprocessing tool. Clouds were predicted on the nadir-view stripe “A” data, and the results presented in Figure 5.7. As in our experiment using PerúSat-1, predictions at different effective resolutions are made. This time, however, the images are upsampled, artificially raising their resolutions from 500 m, to 250 m and 167 m. This was done because the



Model	Bands	OA	BA	Precision	Recall	F1
SEnSel-DLv3 S2 (2.8 m)	RGB	90.86	90.90	89.03	92.71	90.83
	RGBNIR	90.95	91.00	89.04	92.92	90.94
SEnSel-DLv3 S2&L8 (2.8 m)	RGB	91.63	91.68	89.78	<b>93.52</b>	91.61
	RGBNIR	91.52	91.56	89.97	93.01	91.47
SEnSel-DLv3 S2 (5.6 m)	RGB	92.22	92.20	92.41	91.59	92.00
	RGBNIR	92.51	92.50	92.72	91.88	92.30
SEnSel-DLv3 S2&L8 (5.6m)	RGB	92.70	92.67	93.49	91.42	<b>92.44</b>
	RGBNIR	<b>92.74</b>	<b>92.69</b>	<b>94.36</b>	90.54	92.41
CloudNet [179] (2.8 m)	RGBNIR	94.01	93.91	<b>97.82</b>	89.78	93.63
[180]'s method (2.8 m)	RGBNIR	91.34	91.97	86.52	92.36	89.34
[168]'s method (2.8 m)	RGBNIR	<b>97.50</b>	<b>97.52</b>	96.45	<b>98.46</b>	<b>97.44</b>

Table 5.4: Results of models across the CloudPeru2 dataset, from PerúSat-1. The DeepLabv3+ models are tested at the original resolution (2.8 m) and 5.6 m. The masks at the coarser resolution are then bilinearly resampled to retrieve a mask at the full 2.8 m resolution. The models examined by Morales et al. [168] are included, but direct comparison to them is not strictly valid, given that they tested the models on 5% of the available data, using the rest for training.

original 500 m data contains very high-frequency clouds not seen in the 20–30 m data the model was trained on. What is more, because DeepLabv3’s output is interpolated over an output stride of 8, it would not be able to segment these very small clouds successfully.

Whilst the results for SLSTR are only visual, they show great promise. At higher spatial resolutions, especially, the models seem able to accurately segment clouds, at a resolution roughly 5-10 times coarser than they were trained on, up to 250 m/pixel. Whilst the development and the testing of SEnSel focused on independence with respect to spectral band combinations, these results from SLSTR (and those from PerúSat-1) show SEnSel-based models can work well across large resolution ranges, which massively increases the number of sensors they can be applied to without any retraining.

## 5.4 Discussion

### 5.4.1 Findings

Developing SEnSel has produced a highly performant cloud masking algorithm that achieved OA greater than 92% on 3 test sets from 3 different satellites, Sentinel-2, Landsat

8, and PerúSat-1, and with visually promising results on Sentinel-3's SLSTR instrument, all without any retraining whatsoever. This proven compatibility with multiple sensors, beyond those used in training, is a novel result in the field of deep learning for remote sensing.

The experimental results do not unambiguously show SEnSeI to always offer an advantage to models in terms of statistical performance. Whilst the experiment on Landsat 8 demonstrated that sensor independence *can* lead to improvements in performance, the Sentinel-2 experiment also showed that sometimes an approach similar to [125] (as covered in Section 2.3) can be more successful. In both cases, though, the use of aggregated training data from multiple satellites was shown to be preferable to using data from a

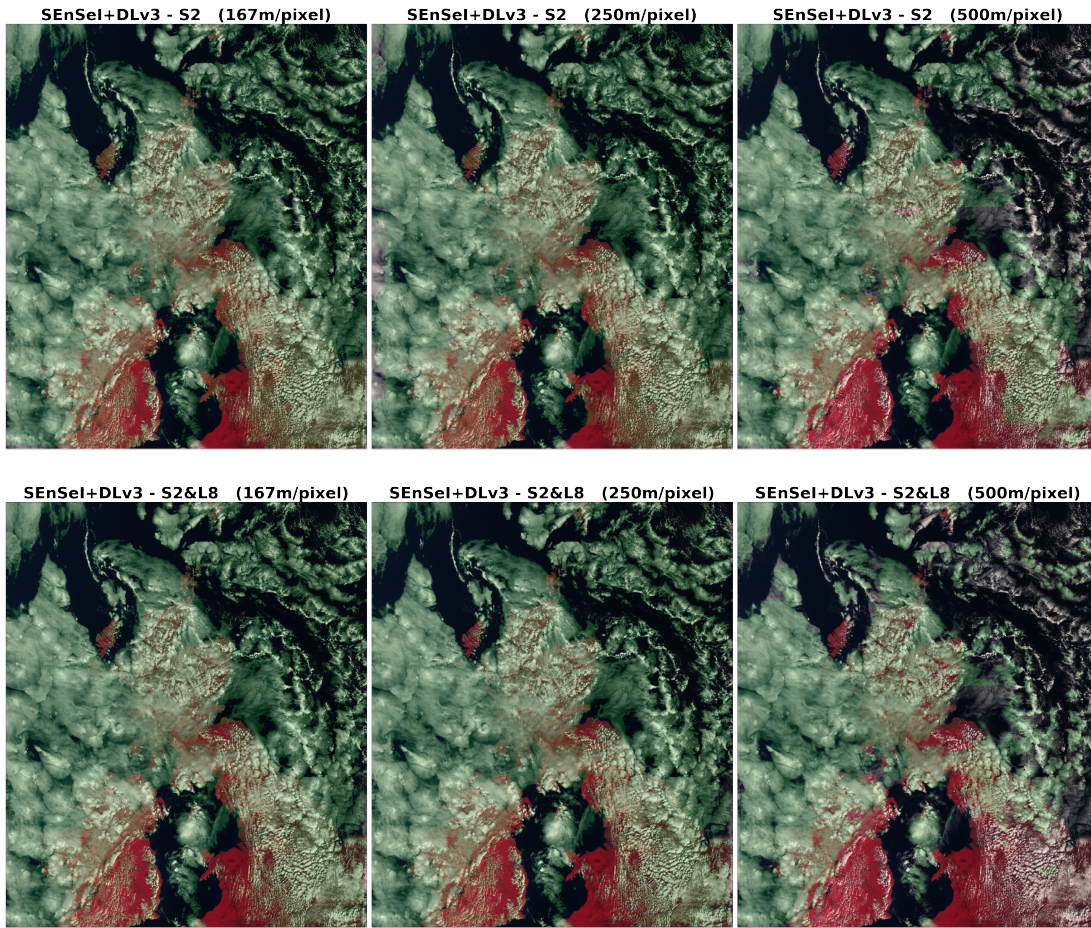


Figure 5.7: Predictions of *SEnSel+DLv3 - S2* and *SEnSel+DLv3 - S2&L8* for a Sentinel-3 scene over the UK and Ireland from August 2020. The scene has been cropped slightly for better visualisation. It is displayed here in false colour with S3 as Red, S2 as Green, and S1 as Blue. The green overlay used for detections does not necessarily indicate correct detection, because there is no ground-truth for this scene. Masks produced at the original resolution of 500 m fail in areas with high frequency clouds, whereas at 250 m and 167 m, these are segmented more successfully. Some visible improvements in performance can be observed in the *SEnSel+DLv3 - S2&L8* predictions, e.g. the false detections over Cardigan Bay, Wales made by the model trained only on Sentinel-2 are avoided. This figure is best viewed digitally in colour.

single satellite. Based on this, it seems that further work into sensor independence—and interoperability more generally—is vital for continuing to improve performance and expand usability of deep learning models in remote sensing.

Throughout Section 5.3, it was pointed out how the results correspond to the three hypotheses from Section 5.1. Here, we consider each hypothesis in turn and pull together the outcomes of our experiments in relation to them:

**Hyp. A:** *“If SEnSeI is added to a deep learning model, there is no reduction in performance when trained and tested on data from a single sensor”.*

Whilst there is some good evidence for this hypothesis being validated in our results, it is somewhat nuanced. The six DeepLabv3+ models tested on Sentinel-2—four with SEnSeI and two without—show a spread of results which fall slightly outside the  $\pm 0.5\%$  range that was measured using cross-validation on one of the models. Clearly, the increase in performance from *DLv3 - S2* to *SEnSeI+DLv3 - S2 [fixed bands]* shows that adding SEnSeI does not necessarily cause a drop in performance, with the OA actually increasing by 0.42%, though this is just below what was deemed to be a statistically significant difference of 0.5%.

However, the drop in performance for *SEnSeI+DLv3 - S2* is significant, at around 1%—and is consistent with the 1% drop seen for between *NN - S2* and *SEnSeI+NN - S2*. Consider, though, that this model has been trained to produce cloud masks on any combination of 3 or more Sentinel-2 bands, which is a much more general problem than one in which all bands are always present. So, this experiment is evidence that SEnSeI has no deleterious effects on performance *when the training task is the same*, but that there is a small cost to performance (1%) when using SEnSeI to produce a more generalised, sensor independent model.

**Hyp. B:** *“If SEnSeI is added to a deep learning model, the model can be trained on multiple sensors without a drop in performance relative to when it is trained on an individual sensor.”*

To fully appreciate whether this hypothesis holds, we must consider two cases. First, what effect adding Landsat 8 data in training has on Sentinel-2 test performance, and second, the reciprocal case, when Sentinel-2 data is added, what affect is seen on Landsat 8 test performance. In the prior case, the performance of *SEnSeI+DLv3 - S2&L8* actually shows a moderate increase (0.65% in OA, 0.63% in BA, and 0.63% in  $F_1$  score) over the equivalent model trained on Sentinel-2 alone, *SEnSeI+DLv3 - S2*. This gain in performance, though, does not quite make up for the initial decrease of 1% seen when adding SEnSeI with random band combinations.

In the second case, on Landsat 8 test data, the results are even more positive. The performance of *SEnSeI+DLv3 - S2&L8 (COMMON)* is significantly higher than the performance of the specialised, non-sensor independent Landsat 8 model, *DLv3 - L8*. OA is 1.21% higher, BA is 7.66% higher, and  $F_1$  6.25% higher. This seems to have been driven by the models trained only on the Landsat 8 CCA dataset (including those published by others) having highly imbalanced Precision and Recall values. This may be because SPARCS does not have the same sampling distribution as the Landsat 8 CCA dataset, as it focuses on clouds which are not extended across the whole scene, and which have complex and diverse structures. This difference in the data distribution between training and test sets appears to lead to models which are too ‘picky’ and miss many of the SPARCS dataset’s clouds. The introduction of Sentinel-2 data at training has produced more balanced predictions on the SPARCS dataset, with Precision and Recall much more equal. This is promising evidence for the power of sensor independence; training on data from more sources, with different labellers and sensors, produces models which are more robust to changes in data sampling, and predictions which draw on information from a wider range of situations. Therefore, the Landsat 8 SPARCS results strongly support the hypothesis, and indeed extends beyond it, by showing in this instance that performance on one sensor, Landsat 8, can be improved by training data from another sensor, Sentinel-2.

**Hyp. C:** *“SEnSeI can enable a model to perform adequately on a previously unseen sensor, given training data that provide close equivalents to the unseen sensor’s spectral bands.”*

Several times in this work, models are shown to perform well on sensors not included in training. First, let us consider the performance of *SEnSeI+DLv3 - S2* in the experiment on the Landsat 8 SPARCS dataset. Across several band combinations, OA remained above 90%, with the highest being for the band combination using all possible bands (except thermal bands, which were not available during training and so cannot be included in inference). This model achieved the highest BA of any model (93.05%), and had very high recall but low precision—the inverse of the models trained solely on the Landsat 8 CCA dataset.

Landsat 7 presented difficulties for our models regarding Hyp. (C), which shows the often unpredictable nature of deep learning models when used in unfamiliar settings. In the case of Landsat 7, the significant saturation over high reflectance areas (often clouds) led to a flat texture that the model was unfamiliar with, and thus struggled to correctly classify. As we see in Figure 5.6, high reflectance, saturated areas (e.g. the thick clouds in the 2nd row in Figure 5.6) are sometimes mis-classified. Another issue is a difference

in annotation style, with some thin clouds that the model detected not being included in the ground-truth (e.g. the thin clouds not annotated as such in the 4th row in Figure 5.6).

Meanwhile, at resolution scales outside those that the model was trained on, SEnSel performed surprisingly well. For the finer resolution data in the CloudPeru2 dataset, sensor independent models achieved OA consistently above 92% at 5.6 m, with a likely drop off in performance in the range of 3–5 m per pixel, with predictions at 2.8 m showing a roughly 1% decrease in performance relative to 5.6 m results.

At the coarser resolution range of Sentinel-3’s SLSTR, no suitable labelled dataset could be found, however using our model on a product over the UK shows that visually impressive predictions can be made, although only by artificially upsampling the scene, which is somewhat inefficient. Predictions seem to become less reliable beyond 250 m/pixel, with the model failing to produce good results at 500 m/pixel. This limitation may be related to the structure of clouds at this resolution, with the high frequency changes in cloud cover being too fine for DeepLabv3+’s output stride of 8 pixels to segment properly, leading to an averaging over class predictions and thus either all-cloud or all-clear predictions over areas with both clear and cloudy pixels.

### 5.4.2 Dataset Design

One factor which greatly affects the practicality of sensor independence, is the homogeneity of labelling styles, and data sampling schemes, between datasets. In some of our experiments, differences in annotation style may have caused drops in performance, notably for Landsat 7. This does not mean either style of annotation was “wrong” per se, but rather that they have chosen to define what is and isn’t included in the category of ‘cloud’ differently. In other experiments, such as those using Landsat 8 SPARCS as a test set, the imbalanced Precision and Recall of some models compared to others is evidence that the data sampling scheme and class distribution of training data can have significant impacts on performance on a fixed test set. This issue was also noted by Jeppesen et al. [177].

Where possible, datasets should be designed with a focus on compatibility with prior efforts, both in terms of their labelling styles, and in their file formats and structure. Not only will this help to accelerate sensor independent model design by reducing the time spent on preprocessing work for each sensor and dataset, but will also allow studies with non-sensor independent designs to rapidly train and test on multiple datasets. Additionally, dataset creators should provide as much information as possible on the sampling of

the data (how it was selected, what the edge cases are in class labels, etc.), so that users can understand what a model’s performance in a given dataset is representative of.

## 5.5 Interim Conclusions

SEnSeI is a possible route to building sensor independence into existing models for remote sensing applications. By treating each spectral band as a member of a set of all possible bands, and using a permutation invariant network design, SEnSeI translates any satellite’s measured reflectances and BTs into a common representation. The framework is presented in the context of cloud masking, and is shown to work well with both a simple neural network, and the large convolutional model, DeepLabv3+. It enables DeepLabv3+ to simultaneously predict clouds over five different satellites without re-training, and using all available bands. Furthermore, the performance in Landsat 8 was significantly higher when trained on data from Sentinel-2 as well as Landsat 8, showing that a better model comes when using training data sampled from multiple sensors.

The sensor independent models appeared to struggle when given data that is very unlike that used in training (e.g. Landsat 7 with its saturation, or resolutions very different from those in training). Nevertheless, the sensor independent model—not trained on any Landsat 7 data—had a comparable accuracy to purpose-built models [178] (Table 5.3). Future work could improve matters further by adding information beyond spectra to the inputted descriptors, e.g. saturation levels, Point Spread Function, noise, or resolution. This could allow models to better handle images with different statistics to those used in training, and help to expand training sets to include even more sensors.

Training a model to be sensor independent appears to cause some penalty to performance in some circumstances (e.g. in the Sentinel-2 experiment). However, this is not always the case, as the sensor independent model outperformed others on Landsat 8. The additional usability and versatility of the sensor independent model, being shown to work on many other satellites for which it was not directly trained, is the key advantage of sensor independence. Robustness to unfamiliar sensors is not only essential when training data does not exist for those satellites, but can also be incredibly helpful if deep learning is to be used operationally on-board future missions, where real training data cannot be created prior to launch, or in ground segment processing, where algorithms are usually developed before downlinking of data begins.

It should be emphasised that the main thrust of this chapter is not the raw performance of the proposed models. Whilst comparisons between competing methods are an

important part of machine learning research, these findings invite collaboration, not competition. SEnSeI is an accessory which can be added to existing models, making them sensor independent (although future work should seek to ascertain why it did not work well with CloudFCN, specifically). For many computer vision tasks in remote sensing, we see limited adoption of published techniques in real-world settings, even when performance is high [181]. This is often because the models are highly specialised to a specific sensor or dataset, and cannot be deployed beyond it. Pursuing sensor independence—with SEnSeI or something like it—can give models real-world utility, which may ultimately prove to be more important than further increasing their performance.





## 6 | Crater Detection: Initial Study

### 6.1 Motivation

As indicated in Section 2.4, surface chronology studies rely almost entirely on larger crater populations, given the unknown factors controlling sub-km crater populations, and their tendency to quickly reach an equilibrium. Perhaps naïvely, many publications that propose automated crater detection claim that aging is the primary motivation for their work. However, this rationale fails to take into account that for large craters that number in the hundreds or thousands, manual counting is trivial – and thus automation is pointless. Conversely, small craters are numerous, but currently not always helpful in age-dating. Therefore, we must put forward a scientific use-case independent of direct age-dating, otherwise its uptake in the community will be negligible, as has been seen for previous CDAs.

The processes that confound our understanding of small crater populations are numerous [140]. To begin to unravel and separate the effects of weathering, atmospheric filtering, partial resurfacing, dust cover and so on, the community will likely have to develop physical models. These models will need to be informed and validated by statistically significant CSFDs across numerous terrains, and perhaps on numerous terrestrial bodies. It is here we find a viable opportunity to apply automation to crater detection—in the counting of vast numbers of small craters across regions or entire planetary bodies, as will be pursued later, in Chapter 9.

This chapter documents an initial attempt to develop a deep learning-based CDA, from scratch. Whilst initially promising as a binary classification algorithm, converting it into a *detection* algorithm revealed difficulties that were only partially overcome. Nevertheless, this work motivated the later research described in Chapters 7 and 8, and the exercise provided some useful lessons. Instead of placing all the methods in one section, followed by all the experimental results in another, this chapter instead approaches the work in a roughly chronological fashion. First, an initial crater *classification* algorithm is described and tested. Second, this classification algorithm is converted into a full detection algorithm, and issues concerning performance and computation which then arose are discussed. Third, the attempts to resolve these issues are explained, and these improvements’ effects are explored. Finally, an experimental comparison is given between

the model before and after the improvements were implemented. However, before these technical sections there is a brief description of the dataset used in the study.

## 6.2 Data

Training and testing of the models developed in this chapter was all done using the same dataset. It comprises six 1700-by-1700 pixel tiles from HRSC product H0905\_0000, over the Nanedi Valles region (4.9°N, 49.0°W). The resolution is 12.5 m per pixel, with craters between around 120 m and 1 km annotated. In total, 2022 craters across the six tiles are marked, alongside another 2888 non-craters, and are defined by the coordinates of their centres, and their diameters. Originally, Urbach et al. [146] published a crater detection method using two of the six images as a validation set for their method. Later, Bandeira et al. [182] added to this, producing the full six image dataset as it exists today. An automated template-matching method was used to extract candidates, and these were then classified into the craters and non-craters. As such, craters not found by the candidate generation stage were not included in the dataset (adding some noise to the dataset in the form of missed craters).

The six tiles are arranged in three columns and two rows (see Figure 6.1). The columns are referred to as “West”, “Centre” and “East”, whilst the rows are “24” and “25”, derived from their vertical position in the original HRSC product they were taken from.

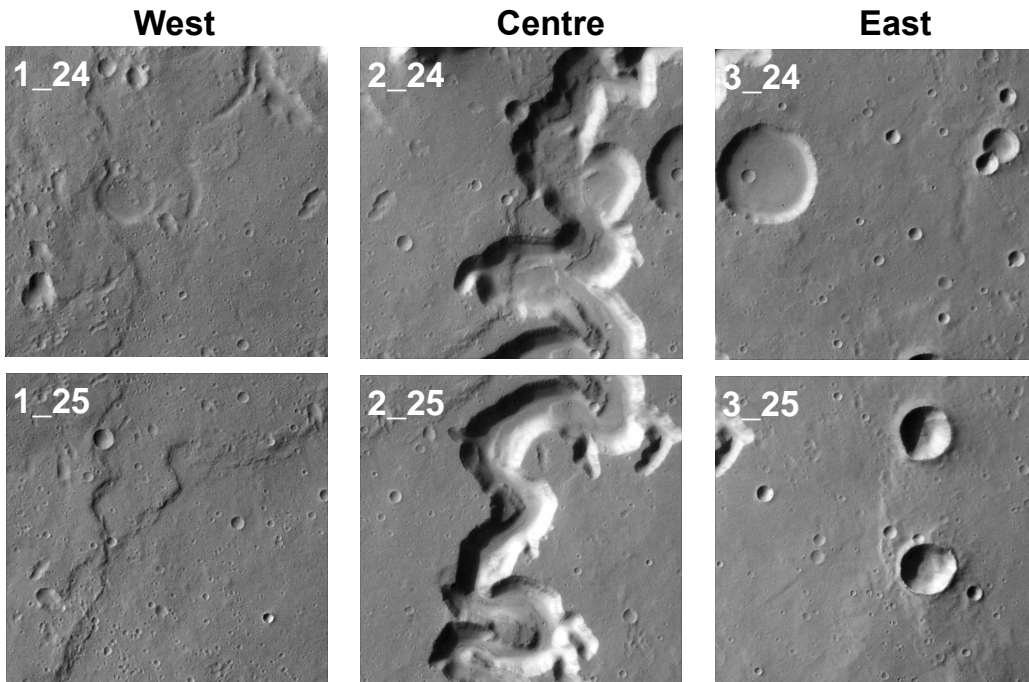


Figure 6.1: The six slightly overlapping HRSC images used for the dataset, taken from product H0905\_0000, over Nanedi Valles (the central canyon feature). Each image is 20.4 km across.

These three columns give a natural way to split the dataset into training and validation. These experiments follow the lead of others' work in order to make direct comparisons, by training on two of the three columns, and testing on the other. This was done three times, with each column used as the test data once, and results collated across these three runs.

Crater densities follow a negative power law with respect to diameter. This dataset is no exception, with few larger craters and more smaller ones. The CSFD for the dataset is shown in Figure 6.2. Unfortunately, an issue becomes apparent when considering this distribution, which is that craters smaller than around 250 m in diameter no longer follow this distribution, and in fact sharply decrease as the diameter becomes smaller. This is not because there are no craters below this diameter, but that the annotations miss many of the craters at lower diameters. Whilst the number of small craters on any surface will—at some diameter—plateau due to saturation [183], the sudden drop in

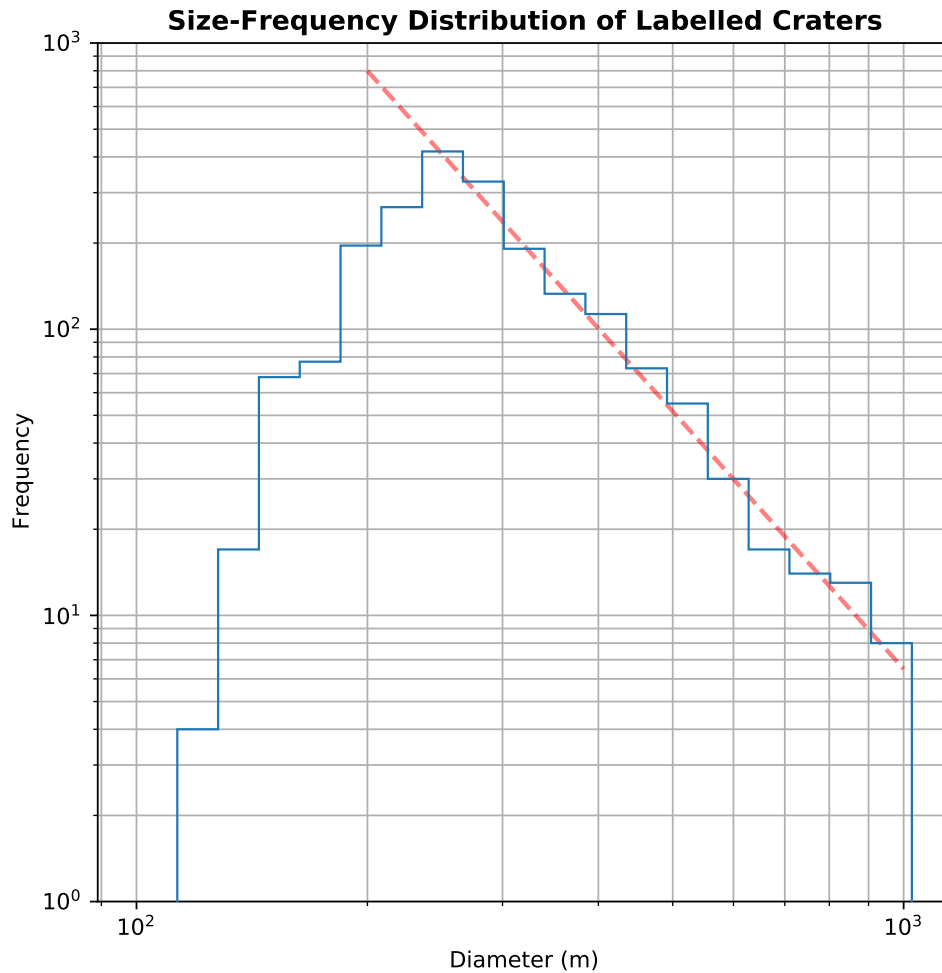


Figure 6.2: The labelled craters of the dataset are between around 120–1000 m diameter. The red dashed line indicates a power-law fitted to the distribution, which one would expect to continue below 250 m diameter. The fact that there is a sharp drop off in frequency below this indicates that many craters below 250 m diameter were missed during annotation.

density below 250 m is too steep and at too high a diameter to be consistent with this effect [183, 184].

This is a returning theme in my studies on crater detection—whatever camera system one is using, the majority of craters visible in the image are at or close to the limit of resolution, because there are always exponentially more small craters. This means that for every dataset, and every detection algorithm, craters close to the resolution limit will be responsible for the majority of errors and uncertainties.

Most datasets used in object detection problems only contain annotations for non-background classes, leaving the background passively annotated through omission. In this dataset, however, many non-crater features are annotated specifically as non-craters. This is because the dataset was produced using a semi-automated method. First, a template-matching method based on the shadows and highlights common to craters was used to extract candidate craters. This stage was designed with high recall and low precision (find almost all the craters, but also find lots of non-craters). Then, a human annotator classified each of these candidates as crater or non-crater. This gives us the opportunity to first tackle crater classification, rather than detection, which is a much more straightforward problem.

## 6.3 Classifying Craters

A helpful component of a full CDA is a module which can decide whether or not a given feature is a crater. This part of the pipeline is not responsible for locating where the crater is, but simply determines whether or not an image has a crater in it. Hence, the problem of crater detection over an extended image (“where are the craters?”) is simplified into a classification problem on a small part of the image (“is this a crater?”).

As explained in Section 2.6, a string of studies have used the same dataset [146] to develop crater *classification* models. This dataset comprises candidate craters generated by a shadow/highlight template matching method, and so classification of the candidates into positive or negative examples is what matters, as they have already been localised by this prior technique. Several studies have shown promising results using non-deep learning based approaches, ranging from traditional computer vision techniques, through to hybrid methods involving both hard-coded feature extraction and non-neural network machine learning models like SVM or random forest. Cohen et al. [151] proposed using a CNN to classify crater candidates and demonstrated higher accuracy than previous attempts.

### 6.3.1 Design

Like Cohen et al. [151], the method used here is also a CNN. The main difference is that the CNN developed here is pre-trained using unsupervised learning. This was motivated by the fact that the dataset consists of only 3000 craters and around 2000 non-craters, over only 6 images. By pre-training the network on Mars imagery as a denoising autoencoder [185], the model can benefit from a substantially larger amount of image data, converging on a latent representation which is less prone to overfitting on the relatively small labelled dataset.

Training of the network is undertaken in two stages. Firstly, the convolutional layers are trained as a denoising autoencoder [185] for image patches taken from a diverse range of Martian terrains, in HRSC imagery. These images were selected at random from across Mars. This allows the feature extractor to gain a general understanding of the parameter space that Martian terrain inhabits without the need for a labelled dataset.

Once this is complete, the decoder of the denoising autoencoder is discarded, and a set of 4 fully connected layers interspersed with dropout and ReLU layers are appended to the feature extractor, ending with a softmax activation on 2 outputs, in order to perform binary classification. Section 2.2 goes into more detail about how these layers function. This network is then exposed to the labelled set of craters and non-craters in supervised training.

The hyperparameters governing the architecture of this model—e.g. the number and types of layers, number of features per layer, input patch size—were all arrived at by using rules of thumb and trial-and-error. Undoubtedly, the final model (see Figure 6.3) is unlikely to be the optimal architecture. However, it was found that within reasonable bounds the hyperparameters had little effect on the performance of the classifier. A formal campaign of hyperparameter optimization would have likely been inconclusive, owing to the small labelled dataset available, which makes the performance differences that the hyperparameters are optimised with respect to statistically insignificant.

### 6.3.2 Results

To test the CNN’s ability to classify craters and non-craters, a 3-fold cross-validation was performed, as described in Section 6.2. Each crater and non-crater in the dataset was saved as a separate image, with the crater (or non-crater) in the centre of the frame, spanning 50% of the height and width of the 54-by-54 pixel image patch.

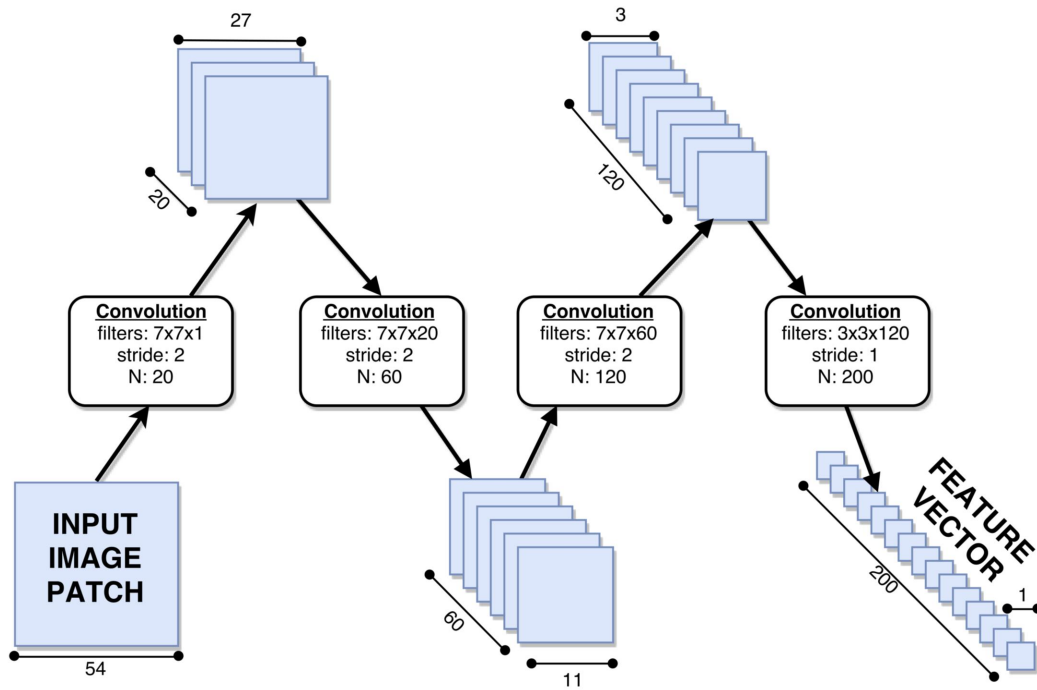


Figure 6.3: Flowchart of CNN used for crater classification. Data begins as a 2D image array, gradually becoming spatially more compact and with a larger number of channels (features). Non-convolutional layers are omitted, such as LeakyReLUs, batch-normalisations, drop-out and dimensionality reduction layers. During unsupervised training, a decoder takes these features and reconstructs an estimate of the image. In supervised training, they are fed into an additional 4-layer neural network, which classifies it as crater or non-crater.

Region	[146]	[182]	[150]	[151]	Ours	Ours (no pre-training)
West (1_24, 1_25)	67.89	85.33	83.98	88.78	<b>90.75</b>	87.92
Centre (2_24, 2_25)	69.62	79.35	83.02	<b>88.81</b>	88.78	88.22
East (3_24, 3_25)	79.77	86.09	89.51	90.29	<b>91.62</b>	89.97
Mean	72.43	83.59	85.50	89.29	<b>90.37</b>	88.70

Table 6.1:  $F_1$  scores for binary classification of candidate craters. The results for others’ models are reproduced from Table 1 in Cohen et al. [151]. The best performing model is in bold for each row. Ours performs best overall, with unsupervised pre-training providing a boost of around 1.6% overall.

The model was trained using SGD, with a learning rate of 0.005 and a batch size of 32. Training continued until performance had converged on the training set, at which point the model’s weights were frozen, and it was used on the craters and non-craters from the two images left out of training. The  $F_1$  scores of the model, both using a pre-trained CNN via the denoising autoencoder, and with a randomly initialised one, is given in Table 6.1. This experiment suggests the proposed method for classification, using an unsupervised autoencoder to pretrain the CNN, works well, and outperforms previous methods.

## 6.4 Detecting Craters

### 6.4.1 Design

The CNN classifies a single image patch, but cannot survey an extended scene and locate multiple craters within it. To do this, several extra modules were designed to work with the classification model. Figure 6.4 gives an overview of the CDA’s pipeline. First, images are split into overlapping patches, where the distance between each patch is the *stride*, which is smaller than the patches’ dimensions, meaning the same point in the full image appears in multiple patches. This is also done at multiple scales, in order to generate image patches suitable for detecting a range of crater sizes, but they are always resized to the 54-by-54 pixels expected by the CNN.

These patches are inputted into the CNN trained in Section 6.3. The CNN outputs a 200-dimension feature vector for each of these image patches. The four final layers of the classification model are then used to classify each of those vectors. In parallel to this, a new neural network is used to estimate the location and size of the crater in the image, the details of which will be returned to shortly. If the classifier reports with over 50%

confidence that there is a crater, then the detection is added to the list of all detections, with its location, size and confidence value (between 0.5 and 1 – the value of the neural network’s softmax output).

Next, these detections are clustered using the mean-shift algorithm. Ideally, each crater is detected in multiple image patches, and the localisation layer accurately places all these detections at the centres of the craters, creating tightly clustered groups on each crater centre that the mean-shift algorithm can easily separate. After this process, however, some double counts can still persist, if the mean-shift algorithm clustered a single crater’s detections into two or more clusters. These are removed by calculating the IoU (the ratio of overlapping to non-overlapping areas) between nearby crater candidate pairs. If the IoU is above a threshold level, the crater candidate with the lower confidence is rejected. After the elimination of overlapping pairs, the final set of crater candidates has been produced.

The mean-shift algorithm was chosen because it has only one parameter to tune (the bandwidth, which dictates the spatial dimensions that the clusters are anticipated to have). It is also ideal because—unlike k-means—it does not assume a specific number of clusters.

As mentioned previously, the CDA’s sliding window scans the image at different resolutions, to allow for craters of different sizes to be detected. Therefore, an important design question was whether to perform mean-shift clustering separately on each size scan, or on all detections together. The former was chosen for several reasons. First, the bandwidth parameter in the mean-shift algorithm assumes that all dimensions are equally important, however, distances in the spatial dimensions are not equivalent to “distances” (differences) in diameter. Therefore, to use mean-shift over all three dimensions, bandwidth would need to be expressed as two tunable parameters, one for spatial dimensions, and another for diameter. Second, if multiple size scans found the same crater, then the double count elimination performed after clustering would find this duplicate and remove it. Third, treating each size scan separately meant it was possible to set the bandwidth as a fixed ratio of the size of a crater. Through trial-and-error, and visual inspection of the results, a bandwidth of  $\frac{1}{6}$  the size of the image patches was arrived at.

As well as the crater locations, a confidence value can also be assigned to each cluster. This is calculated as the sum of confidence values of the members of the cluster. For example, if three sliding window locations were classified as positive, with confidence values 0.6, 0.7, and 0.9, then the cluster they form would have a confidence of 2.2. The final cluster confidence values are therefore heavily dependent on the stride of the model.



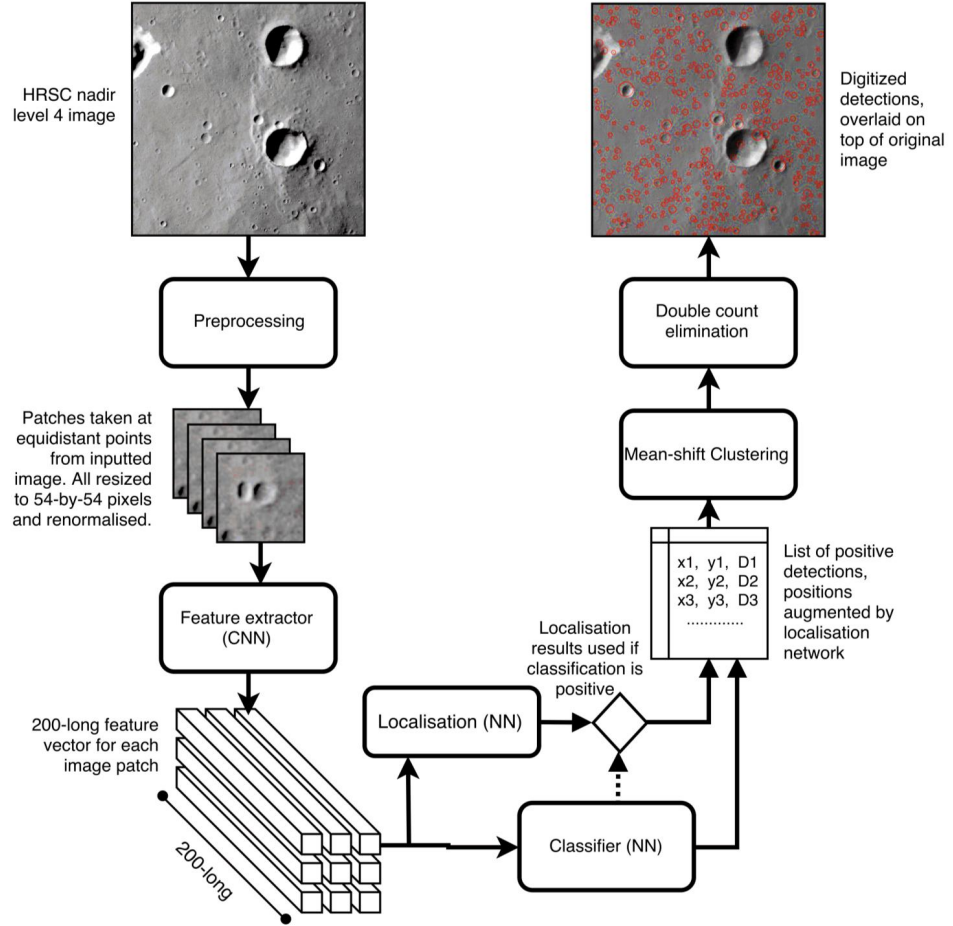


Figure 6.4: Full CDA pipeline. First, images are cropped and preprocessed. Then, each patch is fed into the CNN, which outputs the feature representation of that patch. This is then used to compute the patch’s class (crater or non-crater) and if a crater is thought to be found, the localisation results are used to refine the position and size estimate. Finally, these detections are clustered using mean-shift and double-counts, where they exist, are eliminated.

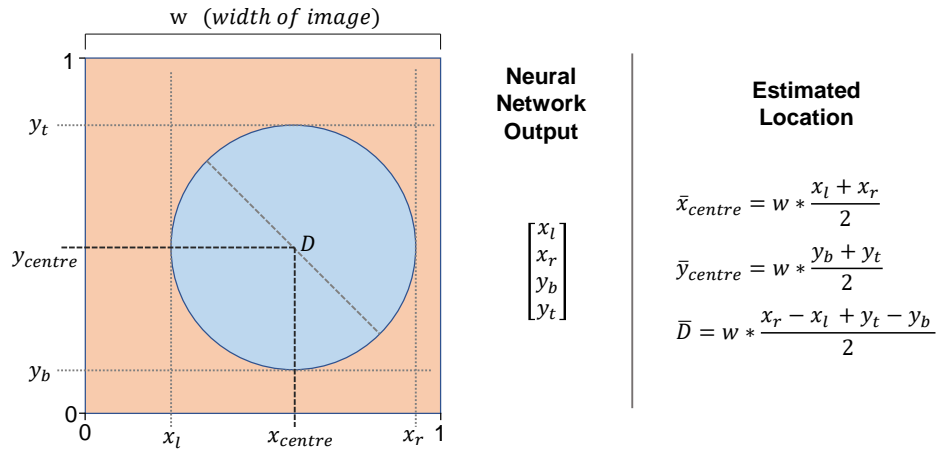


Figure 6.5: Schematic describing how the localisation network’s outputs relate to the estimated location of the crater. The outputs of the neural network are always between 0–1, and are then scaled to image coordinates by multiplying by the size of the image. In this example, the offset of the sliding window’s position is omitted for simplicity, but there would also be an addition to the final estimated location based on the position of the image patch being considered within the whole image.

When the CDA is applied, it is best to set a confidence threshold, below which detections are discarded, which can be set through visual inspection of the results.

Training the neural network responsible for localisation was straightforward. Image patches were extracted from the dataset, containing positive crater examples in random positions and a range of sizes, such that they could appear anywhere in the image patch, with a diameter between 25% and 50% of the frame. These were put through the CNN, to create the same 200-dimension feature vectors used by the classifier. The localisation network took these feature vectors and outputted 4 values: The left, right, top, and bottom bounds of the crater in the image, on a scale between 0 (left-bottom corner) and 1 (top-right corner). The values could then be transformed into image coordinates and a diameter for the crater (see Figure 6.5). Negative examples were not used in training the localiser, as the localisation would only be used when the classifier returned a positive prediction.

### 6.4.2 Performance Issues

With this full detection pipeline, the hope was then to immediately validate the model on the same dataset but as a detection task, rather than a classification one. Unfortunately, several problems arose which made it impossible to continue without significant changes. This section will introduce the issues that arose, and tie them to specific aspects of the CDA's design. Experimental results for both the original, and improved models, are given later in Section 6.6.

**Computational efficiency:** The time to run the algorithm was of major concern, because it made the future application of the algorithm to a large area of Mars very difficult. The computation time was dominated by two processes. First, the sliding window operation (run on a Central Processing Unit—CPU), extracting all the overlapping image patch arrays from the scene. Second, the CNN, which had to process all of those image patches, at different locations and resolutions (run on a GPU). For example, for one of the 1700-by-1700 pixel images in the dataset, with a stride of 6, about 80,000 individual patches needed to be processed. This was then repeated for multiple resolutions, meaning there would be  $> 10^6$  image patches. Regardless of the hardware used, this would take a long time to compute. For a 1700-by-1700 pixel tile (equivalent to 450 km<sup>2</sup> at 12.5 m/pixel) and a stride of 6, the model takes roughly 7 minutes on an *Nvidia RTX2080 Super* GPU. This is orders of magnitude slower than other object detection frameworks would take (see Chapter 8).

Of course, the stride can be increased to speed the model up. The computation time goes quadratically with the inverse of the stride, because the number of processed image patches is proportional to its square. However, a larger stride means fewer chances to predict image patch class, increasing the variance in the results and thus lowering performance.

**Classification Errors:** The classifier was trained to discriminate between image patches which contained craters, and those which contained non-crater features. In the context of classification, this gives high accuracy. However, when used as a sliding window, this led to several prevalent error populations.

First, the classification network was unfamiliar with featureless terrain, because only image patches with craters (or features that were not craters) were given to it in training. These non-crater features were parts of the image that the candidate selection algorithm developed by Urbach et al. [146] returned, meaning they generally had some visual similarities with craters. This led to unreliable classification over flat areas, and over terrain that the classification algorithm had not been trained on.

Second, the size of positive examples which the classification algorithm had been trained on were varied between 25% and 50% of the image patch size, and the location also shifted around the image patch. This meant that the algorithm could successfully identify positive examples in multiple points in the sliding window. However, the geometrical limits that separated positive with negative examples were never well defined for the model. The negative examples used in training were taken from other locations, not necessarily nearby any crater examples. This meant that the model would classify image patches as positive, even if only a portion of the crater was visible, or if it was somewhat smaller or bigger than the size range it had been trained on, because no negative examples had been supplied with that geometry.

Third, was a particularly consistent and problematic error arising from the morphology of craters. The data used, from HRSC, has a fixed illumination angle, meaning the shadows and highlights of craters always appear in the same orientation, with the shadows on northwest, and the highlights on southeast. As well as a central depression, most impact craters also have a rim which is raised above the surrounding terrain. For many craters, this created a field of false positive detections to the northwest of the crater, probably because it replicated the highlight-shadow pair on the inside of the crater (see Figure 6.6). This systematic error meant that immediately next to many true craters with a prominent rim, was a false crater detection of about half the diameter of the true one.

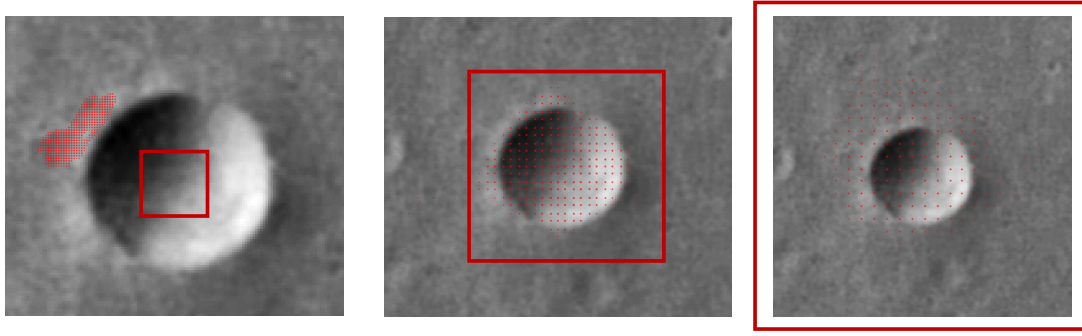


Figure 6.6: Systematic behaviour of CDA at different fields of view. The red boxes indicate the approximate size of the image patch used in the sliding window. Red dots are the central point in the sliding window position for all positive classifications of the image patch. At the smallest scale (left), systematic errors are found on the upper-left rim of the crater. The next plot (centre) show the crater being successfully detected in the correct regions. Finally, the right-most plot depicts a field of view that should be too big for detection of the crater. However, detections are still made, which may affect the accuracy of the final clustered results.

**Localisation Errors:** The neural network responsible for localisation had been trained on the positive examples from the dataset, at different positions and sizes in the image patch. However, recall that the distribution of labelled craters dropped off significantly below a diameter of 250 m (Figure 6.2). This means that many craters at lower sizes are unmarked, and may appear in the image patch alongside the labelled one. Even if both craters are labelled, only one of them will be considered by the loss function in that sample. If both craters are within the size range that the localisation network has been trained to recognise, then it has no way of knowing which of the craters it should be localising. In this instance, the shape of the loss function (the root mean square error, see Section 2.2.2 for more details) dictates that the minimum expected loss is the average between the two craters, rather than strongly selecting one or the other. Therefore, the localisation network, when given multiple craters in its field of view, tends to average their positions, leading to a single false detection somewhere between them (Figure 6.7). Given that craters are very often clustered, thus lying close to another of a similar size, this effect is incredibly common, and severely impacts the detections made by the CDA.

## 6.5 Improvements

### 6.5.1 Hard Negatives

Originally, the classification model was given positive and negative examples taken from the labelled craters and non-craters respectively. This created many issues with the classification algorithm, as discussed in the previous section. To resolve this, *hard negatives*

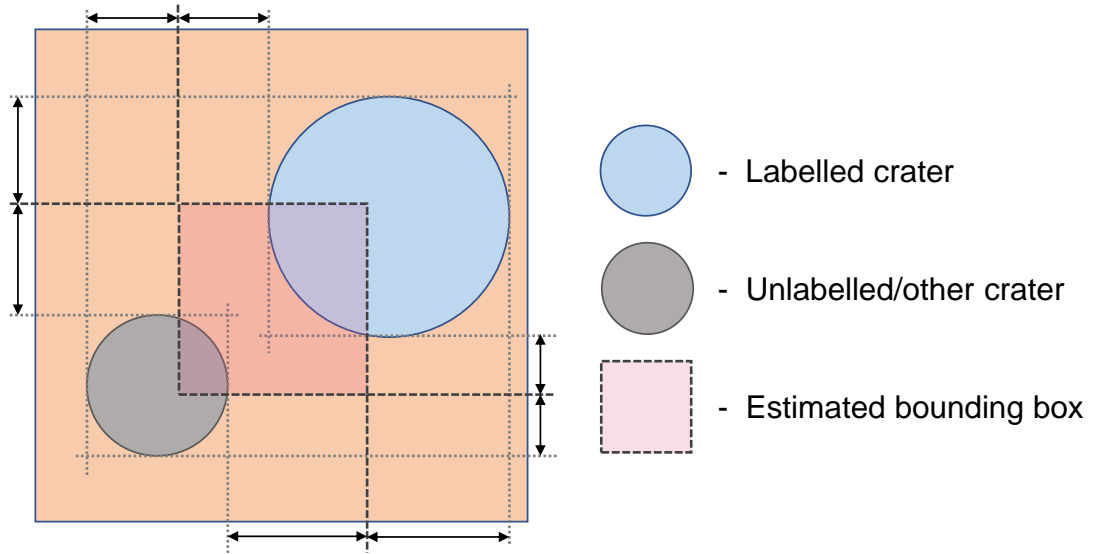


Figure 6.7: Localisation of craters relies on estimation of the bounding box around the crater in an image patch. However, this assumes that there is only one crater in each patch, whereas in reality there are often multiple craters. As shown here, this situation can lead to a bounding box estimate that is an average of the positions, producing an erroneous detection.

are taken very close to positive examples, so that the model has a strong set of examples close to the class boundary (see Figure 6.8). In this refined training set-up, a positive example is defined as positive only when the central pixel of the image patch is within the crater, and when the crater is between 25% and 50% of the image patch width. Negatives are also sampled from just outside the crater rim (making sure the central pixel does not fall into another crater in the vicinity), or at a size just outside the 25–50% range.

An experiment was conducted to measure the effect of these hard negatives on the classification model’s ability to reject examples close to, but not fulfilling, the geometrical requirements for a positive detection. To do this, the model was trained twice on data from the western and central pairs of image tiles (1\_24, 1\_25, 2\_24, and 2\_25), once without hard negatives, and once with them included. They were then tested on examples drawn from 3\_24 and 3\_25, at a range of positions and scales, such that the performance of the model could be measured against the distance of the crater’s centre from the centre of the image patch, and against the ratio of the diameter to image patch size.

Both the precision and recall of the models for negative and positive classes are plotted in Figure 6.9. The precision and recall of the negative (non-crater) class is somewhat less intuitive than the positive case, as it imagines a situation in which we focus on the statistics of non-craters rather than craters. Nevertheless, in this experiment, the negative statistics are useful in understanding the gains in performance across these hard negatives. Overall, introducing hard negatives to training increased the ability of the

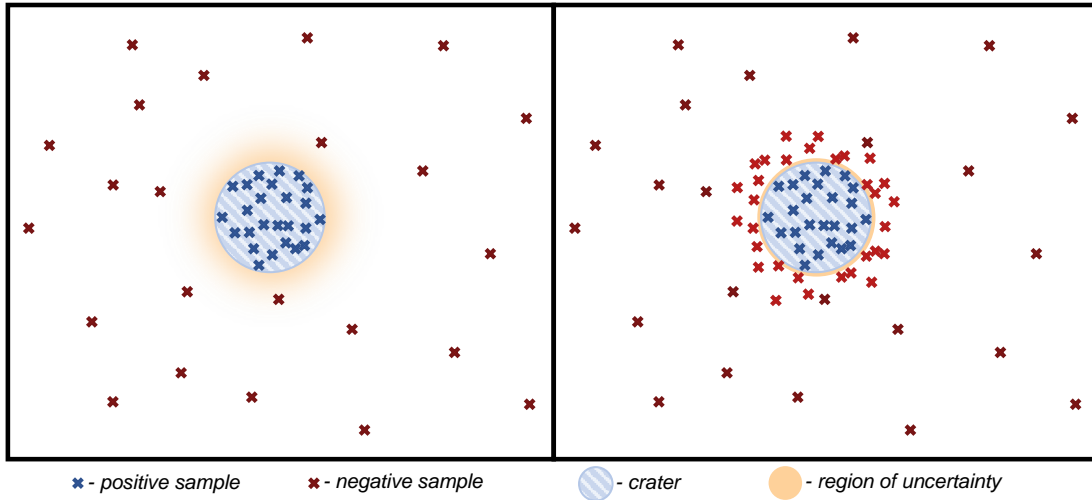


Figure 6.8: Representation of the positive and negative spaces around a crater during classification. The crosses denote the centre point of a sample taken during training. The negative space is significantly larger than the positive one, and thus has a much lower sampling density. This creates a large region of uncertainty around the densely sampled positive space (left-hand plot). With the introduction of hard negatives (right-hand plot), the density of both positive and negative spaces are high close to the boundary, making the model’s region of uncertainty much tighter to the actual boundary of positive and negative spaces. In reality, these spaces are 3D volumes, because of the possible variation in size of the crater. However, for simplicity, only the two spatial dimensions are shown here.

model to discriminate between positive and negative examples. In particular, the recall of hard negatives was boosted substantially.

### 6.5.2 Localisation

The stricter definition of positive craters that was implemented to improve classification accuracy also aided localisation. Originally, the localisation neural network suffered because there may have been multiple craters in the field of view, which made it impossible for the network to know which it was supposed to localise. The added constraint on positive examples that the crater must cover the central pixel reduced the frequency of this problem, because it is much rarer for two craters to cover the central pixel of the image patch simultaneously (although it still happens when craters overlap with one another, however this is relatively uncommon). This means the localisation neural network learns to only localise the crater in the image patch which covers the central pixel.

We can measure the positive impact this change has on localisation by seeing how the average IoU (see Section 2.2.7) changes with respect to the size of the crater in the image patch, and its distance from the centre (see Figure 6.10). An increase of 0.05 to the mean IoU is a significant improvement, especially considering that this improvement

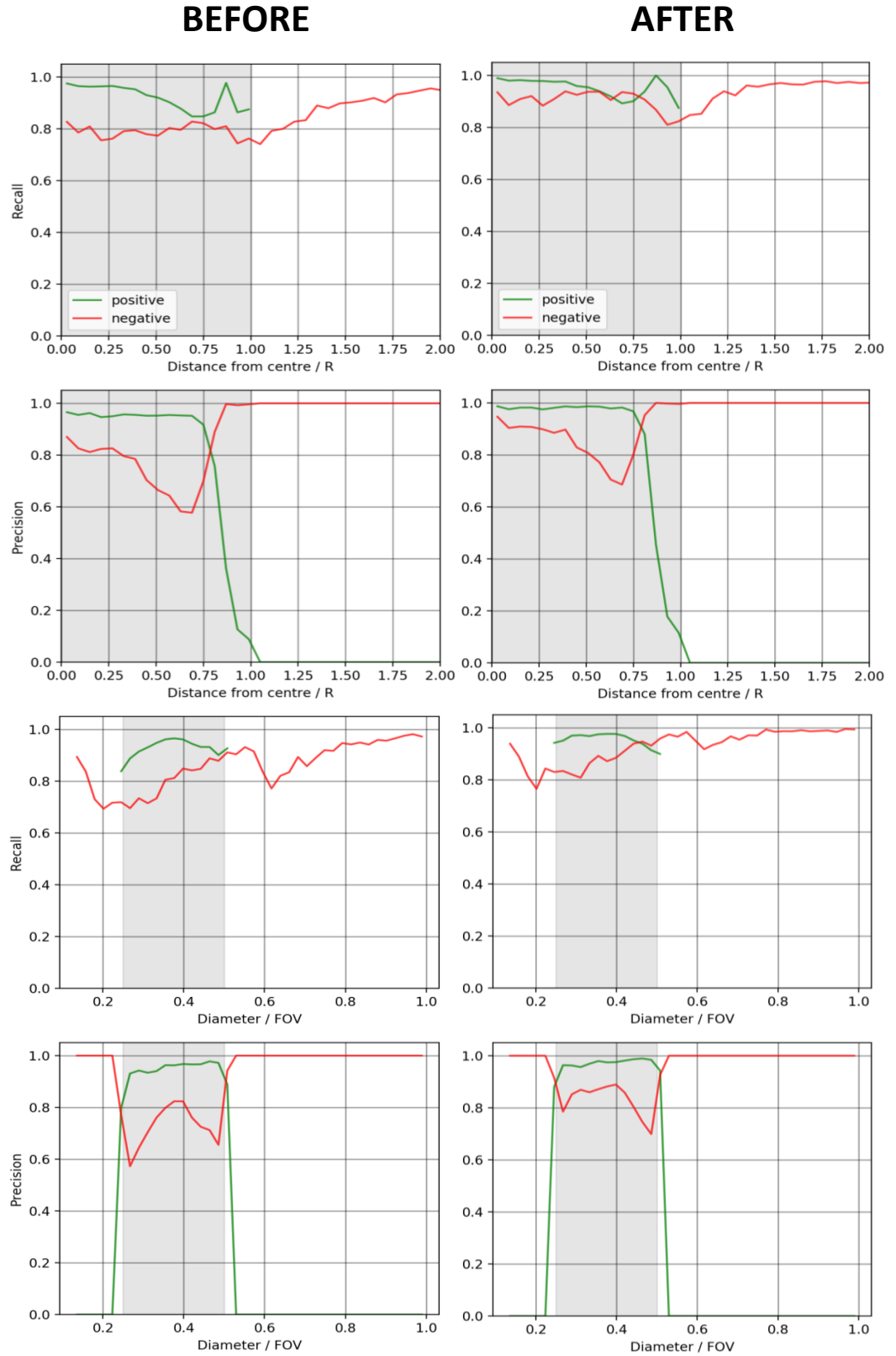


Figure 6.9: Recall (rows 1 and 3) and Precision (rows 2 and 4) for both craters (green) and non-craters (red) before and after the addition of hard negatives in training. These are measured against geometrical parameters: the ratio of the distance of the crater's centre to its radius (top two rows), and the size of the crater divided by the field of view of the image patch (bottom two rows). The grey region denotes the geometrical range for which positives can exist, because they cover the centre pixel of the image patch, and are between 25–50% of the image patch size.

is mostly from examples where the crater is far from the centre, or close to the lower or upper size limits.

### 6.5.3 Computational Efficiency

The driver of this project was to produce a CDA which could be employed to detect craters over a significantly large area of the surface of Mars. Isidis Planitia (the focus of Chapter 9) was not specifically targeted as a study site whilst the work in this chapter was conducted, however a region of roughly this size was thought to be desirable. For an area the size of Isidis Planitia (roughly 1300 km across), at the 5 m/pixel resolution of the CTX mosaic used, the algorithm would need to be run across the equivalent of 70,000 of the 1700-by-1700 pixel images in the HRSC dataset. Given the original speed of the algorithm was around 7 minutes for one of these images, this would lead to almost a year of continuous computation to cover the area of Isidis Planitia. Therefore, optimising the speed of the CDA was paramount.

Two major changes were made to the algorithm’s code to increase computational efficiency, neither of which affected the results outputted by the model in any way. The first of these was the use of *multithreading*. Originally, the operation responsible for creating the cropped image patches from the full image was called sequentially, making one patch after another. By using multithreading, this could be parallelised across many (e.g. 20) threads, which sped this part of the code up substantially.

The second modification to the code was to use *queueing*. Before this change, all cropped image patches were created before then inputting them all into the CNN. By queueing, the images were processed in batches and fed to the CNN as soon as it was ready to accept more, whilst the cropping continued apace. In this way, the two processes could be run concurrently, which was ideal given that one was CPU-bound (cropping) and the other was GPU-bound (the CNN).

## 6.6 Experimental Results

To test the impact of the improvements discussed in Section 6.5, some comparative experiments were carried out. The first was to measure the ability of the model to successfully detect craters, and the second was to measure its speed.

Like in Section 6.3, the data was divided using 3-fold validation, where 4 of the 6 images were used for training, and the other two then used for testing. These were then swapped, and the model retrained, so that each image was used in validation once.



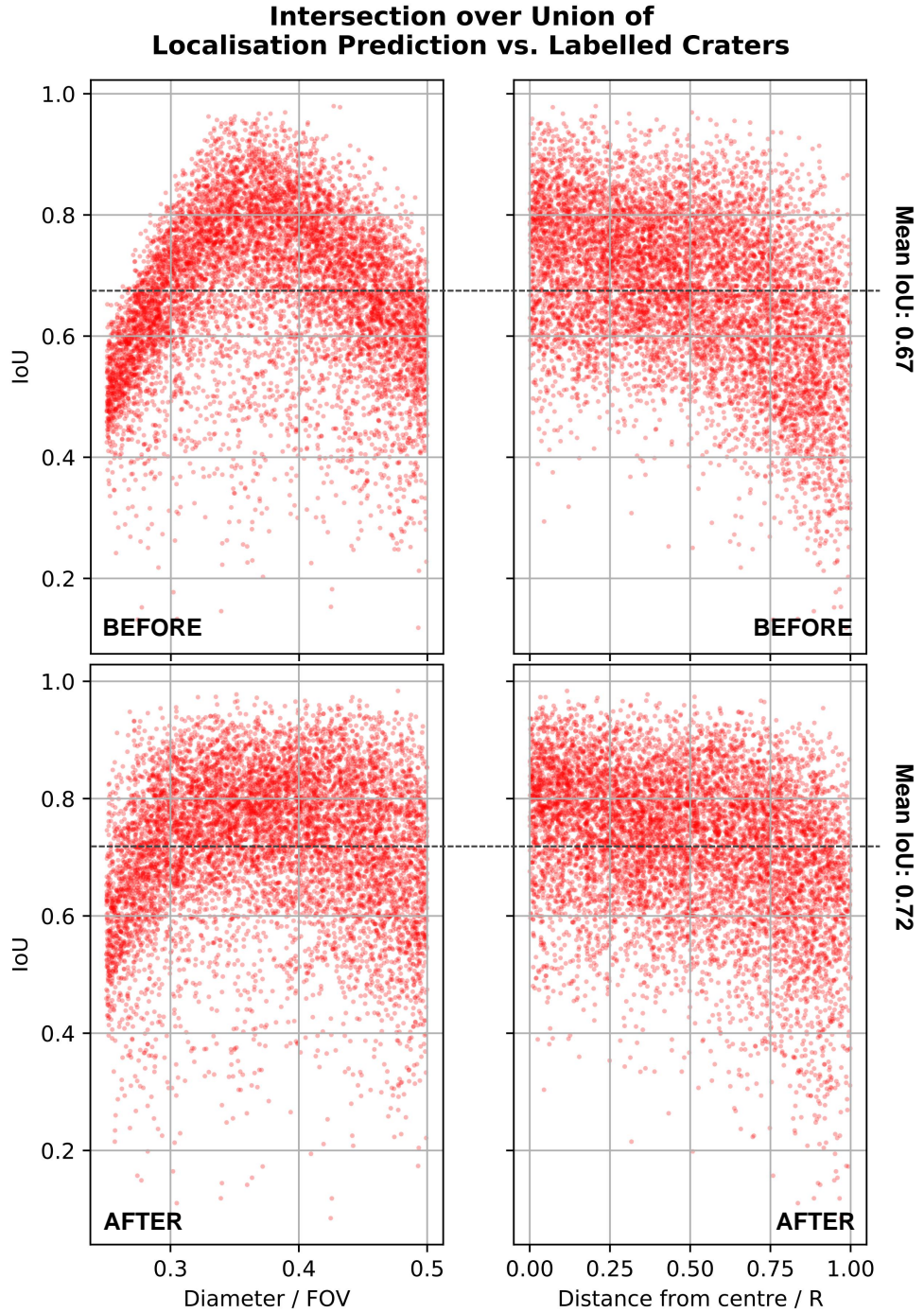


Figure 6.10: Scatter plot showing the IoU of the bounding box prediction against the dataset. The IoU for 5000 examples is plotted against the size of the crater relative to the image patch (left-hand plots) and the distance from the image patch centre as a ratio of the crater's radius (right-hand plots). The improvements made, including hard negatives and disallowing positive detections outside the crater's rim, have boosted the IoU from an average of 0.67 (top row) to 0.72 (bottom row). In particular, the IoU improved greatly for craters far from the centre, and at the extremes of the size range.

Hard negatives	Stride	mAP@50%	Precision	Recall	F <sub>1</sub>
N	6	44.45	54.00	53.15	53.57
N	12	37.03	45.71	44.09	44.89
N	18	37.30	47.69	48.81	48.24
Y	6	59.51	65.06	63.78	64.41
Y	12	51.56	54.18	53.54	53.86
Y	18	52.96	59.23	60.63	59.92
Bandeira et al. [148]		-	88.0	80.7	84.19

Table 6.2: Performance metrics for the CDA in different configurations, and a method proposed by Bandeira et al. [148] (although this method uses a much less strict matching criterion between prediction and labels, and so direct comparison with it is somewhat unfair). Including hard negatives and using a small stride benefited performance. Confidence thresholds were chosen for each configuration so that precision and recall were roughly balanced.

Unlike in the experiment to measure classification performance (Table 6.1), the model was shown the entire 1700-by-1700 images, and sought to find the craters within them.

Measuring detection performance is less straightforward than classification, because the predictions must be matched to the labelled craters that they are estimating the position of. To do this, the standard object detection metric of mAP@50% was used (see Section 2.2.7 for further details). This metric gives an overall picture of model performance, encapsulating both its ability to find craters, and to localise them. It is also independent of the specific confidence threshold used for the model, because it is a mean value over many different threshold values. This is ideal for comparing the models at different strides, because changing the stride alters the distribution of confidence values (by changing the number of points included in each cluster). As well as the mAP@50%, the precision, recall and  $F_1$  for detections with  $IoU > 50\%$  were also calculated, in order to compare results with others’.

The experiment lent strong evidence that including hard negatives in training benefits the model. Table 6.2 gives the performance of models trained with and without hard negatives, and at different strides. There is a massive performance increase when using hard negatives, and as expected, the lowest stride of 6 performed best. Surprisingly, the performance at strides of 12 and 18 seemed to plateau. The results of the models are also shown in Figure 6.11 for one of the images (3\_25). Visually, the models are harder to tell apart, however one interesting thing to note is that there are many false positive detections with small diameters. On inspection, many of these are in fact craters, but due to the fall off in recall in the labelled dataset (Figure 6.2) they were not included in the labelled data. Therefore, it is likely that the models performed somewhat better than is suggested in Table 6.2.

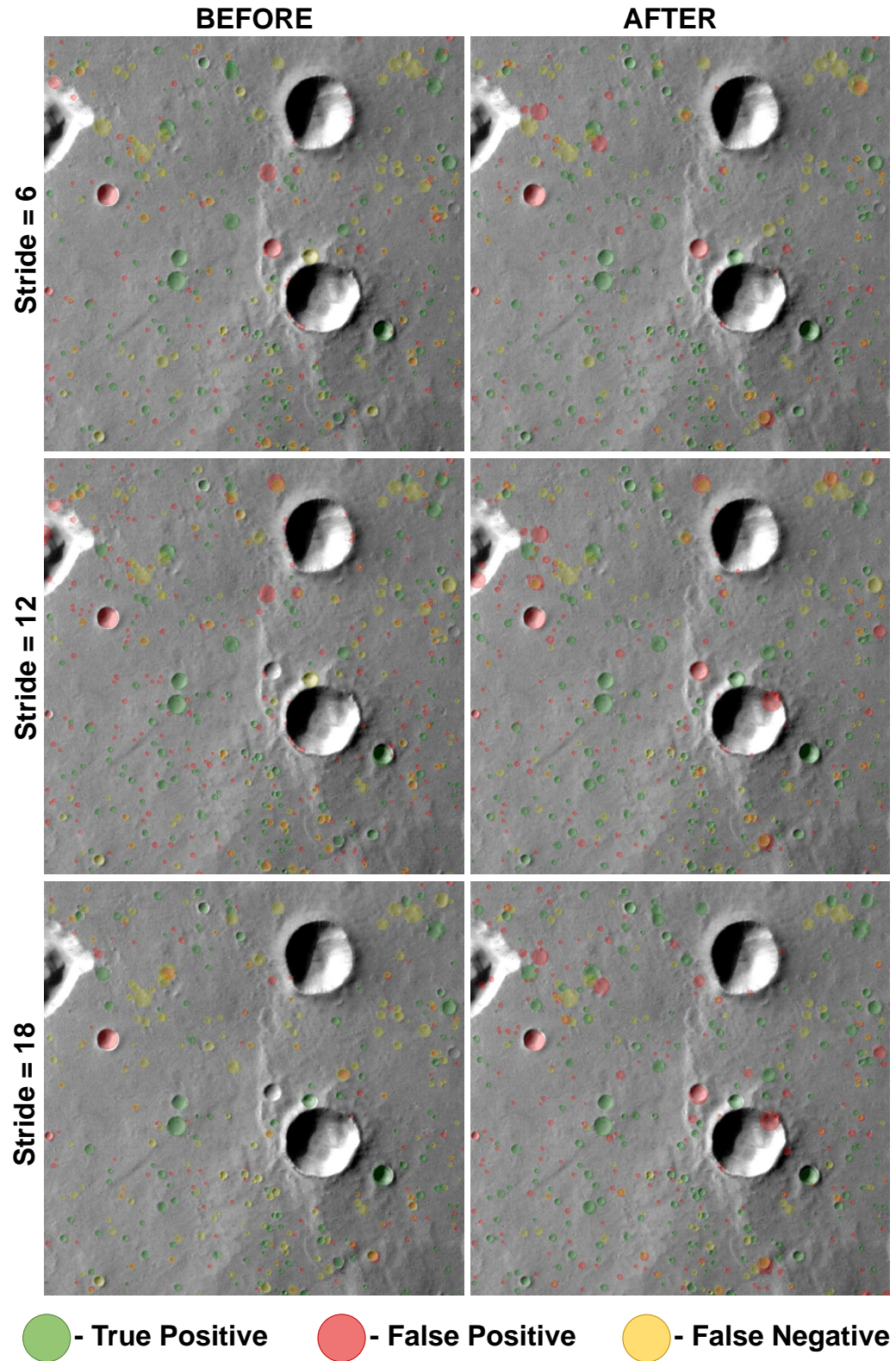


Figure 6.11: Image 3\_25 from the dataset, with detection results from the 6 versions of the CDA overlaid. Some craters were consistently missed by all models, whilst in other instances the models behaved differently. Where red and yellow circles are co-located, this indicates that the model found the crater, but that it did not localise it well enough for the IoU of the detection and label to be greater than 50%. In some cases, real craters are marked as false positives. These are missed in the ground truth dataset, and lead to a reduction in a model's apparent performance.

Stride	Multithreading	Queueing	Cropping(s)	CNN(s)	Clustering(s)	Total(s)
6	N	N	340.5	67.4	8.7	416.6
6	Y	N	17.8	67.2	8.7	93.7
6	Y	Y	71.1		8.7	79.8
12	N	N	85.1	33.3	2.1	120.5
12	Y	N	4.5	33.3	2.0	39.8
12	Y	Y	34.8		2.1	36.9
18	N	N	45.4	28.0	0.9	74.3
18	Y	N	3.5	27.9	0.9	32.3
18	Y	Y	30.1		0.9	31.0

Table 6.3: Times taken for CDA to process a 1700-by-1700 image. When queueing was used, the cropping operation and the CNN were run in parallel, and thus their time is combined. Clustering was strongly affected by stride, because there were fewer points, making mean-shift clustering easier.

Finally, the impact of multithreading, queueing, and the model’s stride on computation time was ascertained. The CDA was used to detect craters on a single 1700-by-1700 image, and the time taken for each of the stages was measured. As expected, increasing the stride, and using multithreading and queueing made computation much faster (Table 6.3). At its fastest, the CDA was able to output results in a total of 31 seconds. However, as seen previously in Table 6.2, using such a large stride would result in a severe reduction in performance. When using multithreading and queueing, the model with a stride of 6 took 80 seconds. This is a significant speed up from the 7 minutes it took previously, however it would still take approximately 10 weeks to process an area equivalent to Isidis Planitia.

## 6.7 Interim Conclusions

Detecting small craters in optical imagery will open up avenues for research into their formation and distribution, if it can be done accurately. This chapter documents an attempt to create such an automatic method from scratch. The proposed method has many merits, with state-of-the-art performance as a classification algorithm. Ultimately, however, the CDA was shown to be unsuitable for real-world application.

This study is a good example of the trap discussed in Section 1.3.2. In this case, performance of the algorithm has been pursued in a toy problem (classification). When transferred to the real-world application (detection), problems arose which deleteriously affected both the CDA’s performance and utility. Many of these emergent issues were ameliorated or resolved through improvements to the algorithm (Section 6.5). However, whilst performance was shown to increase when hard negative samples were included

(Table 6.2), it is clear that compared to previous methods based on candidate generation through template-matching (e.g. [148]), the CDA does not offer significant benefits.

As well as limitations in performance, the CDA was still slow when computing predictions, despite the order of magnitude increase in speed due to multithreading and queueing (Table 6.3). State-of-the-art object detection methods such as YOLO or Faster-RCNN (further details in Section 2.2.5) can run at multiple frames per second, allowing for real-time object detection in video. Even for high resolution images, these models can output detections several orders of magnitude faster than the CDA proposed here. The utility of the approach is therefore severely limited, as deploying it across large regions of Mars would take a long time.

Whilst this study failed to produce a CDA with practical value, many valuable lessons were learnt during the exercise. First, the existing labelled data for small crater detection is sparse, with this dataset of 6 images being the largest available. This observation led directly to the creation of a new, larger and more varied small crater catalogue, documented in the following chapter. Second, designing a new and original algorithm, whilst a valuable learning exercise, will not necessarily produce the best results. Instead, deploying and tweaking existing object detection methods—which have a proven track record in other similar tasks—may be more fruitful, as is demonstrated in Chapter 8, especially when used in a sensor independent fashion, to increase the quantity of available data.



## 7 | ORBYTS Crater Dataset

*This chapter is based heavily on a paper published in 2020 [2]. I devised and organised the project, which involved 16 school students at The College of Richard Collyer, Horsham. The students performed annotations of craters for the dataset, which was then collated, analysed and validated by me. This chapter is a rearrangement of the paper’s technical sections, with some minor rewording.*

### 7.1 Motivation

Manual surveys of Martian craters have successfully mapped all large ( $D > 1\text{km}$ ) craters, with a high degree of accuracy [11]. However, given that hundreds of millions of craters exist on the Martian surface, it will be impossible to conduct a manual survey globally, or indeed of any substantial portion of the surface. For this reason, automated detection is likely to lead to the most fruitful results. To this end, machine learning models—in particular Convolutional Neural Networks (CNNs)—have been the state-of-the-art method for many computer vision tasks since 2012 [5]. Specifically, CNNs have been successfully employed for a huge variety of Object Detection applications (e.g. [71, 75]). The power of these methods is revealed most spectacularly when the datasets used to train them are very large, because of their ability to generalise and interpolate in a hugely complex input space. These new techniques present a huge opportunity for the automation of regional or global crater surveys.

For Martian cratering, few datasets exist, and those that do (e.g. [11]) often do not include small craters. To date, the biggest openly available dataset of small Martian craters was released in 2012 [148]. This contains 3,050 craters from the HRSC [30] over Nanedi Valles between 200 m and 5 km diameter. Our dataset extends and complements this prior effort, as it uses multiple annotators, a different camera system, is located in a different region of Mars, and expands the diameter range down to tens of metres. Multi-annotator datasets have some precedent in the crater detection field: for example, the Moon Zoo project employed several thousand citizen (volunteer) scientists to annotate Lunar craters over the Apollo 17 landing site [12]. The novel metric we use to evaluate the consistency of our results (Section 7.2.2) provides a framework for future crowdsourcing efforts in a variety of object detection problems, not restricted to crater counting.



By pursuing a labelled dataset with CTX imagery, we expand the number of satellites for which datasets exist, making testing of sensor independence easier. Additionally, CTX is the highest resolution camera for which a near-global mosaic is available, and therefore, as shown in Chapter 9, it is the most appropriate satellite to use for small crater detection at scale over the surface of Mars.

## 7.2 Methods

### 7.2.1 Collection

Our annotations were conducted by a group of 16 young ORBYTS (Original Research By Young Twinkle Students) [186] participants, at the sixth form College of Richard Collyer, Horsham, UK. In order to prepare, the annotators were given approximately 15 hours of seminars on Martian science, machine learning, and statistics. In addition to this, we worked collectively to closely define how annotations would be made, what was counted as a crater and what was not, and any possible edge-cases. This was all done in order to ensure a high accuracy and as low a variance as possible between the annotators. The tool used in annotation was a customised open-source image annotation tool *LabelImg*<sup>1</sup>. Our version was modified to restrict annotations to be equal width and height, as craters are usually and most easily defined as a circle, rather than an ellipse. We also added a toggle for ‘difficult’ annotations, which could be triggered when annotators were unsure of their own marking.

After a training session using the software, annotations were conducted intermittently over a period of around eight weeks. Six annotators were selected using a random number generator for each tile and were then separated so as to not be influenced by one another whilst labelling. Once a tile was completed, an annotator would then be assigned another tile randomly from those still to be done. Due to absences, differences in speed, and variation in the difficulty of the images, there is some variation in the quantity of annotations made by each annotator.

Craters of all sizes that were visible were annotated. The lower diameter limit was determined by the resolution and quality of the CTX images. At the 6 m/pixel of CTX, we found that in practice few craters under around 30m (5 pixels) in diameter were visible. However, given that some craters smaller than this could be resolved, we did not enforce a minimum diameter limit on individual annotations (the clustering process does filter results smaller than 3 pixels in diameter). If craters extended partially beyond the

---

<sup>1</sup><https://github.com/tzutalin/labelImg>



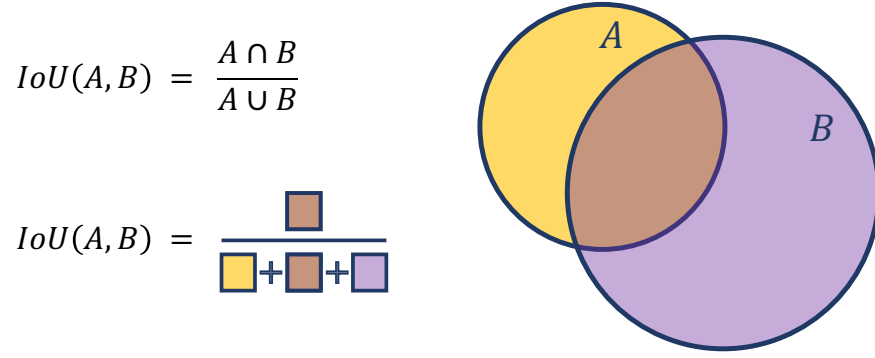


Figure 7.1: Graphical depiction of the Intersection over Union, or Jaccard index, for two annotations. The areas are calculated using circles with diameter equal to the width and height of the bounding box of the annotation.

sides of the image they were ignored, as it could lead to ambiguity in size and position. Annotators began labelling larger craters and gradually zoomed in further, panning across the image systematically in order to comprehensively survey all sizes and locations.

Combining individual annotations is done using agglomerative clustering, a technique used since the 1950s for a variety of clustering problems [187]. This has the benefit of requiring only two parameters: one is the function with which a distance is calculated between two annotations, and the other is the cut-off distance at which no more clustering is performed. First, all distances between different pairs of annotations is calculated; then the closest two are paired, and the new point is calculated as the mean between those which were clustered. This process is repeated until no distances remain below the cut-off. For our application, we would like to ensure that the distance between any two markings by the same annotator is 1, meaning clusters can never contain more than one entry from a given annotator.

For two markings,  $L_i^{(n)}$  and  $L_j^{(m)}$ , by annotators  $n$  and  $m$ , we define the distance as

$$d(L_i^{(n)}, L_j^{(m)}) = \begin{cases} 1 - IoU(L_i^{(n)}, L_j^{(m)}) & \text{if } n \neq m \\ 1 & \text{if } n = m \end{cases} \quad (7.1)$$

Where  $IoU(L_i^{(n)}, L_j^{(m)})$  is defined as the Intersection over Union of the two circles, also known as the Jaccard index (Figure 7.1).

The cut-off distance threshold was decided largely by visual inspection. Too low a value, and repeat entries for the same craters might be included in the final survey; too large and multiple craters might be combined erroneously. We found that a value of 0.9 (clustering any  $IoU > 0.1$ ) was close to optimal, and therefore used this for all images. Table 7.1 gives details regarding the results of the agglomerative clustering, and a visual example of the results can be seen in Figure 7.2.

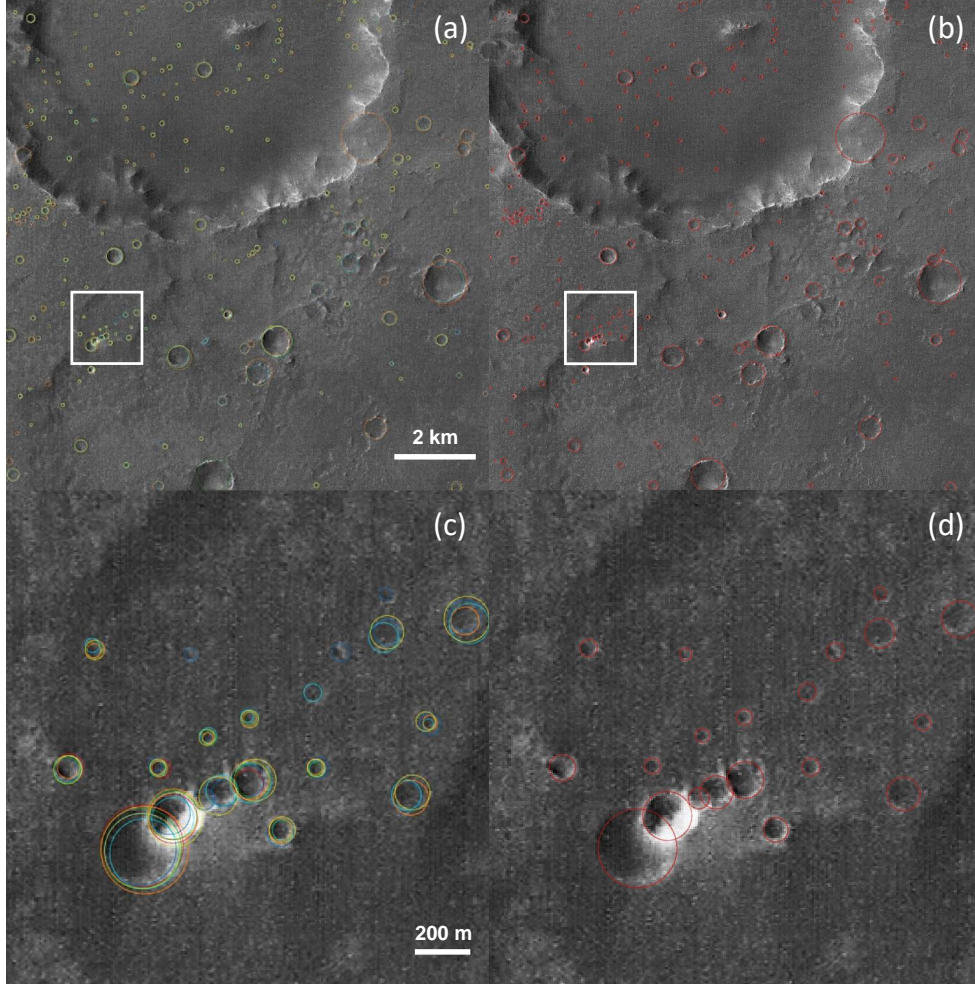


Figure 7.2: Image ‘C’ from the dataset, with annotations displayed. In (a) we see the full image with unclustered labels, each colour representing a different annotator; (b) shows the output labels from agglomerative clustering; (c) and (d) show the same process, zoomed in on a portion of the image. As we see, the initial annotations are in close agreement with one another, and the clustering successfully separates craters, and accurately estimates their centres and diameters. The very large crater in the upper portion of the image was not labelled, as it extends past the bounds of the image.

### 7.2.2 Validation

To validate the accuracy of our dataset, we use statistical measures that quantify the amount of agreement between our annotators, and also between our annotators and a more experienced crater counter (myself). It should be noted that these measures of agreement do not account for systematic errors (bias), but do tell us about variance to some extent. Ultimately, crater counting is an ambiguous pursuit—especially at smaller diameter ranges—because of the limited resolution of our instruments. To calculate each annotator’s *agreement score* on a given image, we perform the following calculations:

1. Let the  $i^{th}$  label from annotator  $n$  be denoted as  $L_i^{(n)}$ .
2. For each  $L_i^{(n)}$  made by annotator  $n$ , compute the intersection-over-union of it with all labels from all other annotators.
3. Let the maximum of all these intersection-over-unions be  $MIoU_i^{(n)}$ . This is the highest intersection-over-union of one annotator’s label when compared to all other annotations from other annotators.
4. Take the mean average of  $MIoU_i^{(n)}$  across  $i$ , to calculate the  $n^{th}$  annotator’s ***Agreement Score***.

This score rests on the assumption that for any given annotator, the combined labels of all other annotators will be a more complete and reliable survey. However, this does become less reliable at extremely low numbers of annotators. The score also cannot factor in craters that were not found by any other annotators, however it does describe the consistency of detection within the group of annotators, and tells us about the positional accuracy of markings. On top of this quantitative validation, all images were scanned visually to look for any obviously spurious annotations, which were then removed from the dataset. These likely came about from accidental inputs and were easy to spot, although they were not common. Table 7.2 outlines some spatial statistics for the annotations, that detail the level of positional agreement between individual annotators. Overall, the positional agreement of clustered annotations is high, with a standard deviation in measured diameter of around 15%, and the central locations varying with a standard deviation of less than a pixel.

To further measure the reliability of the multi-annotator labels, we conducted a simple comparison exercise with an experienced crater counter. The ‘expert’ annotator labelled craters in 4 randomly selected tiles from the dataset (D, F, K and L). Then, the *Agreement*

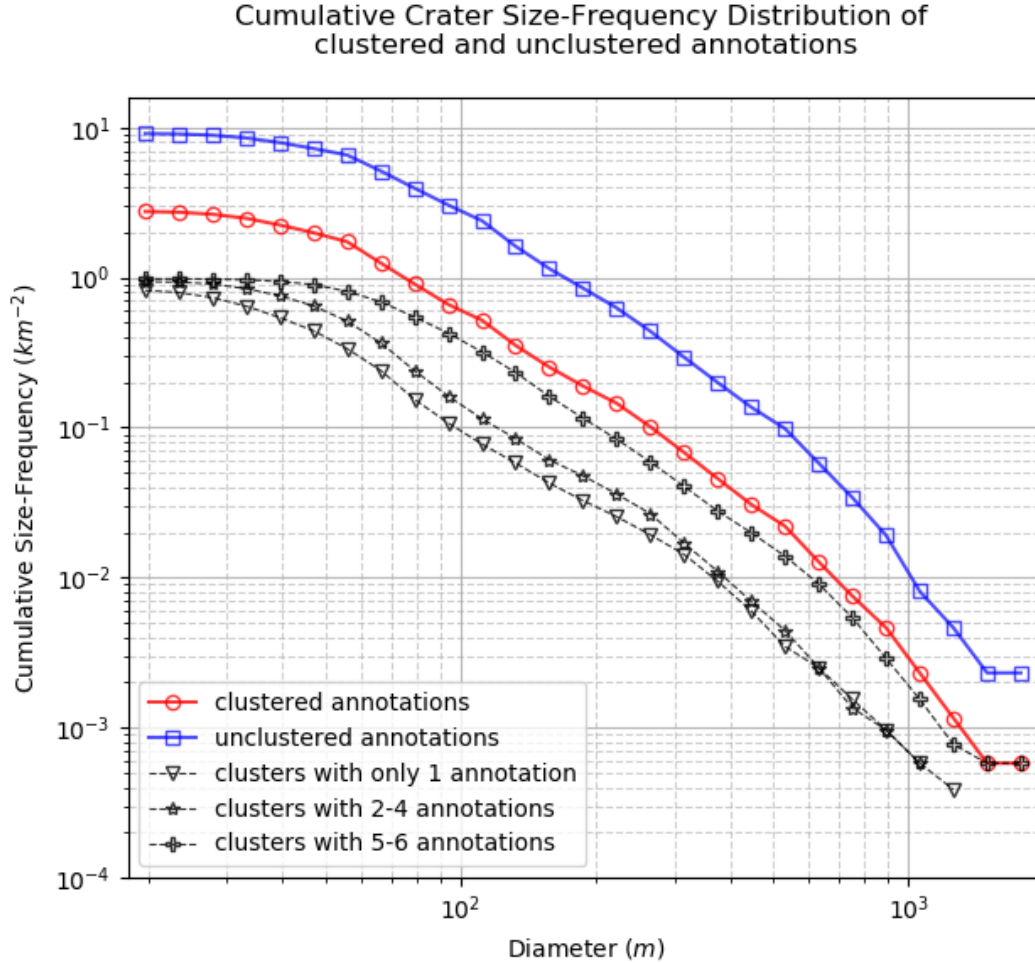


Figure 7.3: Cumulative CSFDs of the whole dataset. As expected, there are more unclustered annotations than clustered. At small diameter ranges ( $<50$  m) we see that the gradient tails off, which is an expected symptom of the limited resolution of our data. At the other end of the size range, the statistics become more unreliable, as the number of craters becomes very low. In relative terms, there are more clusters with single annotations (craters with low confidence) at smaller diameter ranges, whilst most of the large craters were found by 5 or 6 annotators, which is to be expected, as they are much more visible.

SCENE	Valid individual annotations	Clustered annotations	Average annotations per crater	Diameter (m)			
				Median	Mean	Min	Max
A	3230	1,182	2.73	60.0	66.9	18.0	366.0
B	724	196	3.69	108.0	136.0	36.0	918.0
C	1,038	269	3.86	78.0	118.8	30.0	1188.0
D	397	112	3.54	66.0	106.4	24.0	1152.0
E	3,946	1,042	3.79	42.0	57.7	18.0	1938.0
F	1,430	372	3.84	78.0	105.4	18.0	642.0
G	267	72	3.71	105.0	146.4	18.0	642.0
H	223	60	3.72	222.0	263.3	66.0	774.0
I	1,110	325	3.42	72.0	93.2	24.0	552.0
J	297	168	1.77	54.0	83.1	18.0	798.0
K	304	81	3.75	90.0	118.2	24.0	672.0
L	2,780	884	3.14	60.0	64.6	18.0	630.0
TOTAL	15,746	4,763	3.31	60.0	81.1	18.0	1938.0

Table 7.1: Scene-wise analysis of clustering process. On average, each crater was found by more than three annotators, and has a mean diameter of less than 100 m. Annotations were filtered to disallow labels with  $D < 18$  m (3 pixels), hence the small discrepancy in individual counts between Table 7.4 and this. Tiles with fewer craters tend to have high average diameters, suggesting that either small craters are too difficult to see, or that they are preferentially eroded or filled by dust quicker in these areas, relative to those with lots of small craters.

No. of annotations for crater	No. of craters	Mean Diameter (m)	Standard Deviation of diameter (%)	Standard Deviation of centre (m)
1	1442	61.8	-	-
2	636	68.4	16.6	4.37
3	524	72.1	18.1	4.87
4	467	72.6	17.8	4.97
5	572	82.0	16.9	5.51
6	1122	120.5	13.8	5.49

Table 7.2: Positional accuracy of annotations, by number of annotations per crater. Larger diameter craters tend to be annotated more, leading to a higher mean diameter for craters found by more annotators. The standard deviation of these annotations, in comparison to the final clustered value, is around 14-18% overall. The accuracy of the craters' centres is high, with a standard deviation consistently below 1 pixel (6 m).

SCENE	Non-expert annotators						Expert annotator	
	No. of labels			Agreement Score (%)			No. of labels	Agreement Score (%)
	Min	Max	Mean	Min	Max	Mean		
D	32	94	66.5	62.9	77.1	72.9	52	81.6
F	187	305	238.5	65.3	80.6	74.3	240	81.4
K	28	66	50.7	66.8	100	78.4	51	78.2
L	273	696	464.8	46.4	74.1	62.8	589	69.4

Table 7.3: Results for experiment using expert annotations on a subset of the dataset. In general, the expert annotator scores highly in comparison to the other annotators, and has a total count similar to the mean of the non-experts for each tile. This shows that when considered as a whole, the non-experts' annotations have a high agreement with an expert's, suggesting multi-annotator methods can boost the performance of non-experts for object detection labelling tasks.

*score* between these expert labels were calculated against the set of other annotators. As we can see from Table 7.3, the expert achieved scores that exceeded the average inter-annotator score, and in the case of tiles D and F exceeded all the other annotators. Given this—and the fact that the number of labels from the expert in each tile was always within the non-experts’ range—we see that the expert’s annotations are somewhat more reliable than an individual annotator’s, but also that the agreement between the expert against the entire body of non-experts is high. This suggests that using 6 non-experts can provide results which match those of an expert considerably more closely than those of an individual non-expert. This is consistent with Moon Zoo, the citizen science project mapping Lunar craters [12].

The *Agreement score* and positional accuracy measures gives us a way to validate the consistency of individual annotations. However, it does not provide insight into whether the emergent statistics of our crater surveys are reliable. For this, it is helpful to consider the cumulative CSFD of our annotations (Figure 7.3). Cumulative CSFDs are generally defined as the number of craters above a given diameter per  $km^2$ , and are used commonly in age-dating work [188]. Our data holds to the expected inverse power law found in cumulative CSFDs, showing a broadly constant gradient above 50 m diameters. Although no ground-truth exists by which to verify our results, we believe that with the *Agreement score*, positional accuracy metrics, comparison with experts and the cumulative CSFDs all suggest that the annotations comprising our dataset are reliable both individually and in aggregate.

## 7.3 Data Description

The final dataset comprises 12 annotated scenes, each 2,000-by-2,000 pixels in size, from the Context Camera (CTX) [34] aboard the Mars Reconnaissance Orbiter. More specifically, the tiles have been selected in a pseudo-random fashion from the co-registered and orthorectified mosaic over the MC-11 East quadrant, the creation of which is detailed by Michael et al. [189], and Sidiropoulos et al. [190]. These data are co-registered to a baseline Digital Elevation Model which uses HRSC data [191]. The resolution of the data is 6 m, and each image measures 12 km by 12 km. CTX is in a sun-synchronous orbit, meaning all images are taken at 3 pm local time [33], leading to consistent illumination angles across the dataset. The 12 images cover a diverse set of locations, terrain types, and altitudes, as can be seen in Figure 7.4.

Each image comes in both png format and as a geo-coded tif image in *data/images*. Associated with each image is a folder in *data/annotations/raw* containing 6 PASCAL

SCENE	Labels per annotator							Agreement score (%)						
	i	ii	iii	iv	v	vi	TOTAL	i	ii	iii	iv	v	vi	MEAN
A	680	255	396	376	1,111	413	<b>3,231</b>	62.4	74.8	73.5	79.0	40.8	63.5	<b>65.67</b>
B	88	90	93	151	177	125	<b>724</b>	75.5	77.7	82.1	67.9	58.7	71.7	<b>72.27</b>
C	125	212	97	178	230	196	<b>1,038</b>	82.6	69.0	78.6	74.6	68.9	75.4	<b>74.85</b>
D	72	43	32	94	73	85	<b>399</b>	75.7	77.1	82.3	63.9	75.7	62.9	<b>72.93</b>
E	277	503	690	778	869	837	<b>3,954</b>	79.2	73.0	67.3	67.7	62.2	66.6	<b>69.33</b>
F	197	229	235	278	305	187	<b>1,431</b>	75.1	74.7	80.6	69.7	65.3	80.4	<b>74.3</b>
G	45	22	45	45	60	51	<b>268</b>	74.9	83.2	80.0	79.5	66.2	75.7	<b>76.58</b>
H	24	36	43	37	40	43	<b>223</b>	86.5	78.6	74.8	83.7	72.3	77.6	<b>78.91</b>
I	147	135	174	209	183	262	<b>1,110</b>	68.9	77.7	75.2	68.0	73.7	52.3	<b>69.32</b>
J	25	95	32	40	69	36	<b>297</b>	71.2	22.4	50.5	56.7	35.1	50.8	<b>47.78</b>
K	66	45	28	63	36	66	<b>304</b>	66.8	100.0	74.8	86.8	74.6	67.6	<b>78.43</b>
L	375	696	273	281	583	581	<b>2,789</b>	74.1	46.4	71.7	73.9	63.6	47.1	<b>62.78</b>
<b>TOTAL</b>							<b>15,768</b>							<b>70.26</b>

Table 7.4: Scene-wise breakdown of annotations per annotator, including the number of labels, and agreement score (as defined in Section 7.2.2). The variation in agreement score is large, highlighting the difficulty of some of the scenes in comparison to others. For each tile, the set of 6 annotators (denoted ‘i’ to ‘vi’) is different, and the ordering of the columns is arbitrary.

VOC annotation format xml files, each one from a different annotator. Finally, there is a clustered annotation file, also in PASCAL VOC format [79].

Table 7.4 breaks down the number of individual annotations, and inter-annotator agreement (as defined in Section 7.2.2), for each tile. There is a large variation in the number of craters found within each of the 12 images, with the least cratered having 223 individual annotations, and the most having 3,954. We found there was a high variance in the number of labels produced by each annotator, with some annotators consistently marking fewer or more craters in all the images they annotated. Generally there was a ratio of 2–4 between the most and least numerous surveys on a given tile, this provides a strong argument for a multi-annotator strategy, which reduces the impact of this variance on the accuracy of results.

Returning to Table 7.1, we see information about the clustering of results, with average, minimum and maximum diameters for all images after clustering. During clustering, we first remove any annotations that have a diameter less than 3 pixels. Often, these very small craters are resolvable, however accurate labelling is difficult and may have resulted in unreliable final results. At 3 pixels or less, craters appear simply as a neighbouring pair of light and dark points. It is thus likely that the only craters found at this size are those with the steepest sides, which results in a higher contrast shadow-highlight pair. It is likely that many craters at this size range are missed, because they do not have a high enough contrast. Additionally, other features, such as pits, that are not of meteoritic origin, may easily be confused with a crater, because other physical characteristics (e.g. a circular rim) are not resolvable.



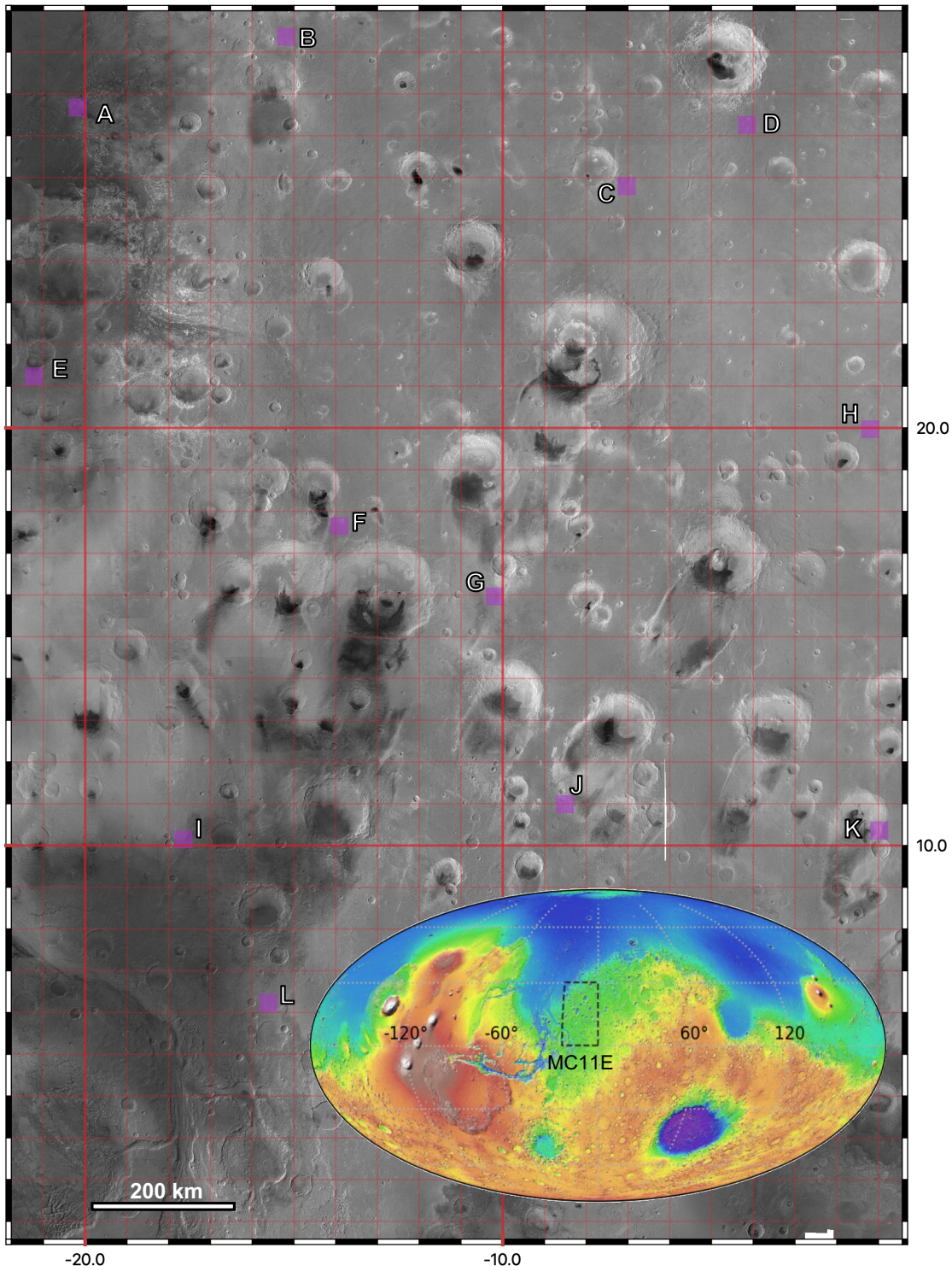


Figure 7.4: CTX Mosaic over MC-11 East quadrangle [190]. Purple squares indicate the 12 regions randomly sampled for annotations. The lower right shows a global elevation map of Mars from MOLA, with MC-11 East outlined.



Due to this uncertainty, annotations below 3 pixels were omitted. Users may choose to set a different lower diameter, at a higher diameter, in order to be more sure that the labels are more reliable and comprehensive, however providing the dataset with annotations down to 3 pixels allows for flexibility in how it is used. Because of this 3 pixel limit, many of the images have a minimum diameter of 18 m, whilst the upper limit varies much more, given the sporadic distribution of larger craters in our images. Median average diameters are consistently below mean averages, suggesting a skewed distribution towards lower size ranges, which is to be expected due to the negative power-law distribution of CSFDs.

Each PASCAL VOC format xml file contains information about the image it is related to, including path information, and its dimensions. It then contains a list of *object* fields which denote each crater label. The *difficult* field is used to denote when an annotator chose to express that they were unsure about a given annotation, whilst the *bndbox* field gives the boundaries of the crater. No geographical information is provided in the xml, with all dimensions in pixel coordinates. Many of the fields included in the xml file are somewhat redundant (e.g. *pose*, *truncated*) but are included to increase compatibility with software which expects PASCAL VOC format data as input. The same format is used for the clustered annotations, but they also contain an extra ‘confidence’ value in each object instance denoting the number of annotators who found that crater, between 1 and 6.

## 7.4 Interim Conclusions

This small crater survey offers a valuable training and validation set for researchers aiming to develop CDAs. In the specific case of deep learning, we believe a combination of this with other datasets (e.g. [11, 148]) may provide enough data at many scales, locations and with different instruments to create a highly generalised CDA, which is robust to changes in surface features, noise, crater morphology and crater size.

The ability to mark craters as ‘difficult’ was designed to give further control over the confidence assigned to the final craters. However, in practice, it was used too sparingly to be of much use in the analysis or validation of the data. Future efforts in dataset creation could find better ways of expressing each annotator’s confidence in their markings, giving more rich information about the confidence assigned to each final crater.

Many previous crater surveys have focused on annotating larger diameter ( $> 1$  km) craters, which brings with it different challenges. The global survey by Robbins et al. [11] used both Digital Elevation Models and infrared images, as opposed to high-resolution

images. Resolving these larger craters was likely less difficult, however the extreme erosion of some large craters, and their tendency to more frequently overlap with one another, posed difficulties that we were less affected by in our size range.

The multi-annotator approach we have used is scalable, and can be adapted to many other object detection tasks, both within and beyond remote sensing. We believe a more nuanced approach to training and validating models, which accounts for confidence, is particularly useful for problems wherein ground-truth data is unreliable, because of issues like noise and resolution. Framing an ambiguous problem such as small crater detection as a binary one leads to model performance being measured in a way that does not reflect real-world capabilities. Nevertheless, expert annotations are still likely to be of value in labelling tasks such as crater detection, because of the greater contextual information that experts have. For example, in very complex terrain with many non-crater features, it may be more useful to have few expert annotations, than many non-expert ones, given the higher number of potential false positives.

We believe, based on our experience in this project, that the extended training and seminar series conducted prior to labelling resulted in annotations with higher accuracy. In addition, the sense of genuine scientific collaboration (as opposed to labouring at a task one does not hold a stake in) between those involved was a strong motivator that helped annotators stay focused and productive during the labelling sessions. Future efforts that use this hybrid expert/crowdsourcing approach should continue to optimise the balance between speed (fewer repeated annotations and less training) and accuracy (more repeated annotations and more training) on a case-by-case basis. For small craters, we believe the community would benefit from future work that samples areas from across different parts of the planet, and provides more rich annotations of physical crater characteristics where possible (e.g. estimates of crater erosion, ellipticity, depth, etc.).

## 8 | Crater Detection Revisited

### 8.1 Motivation

The performance issues and computational inefficiencies of the CDA proposed in Chapter 6 indicated that—rather than designing a model from scratch—fine-tuning existing object detection models may be more fruitful. Compared to other object detection tasks, the data available for training was relatively scarce, initially comprising only the 6 tiles from the HRSC dataset used in Chapter 6. Now, with the CTX dataset described in Chapter 7, as well as other data sources (which are returned to in Section 8.2), the application of large, state-of-the-art deep learning object detection models seems more plausible.

As found in the case of cloud masking, sensor independence offered some significant advantages in situations where training data is spread across many satellites. In the case of cloud masking, a unique architecture was proposed to allow the model to ingest data from arbitrary multispectral visible/infrared satellites (SEnSeI). However, with regards to crater detection, all datasets come from *panchromatic* sensors. This simplifies sensor independence considerably, as the convolutional models proposed for all manner of single-sensor applications will be able to take data from any and all panchromatic sensors, if trained appropriately.

A more comprehensive approach to sensor independence could take into account and control for geometrical parameters, as well as spectral ones. For example, the viewing geometry, resolution, and illumination conditions could all be explicitly considered by a model, in the same way that SEnSeI is given explicit information about the wavelengths of light being measured. In the application considered here, however, these are mostly superfluous. This is because the illumination conditions can be made largely consistent (most sensors at Mars are sun-synchronous, and those that aren't—if used—could be corrected through image rotation), and the look angle is relatively close to the nadir for the satellites considered here, making the viewing geometry similar. Given this context, a more general sensor independence approach is not required for Martian crater detection, but could be of use in other applications.

So, the work in this chapter has three primary aims. First, show that the ORBYTS dataset, in combination with well-known object detection frameworks, leads to a state-of-the-art crater detection model. Second, to investigate the benefits of sensor independence

in a setting where it is significantly simpler to apply than in multispectral imagery. Third, to create a practical, well validated method to detect craters over large regions of Mars (which is then put to use in Chapter 9).

In Chapter 5, many experiments were conducted with test data from different sensors. This is because the primary aim of that chapter was to quantify SEnSel’s performance in different situations, and show sensor independent cloud masking to be a generally applicable technology for many sensors. In this instance, however, there is a specific scientific aim, which is to make a model that can detect craters over a large region of Mars. For that task, CTX is the most suitable sensor, as it is the one with global coverage that has the highest resolution. Running multiple experiments with test data on different satellites would not have directly aided this aim.

This chapter continues with a section covering the datasets used in the experiments. Then, Section 8.3 reports the choice of models, and how they are varied and trained in the experiments. The experimental results are set out in Section 8.4, showing both model performance and computational efficiency. Finally, the chapter ends by drawing some conclusions from the work.

## 8.2 Datasets

In total, four datasets are used in this chapter’s experiments, from three satellites. The HRSC dataset introduced in Chapter 6, and the CTX dataset introduced in Chapter 7 are both present. In addition to these, a further two datasets were added: one from CTX, and the other using THEMIS-IR. This section provides some details about these latter two datasets, which have not already been discussed.

### 8.2.1 Additional CTX data

During development of these models, Isidis Planitia became the primary region of interest for further study (further details in Chapter 9). Because of this, a test dataset from Isidis Planitia became desirable, as it would allow model performance to be measured in a way that was reflective of real-world use. This dataset is not used for training the models, but rather as the final test set on which performance is compared.

To this end, a colleague, Reuben Crawford Clarke, annotated three test sites from Isidis Planitia. Each site is 2370-by-2370 pixels across (roughly 12 km-by-12 km) and taken from the CTX mosaic produced by Murray Labs [192] at 5 m/pixel, Section 9.2 describes the mosaic further.

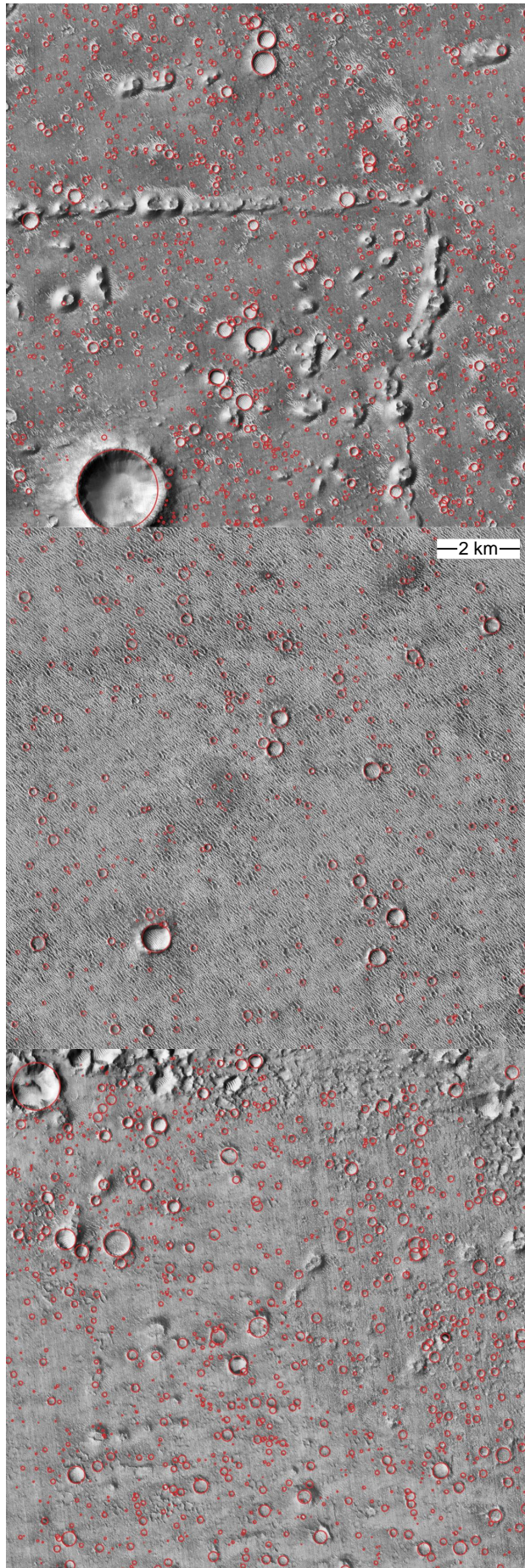


Figure 8.1: The three CTX images used as a test site over Isidis. Top pane shows *thumbprint* terrain, middle is *plains*, and bottom is *cones*. Labelled craters are marked in red. The scale bar is consistent across all three images.

The three test sites were annotated with the same software and approach as was used in Chapter 7, albeit with only one annotator. The three sites were annotated as comprehensively as possible, and contain 3230 craters in total with a diameter of 3 pixels or greater. The sites—along with their annotations—can be seen in Figure 8.1.

The sites were selected from three different regions of Isidis, each with different surface characteristics, in order to be as representative of the entire region as possible. The three terrains were “thumbprint”, “plains”, and “cones”. Thumbprint terrain consists of many small, thin linear features (top panel, Figure 8.1). Plains are regions with few distinguishing morphological features besides craters (middle panel, Figure 8.1). Finally, some regions of Isidis have rootless cones (bottom panel, Figure 8.1).

### 8.2.2 Global Crater Database

The global crater database [11]—as mentioned previously in Chapters 2 and 7—contains over 380,000 individual craters. This is by far the largest single resource of human-labelled craters on Mars. Whilst its existence makes large crater detection models somewhat redundant, if used with a sensor independent model, it may aid in training of a small crater detection model. Happily, craters of all sizes share many morphological characteristics. Because of this, large craters sampled at coarse resolutions look somewhat similar to smaller craters at finer resolutions.

The database of craters was helpfully converted into a format suitable for machine learning by DeLatte et al. [152], and is available for download [193]. This formatted dataset gives 30°-by-30° tiles, taken from the global THEMIS Daytime Thermal IR mosaic, which is at a resolution of 100 m/pixel. Whilst the database of craters is global, only those lying between  $\pm 30^\circ$  are used for this study. This is because projection begins to distort craters at higher latitudes, and so the appearance of craters would be very different.

For this study, the THEMIS daytime IR data is then preprocessed in order to more closely match the other datasets. Gaps with no data values in the mosaic are filled with the mean value of the mosaic tile, rather than with zeros, to minimise any sharp edges which might impede the model. The tiles are also further subdivided into 1024-by-1024 pixel patches, and any craters which are larger than 256 pixels (25.6 km) in diameter are discarded, because the models’ anchor boxes are already reduced in size, in order to optimise the detection of small craters, and so these would be too big for the models to easily detect. As well as these very large craters, any craters whose centre lies outside the image patch is also not included in the labels for that image patch. This preprocessing leaves a total of 220,000 craters for use in training.

## 8.3 Models

The models selected for this study cover two broad categories of deep learning-based object detection frameworks: *one-stage* and *multi-stage*. Object detection requires both localisation and classification, and so models can either do these simultaneously (as in the YOLO family of models), or in multiple stages (as in the models using R-CNN architectures). Further details about the development of these models can be found in Section 2.2.5, whilst this section explains what aspects of the model are changed during experiments.

### 8.3.1 YOLOv5

The effect of data and sensor independence was investigated by training YOLOv5 with a variety of different combinations of datasets. First, the *CTX* configuration was trained on the dataset made in Chapter 7, which represented a non-sensor independent set-up. *CTX+HRSC* added the dataset from [146], using all six tiles in training, and thus combining all the high resolution training data available to us. Similarly, the *HRSC* configuration only used this dataset, without the CTX data. *THEMIS* used only training data from the global crater database [11] in the THEMIS daytime IR mosaic. Finally, *ALL* uses all available data from CTX, HRSC and THEMIS for training. The data from THEMIS contained far more craters than the others, and so sampling was roughly balanced when used together, so that 50% of samples came from the THEMIS dataset. For all training configurations, roughly 10% of the dataset was held back for validation. The performance on this validation set was used to find where performance of the model had converged, and to stop training at this point.

For YOLOv3, using focal loss (see Section 2.2.2) for classification predictions was tried and shown to perform more poorly than the original binary cross-entropy [75], and so was discarded. YOLOv5, however, reintroduces the option to use focal loss, and so in this experiment, both a model with cross-entropy (equivalent to Equation 2.7 when  $\gamma = 0$ ), and a model with a focal loss ( $\gamma = 1$ ) were tested.

YOLOv2 [74] introduced a system for deciding the exact size and location of its anchor boxes by using k-means clustering on the training set bounding boxes, and was carried out specifically for Common Objects in Context (COCO) [194] and PASCAL VOC datasets [79]. YOLOv5 instead uses a genetic algorithm to find anchor box sizes which best fit the specific training data being used. Models with and without this feature enabled were tested.



Variable	Description	Initial	Evolved	% change
lr0	Initial learning rate	0.01	0.0105	5
lrf	Final learning rate	0.002	0.00173	-13.5
momentum	Momentum in weight updates	0.95	0.91	-4.2
weight_decay	Decay per iteration in weights	0.0005	0.00045	-10
fl_gamma:	Focal loss parameter	0.5	1.2	140
box	Bounding box loss factor	0.05	0.048	-4
cls	Classification loss factor	0.5	0.522	4.4
obj	Objectness loss factor	1.0	0.901	-9.9
iou_t	IoU threshold for using bounding box in training	0.2	0.2	0
anchors	Number of anchor boxes per output layer	3	4.57	52.3
degrees	$\pm$ range for rotation (degrees)	5.0	0.0	-100
translate	$\pm$ range for translation (fraction of size)	0.1	0.0949	-5.1
scale	$\pm$ range for zoom (fraction of size)	0.5	0.536	7.2
shear	$\pm$ range for shear (degrees)	1.0	0.0	-100
mosaic	Probability for mosaic augmentation	1.0	0.907	-9.3

Table 8.1: Hyperparameters varied in evolution. This table breaks them up into categories. Green (top 4 rows) affect optimization and weight updates. Yellow (next 4 rows) control the loss function. Red (next 2 rows) are related to anchor boxes. Finally, blue (last 5 rows) influence the data augmentation. Some parameters change dramatically (e.g. *fl\_gamma*) whilst others remain fairly stable.

Finally, YOLOv5 also offers a flexible hyperparameter evolution strategy, which randomly alters many hyperparameter values in the model, over successive runs, giving models which have good performance the chance to mutate, and discontinuing others. 105 runs were conducted over roughly 3 days to find an optimal architecture. The hyperparameters which were varied, and the values that were arrived at, are listed in Table 8.1. The hyperparameter evolution was only carried out for a model trained with all the available data, and so it is possible that different training set configurations would have led to different final values.

All configurations of YOLOv5 were trained using SGD and a batch size of 6, although the learning rate and momentum were varied as part of the evolution. Each model was trained for 60 epochs, of 1000 steps each, or until performance had converged.

### 8.3.2 Mask R-CNN

Mask R-CNN (see Section 2.2.5 for details) was used without major alterations to the core architecture, however some parameters were tweaked from their default settings as laid out by He et al. [72]. The size of the anchor boxes was reduced, to facilitate better detection of small objects. The anchors were halved in size, so that the smallest anchor box had a width and height of 16 pixels. Given that all annotations assume a



circular crater, the sets of rectangular anchor boxes with different heights and widths were removed, leaving only the sets of square boxes. The maximum number of objects detected in an image was increased from 100 to 500, to ensure that—where craters were highly concentrated—the model would not limit the total number of craters found.

Training of Mask R-CNN took several days of computing time, which is orders of magnitude longer than the dozens of minutes YOLOv5 takes to train. Fine-tuning the model’s parameters over many training runs was made more difficult by this. As a result, the aspects of the model which were varied were more restricted, and instead of the more comprehensive evolution strategy used for YOLOv5, the model was varied in three ways.

First, the training set was set to one of two configurations listed in Section 8.3.1. Namely, *CTX*, and *ALL*. Second, the model’s backbone was set to either *ResNet-50*, or *ResNet-101*, which allows us to see if the size and complexity of the CNN used is limiting performance or whether the 50-layer ResNet is already sufficiently complex. Thirdly, the data augmentation is varied, with three different settings:

- **None:** The data was given to the model in an unaltered form.
- **Light:** This level of augmentation focused on geometric transformations. Each image could be randomly translated, cropped, rescaled, or rotated within  $\pm 5^\circ$ .
- **Heavy:** This level included the geometric augmentations applied in the *light* setting. In addition, the images were altered using additive white noise, salt and pepper noise, Gaussian blurring, brightness and contrast adjustments, all with a 50% chance of being applied to any given image.

Whilst Mask R-CNN provides segmentation masks of the objects it detects, these were not used directly during these experiments. The labels used in training all assumed a completely circular crater, which meant the loss associated with the mask converged to zero very quickly, as the network learnt to simply draw a circle mask for every example. Essentially this made Mask R-CNN into Faster R-CNN, as only the bounding boxes were of interest. This was done for practicality, because the code for Mask R-CNN was readily available in a format that could be customised in the desired ways <sup>1</sup>.

During training, SGD was used with an initial learning rate of  $1e-3$  and a momentum of 0.95. The convolutional backbone of Mask R-CNN (ResNet) was loaded with pre-trained weights, taken from a model trained on the COCO dataset [194]. The large size of Mask R-CNN meant that a batch size of only 1 was possible, on a GPU with 8 GB of memory. Future work may find performance improvements by using a larger batch sizes on systems with more available memory.

<sup>1</sup>Available at: [https://github.com/matterport/Mask\\_RCNN](https://github.com/matterport/Mask_RCNN)

## 8.4 Results

### 8.4.1 Performance

Table 8.2 shows the performance of different configurations of YOLOv5 on the CTX test set from Isidis Planitia. Several interesting conclusions can be drawn from these results. First, focal loss was essential in training YOLOv5. When  $\gamma$  was set to 0 (making the focal loss equation equivalent to categorical cross-entropy), the model failed to train, and detected no craters.

The data used for training had a large impact on performance. Training with all available data led to the best models, showing sensor independent crater detection is beneficial when training data in a single sensor is limited. Surprisingly, the model trained only on the global database in THEMIS daytime IR fared better than models trained on the sensor used in testing (CTX and CTX+HRSC). This indicates that the huge size of the THEMIS dataset, at an entirely different resolution range to the test data, benefits the model more than the smaller but more similar datasets. Evolution of the hyperparameters led to a moderate increase in performance, with 4.1% higher mAP@50% when compared with the model using default parameters. Results for some of the YOLOv5 configurations are visualised in Figure 8.2, in the same way as was done previously in Chapter 6.

Eight Mask R-CNN configurations were tested in the same way as YOLOv5 (Table 8.3). Fewer training set varieties were explored in this set up, because of the time taken to train each model, however the same clear trend can be seen in comparing results trained only on the CTX dataset, in comparison to those trained on all available data. Augmentation seems to help most when training data is limited to only CTX, leading to a 5.7% increase in mAP@50%, whereas when using all available data it adds only 2.1%. This is intuitive, as augmentation—which adds more variety to data—would most benefit the model with limited amounts of data to begin with. Interestingly, using a larger 101-layer convolutional backbone leads to a slightly lower mAP@50% and  $F_1$  than the best ResNet-50 varieties. Figure 8.3 plots all the predictions of models with ResNet-50 on an example from the test set.

### 8.4.2 Computational Efficiency

The speed at which a model can predict crater locations affects how useful it is for real-world application. The model in Chapter 6 was too slow to be applied to large quantities of data. Therefore, the running time of both Mask R-CNN and YOLOv5 were measured.

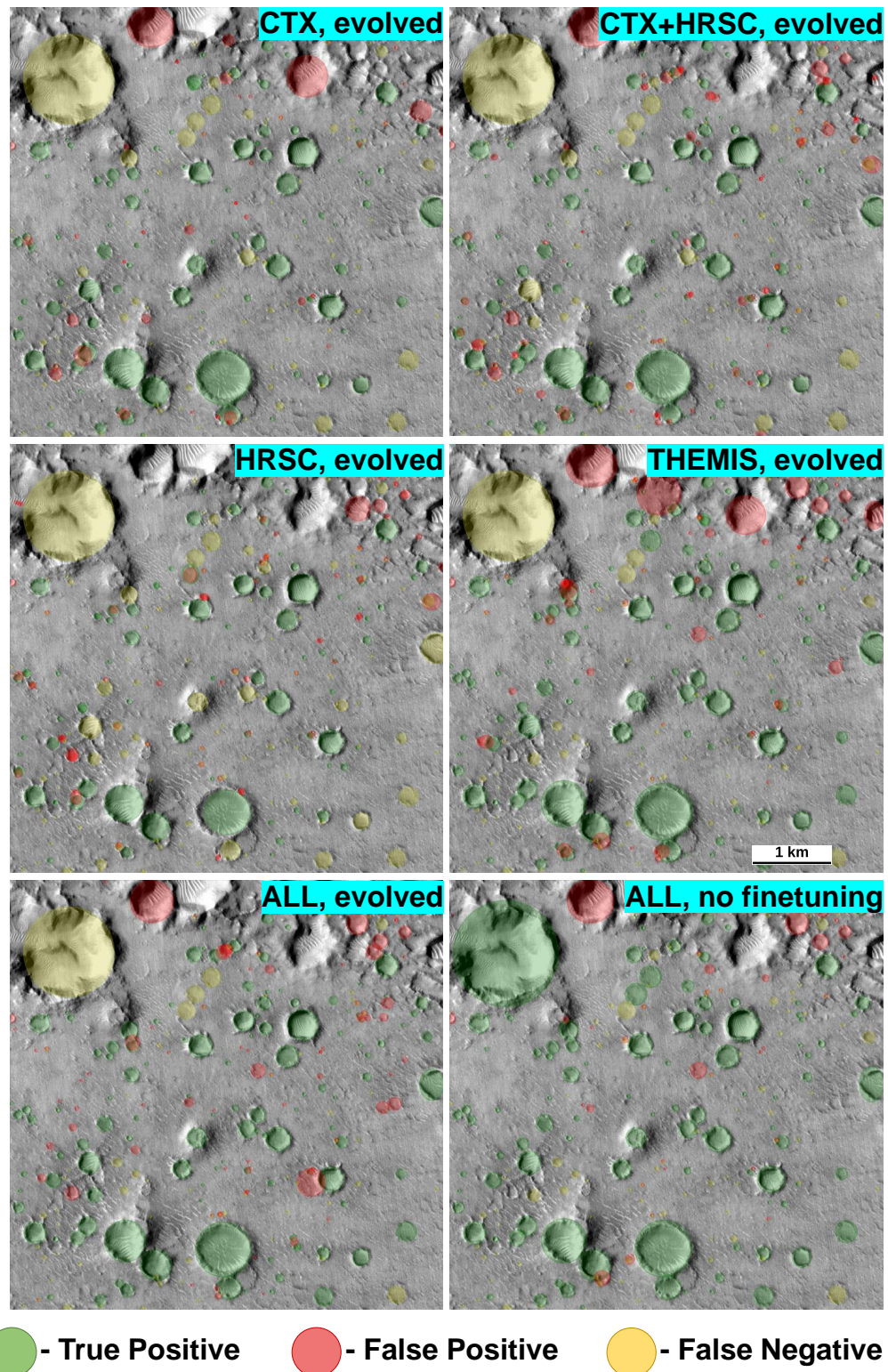


Figure 8.2: The results of several YOLOv5 model configurations are shown over a part of the test set. Five of the models use the evolved hyperparameters with different training set combinations. The bottom-right model is the one which used the default hyperparameters, and no autoanchors, but did use a focal loss. Unfortunately, due to human error, some craters were inevitably omitted in labelling, meaning some of the false positives are in fact correct. This artificially decreases the models' performance, meaning true performance may in fact be higher than is measured here.



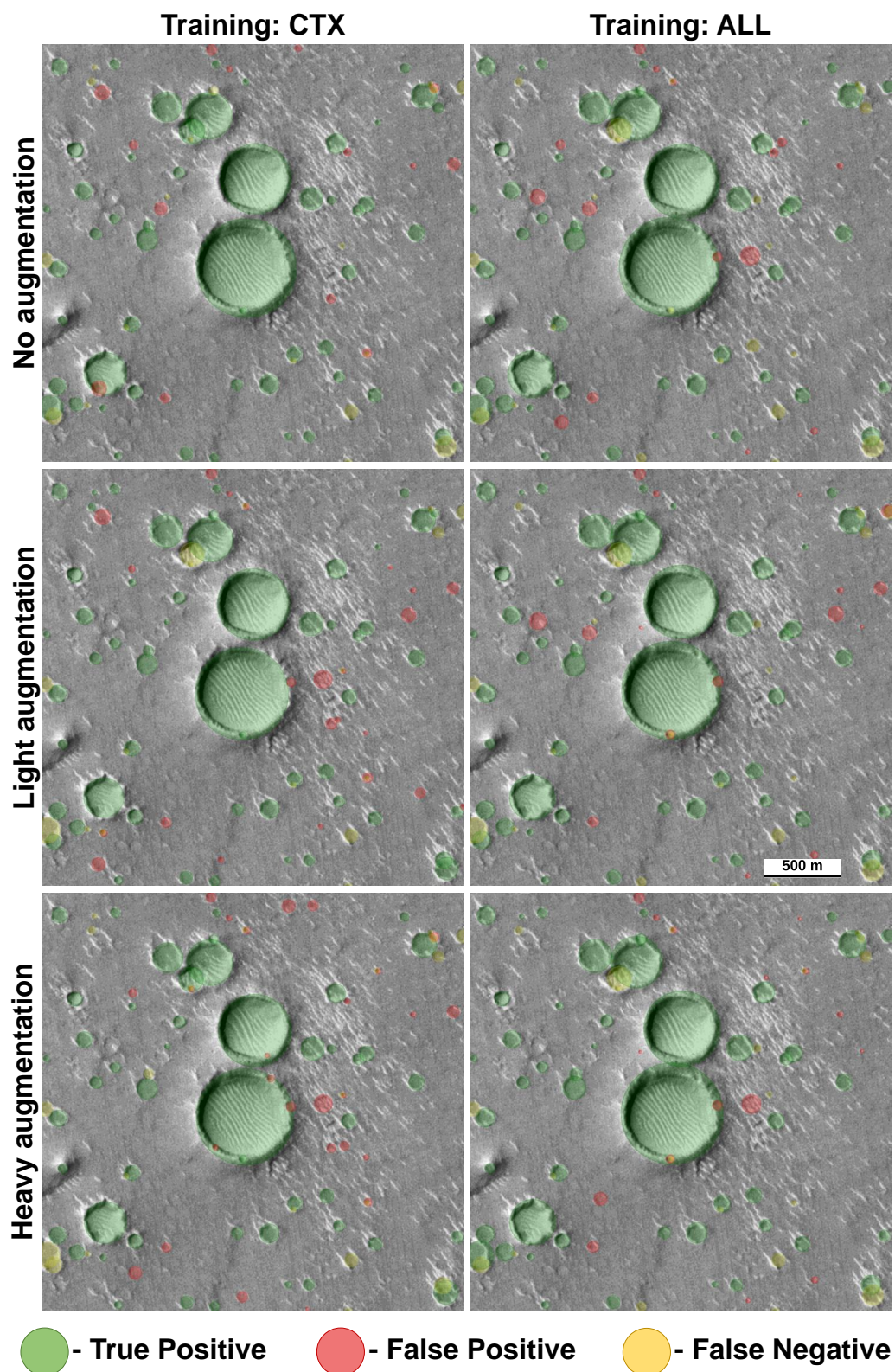


Figure 8.3: Visualisation of the predictions from several Mask R-CNN configurations, all with the ResNet-50 backbone. The two columns show models trained on different combinations of training data, whilst each row corresponds to a different level of data augmentation.

Training Set	Loss	Auto-anchors	Evolved	mAP@50%	Precision	Recall	F <sub>1</sub>
ALL	Cross-entropy	N	N	0	0	0	0
ALL	Focal	N	N	60.2	<b>73.1</b>	53.2	61.6
ALL	Focal	Y	N	64.5	69.4	61.2	65.0
CTX	Focal	Y	Y	52.4	62.2	54.0	57.8
HRSC	Focal	Y	Y	34.8	53.6	40.9	46.4
CTX+HRSC	Focal	Y	Y	52.1	43.9	58.5	50.1
THEMIS	Focal	Y	Y	60.5	70.4	54.5	61.4
ALL	Focal	Y	Y	<b>68.6</b>	63.1	<b>67.9</b>	<b>65.4</b>

Table 8.2: Results for YOLOv5 model configurations on the test set. Both auto-anchors and hyperparameter evolution benefitted the model. Even more substantial was the improvement when using larger amounts of training data, from multiple sensors. When using cross-entropy loss, the model failed to predict any craters.

Training Set	Backbone	Augmentation	mAP@50%	Precision	Recall	F <sub>1</sub>
CTX	ResNet-50	None	61.7	67.2	62.2	64.6
CTX	ResNet-50	Light	65.4	54.9	67.3	60.4
CTX	ResNet-50	Heavy	67.4	47.3	<b>72.3</b>	57.2
ALL	ResNet-50	None	70.5	<b>70.0</b>	66.5	68.2
ALL	ResNet-50	Light	71.3	68.7	68.2	<b>68.4</b>
ALL	ResNet-50	Heavy	<b>72.6</b>	65.4	70.1	68.0
ALL	ResNet-101	None	71.5	62.7	70.9	66.6
ALL	ResNet-101	Heavy	70.0	69.1	65.3	67.1

Table 8.3: Mask R-CNN's performance on the test set, with eight different configurations. Changing the augmentation levels and training set affected the model's performance, whilst increasing the size of the convolutional backbone from 50 to 101 layers made no difference, or perhaps even a small negative one.

Both models were given 100 images at 1024-by-1024 pixels across. The time taken to compute predictions for each was measured, and then converted into a time per megapixel. Table 8.4 gives these times for both models, and also compares them to times derived from the values in Table 6.3. YOLOv5 was shown to be significantly faster than other models, whilst Mask R-CNN was still much faster than the model proposed in Chapter 6.

Model	Best mAP@50%	Inference time per megapixel	Megapixels/s
YOLOv5	68.6	7.3 ms	137
Mask R-CNN	72.6	715 ms	1.40
CDA, stride=6*	59.5	27.6 s	0.0362
CDA, stride=18*	53.0	10.7 s	0.0932

Table 8.4: Computation time for all the crater detection models tested. YOLOv5 is roughly 100 times faster than Mask R-CNN, which is in turn around 40 times faster than the model from Chapter 6 with a stride of 6. \*taken from Table 6.3

## 8.5 Interim Conclusions

This study sought to apply the dataset created in Chapter 7 to create a reliable CDA. This has been achieved, with Mask R-CNN producing visually impressive results that outperform the one-stage object detection method YOLOv5.

In Chapter 7, it was shown that the average *agreement score* between human annotators was around 70.3%. Given that multiple annotators were not used for our test data here, it isn't possible to compare the model's results with those of human annotators in the same way. However, Mask R-CNN's recall and precision were 65.4% and 70.1% respectively, coming close to the average 70.3% *agreement score* of human labellers. Whilst comparing recall and precision with the *agreement score* is inexact, their similarity suggests that Mask R-CNN is not too far from non-expert human-level performance.

Returning to the idea of data, performance and utility in Section 1.3.2, both YOLOv5 and Mask R-CNN have clear advantages over the model presented in Chapter 6. The models had access to more training data, thanks to sensor independence, and because of this, more of that data could also be used as a test set. Performance also clearly improved, with significantly higher  $F_1$  scores and mAP@50%.

The utility of the automated crater detection was also increased substantially, by reducing the computation time, so that predictions could be made orders of magnitude more rapidly. Interestingly, the huge speed advantage that YOLOv5 has does not add a huge amount more utility in this application. This is because Mask R-CNN—whilst still much slower than YOLOv5—can process large sections of Mars in a matter of hours, where YOLOv5 would take minutes. Crater detection is not a time-critical task, and so researchers spending a few hours waiting for detections is not too onerous.

The trained crater detection model opens the door for researchers to rapidly survey large areas in CTX imagery, for small, sub-km diameter craters. The combination of high performance and computational speed makes its deployment practical. The next chapter investigates some of the scientific possibilities which this new tool opens up.

## 9 | Cratering over Isidis Planitia

### 9.1 Motivation

This study is a natural extension to the development of CDAs in Chapter 8, providing a case-study for an application of deep learning with optical satellites leading to scientific outcomes. The formation and distribution of small crater populations (taken here to be those under around 1 km in diameter) are not well understood, due to several poorly constrained factors which affect their production and erosion through time and space. Incomplete knowledge of these processes leads to an inability to estimate the age of surfaces based on small crater counts, which are often the only option given that young surface units do not have enough large impacts on them to make statistically reliable counts. These ambiguous factors include atmospheric filtering (impactors burning up, breaking apart, or at the least being decelerated), fluvial and aeolian erosion, dust deposition, surface hardness, and the flux of small meteorites [140]. Of particular interest to researchers has been the question of the extent of secondary cratering due to larger impacts, thought to artificially increase the number of small craters.

There is substantial disagreement as to the extent of secondary craters' effects [195]. We know secondary craters exist based on measurements from Earth, e.g. from a study on a 100 kt nuclear explosion in Sedan [196], finding innumerable small craters over 1.5 km from the blast. Secondaries have also been considered in planetary research since the 1960s (e.g. [197, 198]). More recent results have suggested that large impacts can produce massive quantities of secondaries.

For example, McEwen et al. [199] estimated that Zunil crater on Mars was responsible for  $10^7$  secondaries between  $10m \leq D \leq 200m$ , and Dundas et al. [200] estimated  $10^6$  secondaries of  $D \geq 63m$  were created by Tycho crater on the Moon. These results both found secondary craters appearing very far from the primary impact, especially at smaller diameter ranges, with those from Zunil extending up to 1600 km away from the impact. They were found to be concentrated in rays emanating from the primary impact. If such numerous and far-flung secondaries are the norm, rather than exceptional examples, then we might expect that the majority of small craters are in fact of secondary origin.

Besides secondary craters in rays, some also theorise massive populations of secondaries landing far from their parent impact without any of the usual visual characteristics

of secondaries at oblique angles (e.g. [201]), making them morphologically indistinguishable from true primaries. So far, it has been impossible to measure the number or distribution of these primary-like, distant secondaries.

It is clear that the nature of secondary cratering is complex, situational, and difficult to disentangle from other processes. Isidis Planitia was selected as the region of interest for this chapter, because of its relatively consistent geology (see Section 2.5 for details), which simplifies some of these poorly understood processes. This study focuses on some specific issues that are pertinent to the current disagreements in the field, but does not seek to fully parameterise secondary production:

1. Whether secondaries dominate primaries at small diameters, and if so, the size range at which this transition occurs.
2. What the local effects of large craters are on small crater distributions in their immediate surroundings, through both obliteration and secondary production.
3. Whether primaries over Isidis show signs of rayed secondaries, as was observed at Zunil and Tycho.
4. If there is any way of separating the population of distant, circular secondaries from primaries of the same size (not by individual craters' characteristics, but by their distributions).

Broadly, the estimation of secondary cratering effects has been approached in two ways. (i) physical modelling and simulations of meteoritic impacts, leading to predictions of the distribution of trajectories and particle masses that lead to secondary impacts. (ii) manual flagging of craters which are suspected to be secondary, due to some physical cues (e.g. low depth-to-diameter ratio, clustering, and “herringbone” ejecta) and studying how they correlate with larger primaries, often in the form of rays.

The goal of this study is to analyse small crater populations empirically, without assuming any small crater is or is not a secondary, using an array of geo-spatial statistical techniques. These provide a third, independent account of the nature of secondary cratering (and small crater erosion). This approach is possible only with automated detection, as the number of craters used (several million) cannot feasibly be counted by hand.

The statistics used in this study comprise both familiar metrics in crater studies (size-frequency distributions and densities) along with new measures. These new measures come from the distribution of Nearest Similar Neighbours (NSNs), which is a tweaked



version of Nearest Neighbour analysis (originally applied to population studies in ecology [202]) that is well-suited to craters. With these NSNs, novel experiments are conducted which focus on the angles between pairs, hinting at the directions from which impactors arrived.

The chapter continues with Section 9.2, outlining how detections were collected across Isidis. Section 9.3 compares the detections from the two deep learning models, and justifies the selection of Mask R-CNN for further study. The spatial statistics of these automated detections are then studied in Section 9.4, introducing each statistical experiment along with its results. This is followed by some concluding remarks in Section 9.5.

## 9.2 Crater Detection over Isidis

This section outlines the procedure used to detect craters over Isidis Planitia. We used a global mosaic [192], sampled at 5m/pixel. This was made from CTX images which were co-registered and illumination-corrected by *The Bruce Murray Laboratory for Planetary Visualization* and is freely available for use from their website<sup>1</sup>. The data is distributed in gridded 2°-by-2° tiles. This mosaic has the advantage of being ready-to-use, and almost gap-free. However, it is likely that for a study region the size of Isidis, a custom made mosaic may have given better results. The Murray Labs mosaic relies on hand selection of images, with the products with the best image quality being chosen. However, given that more images have been collected since the mosaic’s release, it is likely that better quality images could be found. However, creating a mosaic specifically for the project was not possible due to time constraints, and because it would mean that any extension of the crater detection to new areas of Mars would also require a mosaic to be made.

The geological map from the USGS (Scientific Investigations Map 3292) [203] is used to define the Isidis Planitia basin unit. As well as the basin unit, we also include the crater at 10.2°N, 94.3°E and its ejecta curtain, located on the eastern edge of Isidis, as it overlaps strongly with the basin and looks like a later feature (Figure 9.1). Those 2°-by-2° tiles which overlap the defined study area are fed into a sliding window for crater detection.

The sliding window takes crops (in our case 1024 pixels squared) and uses an object detection model to return a list of possible crater detections, as bounding boxes and confidence scores. At this stage, all candidates are kept, including those with low confidence values. The crops taken by the sliding window overlap by 64 pixels, in order to make

<sup>1</sup><http://murray-lab.caltech.edu/CTX/>

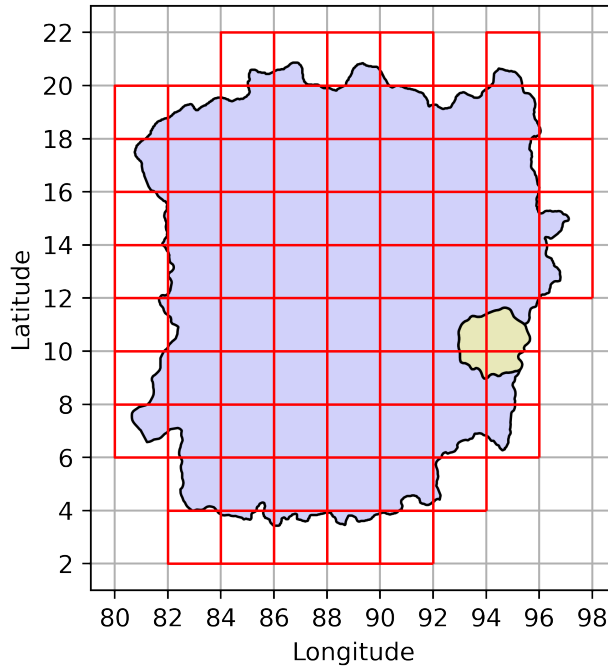


Figure 9.1: Murray Labs CTX Mosaic grid superimposed on the geological units of Isidis basin (blue) and the adjacent crater unit (yellow). The grid boxes in red are the areas processed by the model. Afterwards, craters whose centres do not intersect with the geological units are discarded.

sure the detection algorithm performance does not suffer due to boundary effects (e.g. craters straddling the cropped image’s edge). Detections from the overlapping border are discarded, so that detections aren’t needlessly duplicated.

Craters vary hugely in size, and whilst the models trained in the previous chapter are capable of detecting craters over a wide range of scales, it is still necessary to create detections on a pyramid of different resolutions. The finest sampling takes images at an effective scale of 2.5 m/pixel – twice the resolution of the actual data, in order to maximise the ability of the model to detect what was selected as the minimum diameter for the crater survey (15 m). This is because, as we saw in Chapter 8, the models tended to be sensitive to detections down to around 6 pixels. Then, the data is also sampled at 20 m/pixel, and 80 m/pixel, to capture the larger craters. Both upsampling and downsampling of the CTX data is done using bilinear interpolation.

These scales are chosen as a balance between speed and accuracy, with the diameter range of detections from each scale overlapping comfortably with those from adjacent scales, so as to not miss craters at any size range. Downsampling beyond around 80 m/pixel only provides large craters, which we already know about, based on the global catalogue of  $D \geq 1km$  craters produced by Robbins et al. [11], and whilst it is useful to compare our detection results against these, it is not essential to go up to the largest of diameters.

With detections at several scales, we now have many overlapping, repeated detections. We find pairs of overlapping craters, and keep the instance with the highest confidence value from the model. Any craters which have an intersection-over-union of more than a third are taken to be overlapping, as this is roughly the limit of morphologically separable craters. The intersection-over-union is calculated with respect to the actually circular crater rims, rather than the square bounding boxes around them. Roughly 20% of craters are detected twice or more, so this process is vital for removing duplicate entries.

At this stage, we have retained all crater detections, regardless of confidence values assigned by the model. We use three independent pieces of information to select a good confidence threshold, below which to discard detections. First, we find a rough density for craters over Isidis which are resolvable in CTX imagery, by averaging over the test dataset developed over Isidis, which was used in model development to measure performance. This leads to a density of  $7.5/km^2$  for all craters (which is dominated by small craters, due to the power law of CSFDs [204]). Second, the global catalogue of craters with  $D \geq 1km$  from [11] is used to generate a CSFD, which can be fitted to our own above 1 km. Third, the experiments of Chapter 8 indicate a confidence value for a good balance between precision and recall.

All three considerations lead to an optimum confidence threshold of 0.7 for Mask-RCNN, whilst YOLOv5's confidence threshold needs to be tuned to 0.25 (this is explained in more detail in the following section on data validation). The fact that all three measures lead to the same threshold value is a good sign that Mask R-CNN is a robust CDA. Finally, we also discard craters below 15 m diameter, as it is unlikely that anything below this was resolvable. This leaves us with a total of 5,499,191 detections in the case of Mask R-CNN, and a very similar number for YOLOv5.

## 9.3 Data Validation

### 9.3.1 Inspection of large craters

In the subsequent experiments, the global catalogue of craters from Robbins et al. [11] will be used as a basis for where large ( $D > 1km$ ) craters lie on Isidis Planitia. Therefore, it is important that these are inspected in order to ensure that the craters used are correct. The catalogue is as comprehensive as possible, with every crater found annotated – even those which are highly eroded. Some of the impact craters found over Isidis are extremely faint. For craters this large to be eroded, means they must be very old. Any secondaries

they produced are likely also long gone, as the smaller secondaries would be eroded more quickly than their parent primary.

A visual inspection was carried out to flag and remove the most eroded craters on Isidis, which are too old to have had an effect on the small crater distribution in their vicinity. Figure 9.2 gives an example of one of the eroded craters which were removed. In total, 18 craters were removed for being too eroded, with the majority of these being from the largest craters ( $D > 10\text{km}$ ). This left a total of 1794 large craters over Isidis Planitia and the adjacent impact unit.

### 9.3.2 Size-Frequency Distributions

The most widely used method for analysing crater statistics is to use the CSFD (e.g. [17,135,204,205]). The non-cumulative CSFD itself suffers from noise if too many bins are used, therefore it is more practical to use the *cumulative* distribution, where  $CSFD(x)$  is the total density of all craters with  $D \geq x$ . Note, CSFDs are accumulated as diameters *greater than*, rather than less than, the given value, which makes it easier to visualise due to the negative power law. CSFDs are most commonly plotted on logarithmic scales in both diameter and densities, making the strong negative power law of the distribution clear. The gradient of this logarithmic line is the exponent of the power law.

The CSFD of our detections are shown for both Mask R-CNN and YOLOv5 in Figure 9.3. Both models agree closely with the CSFD derived from Robbins et al. [11], although at very large diameters both models fail to find craters (mostly because the scales used for detection did not go up this high). This is not a problem, as craters this large are included in the global database anyway. We also see that both models have roughly the same total density as the annotations. However, YOLOv5 has a significant roll-off in detection rate at the lower diameter range, suggesting it struggles to detect craters close to the resolution limit, less than around 40 m, or 8 pixels, when compared to Mask R-CNN.

Fitting the CSFDs to isochrons (functions which model the crater distribution of a surface at a given age) can provide an estimate for when Isidis was last resurfaced. By comparing the ages derived using this method for the Mask R-CNN detections, and those from Robbins et al. [11], we can validate the automatically detected craters in an applied scientific use-case. To fit the isochrons, the *craterstats* package [206] was used. This package draws on work from several papers [188,204,207–209]. The distribution is fitted to the isochrons using Poisson timing analysis [209], more details of which can be found in the paper, and in the GitHub repository for the package [206] (see Figure 9.4).

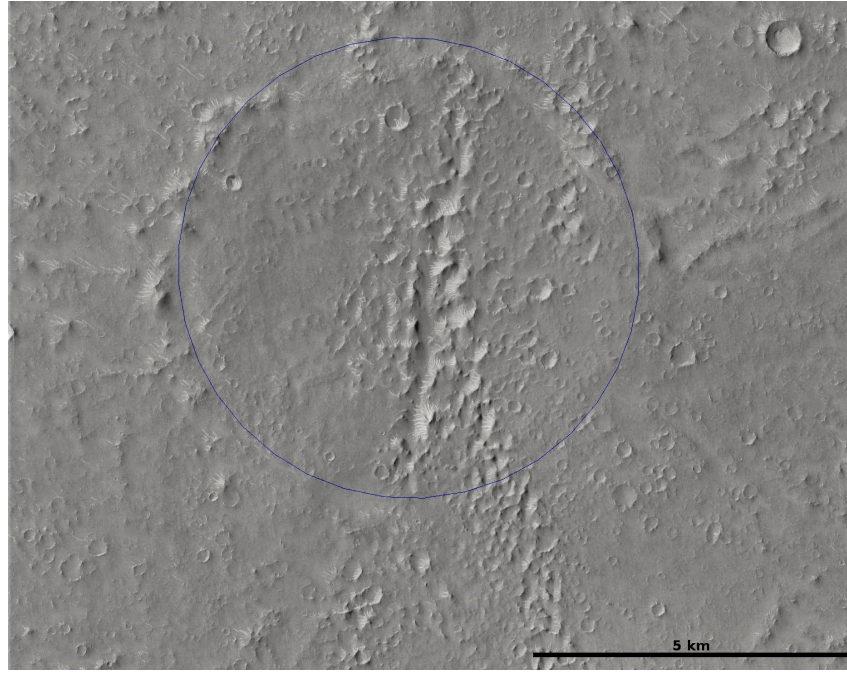


Figure 9.2: Example of a crater on Isidis which is too eroded to be included in the study. The original circular rim is now just a pattern of isolated ridges forming a circle, and the bowl has been infilled. This image is taken from the Murray Labs CTX mosaic [192], and has been contrast-adjusted for easier visualisation.

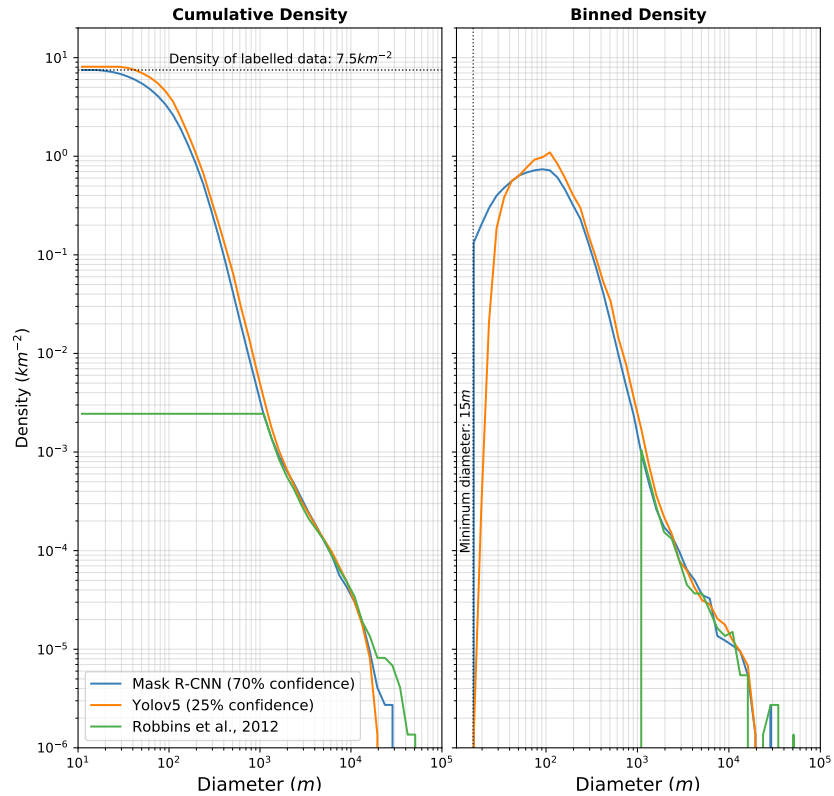


Figure 9.3: CSFDs for both models, and the one found by Robbins et al. [11] (including the 18 craters removed from the survey for later experiments). Left-hand side shows the cumulative distribution, and the right the binned densities. There is close agreement between both models and the Robbins dataset from 1km to 10km, at which point the detection rate of our models declines. The total densities of the model are similar, however YOLOv5 struggles at the smallest diameters ( $D < 50m$ ) compared to Mask R-CNN.

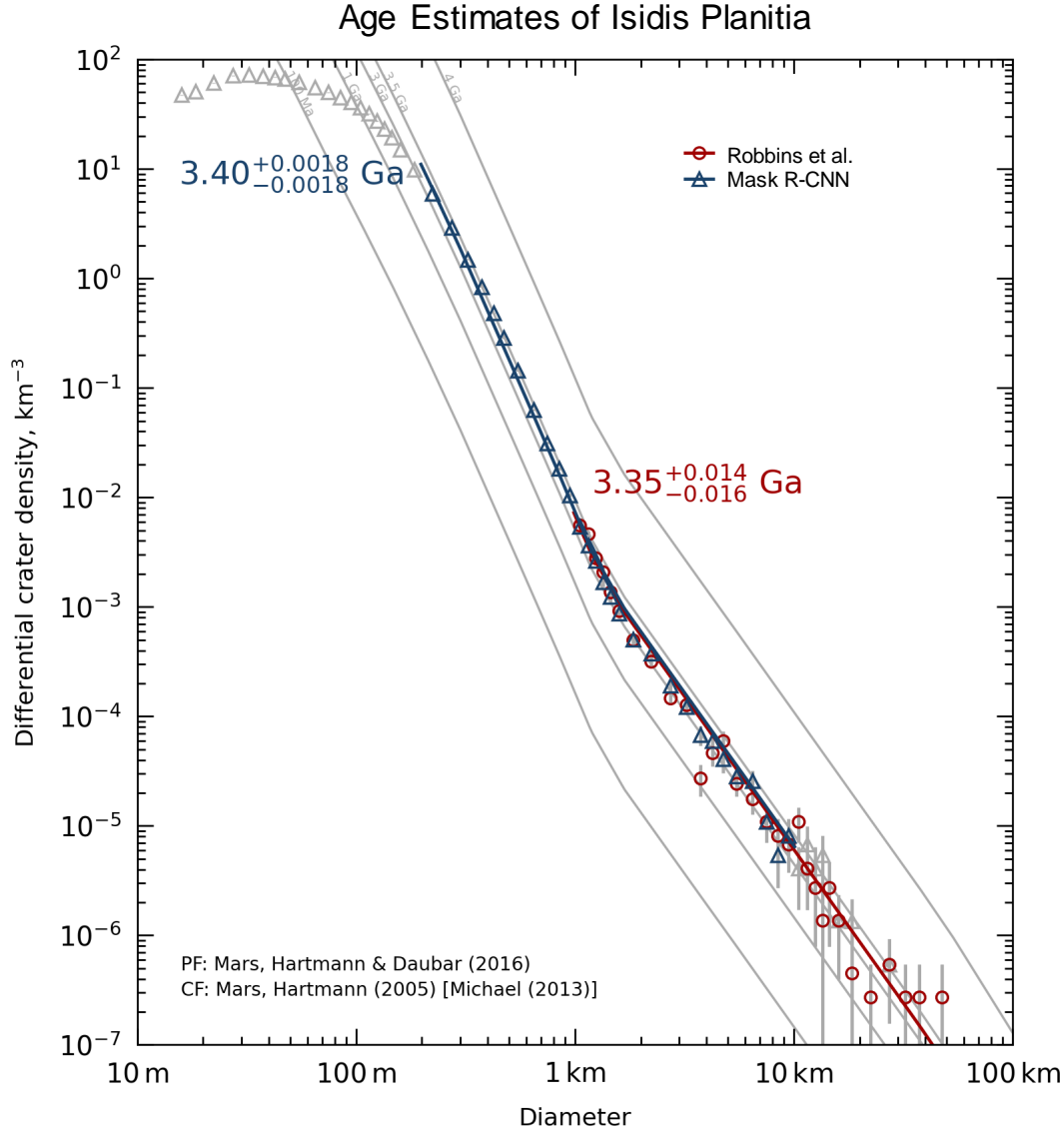


Figure 9.4: Age estimates for Isidis Planitia using Poisson timing analysis. The craters with diameters below 200 m are omitted from the fitting for Mask R-CNN, as it is clear that they begin to deviate from the isochrons due to resolution limits. The age estimates from the two crater surveys are very close, with only 50 million years between them. They also agree with other studies looking at the history of Isidis. For example, Ivanov et al. [144] estimated that there was significant resurfacing of the basin due to fluvial or glacial processes between 3.1–3.4 Ga, which agree well with the results here. The approach used is somewhat crude, given that the crater counts over the whole of Isidis are pooled together, and not separated into different geological units as in Ivanov et al. [144]. However, this result shows that Mask R-CNN can perform crater surveys which are scientifically consistent with accepted results.

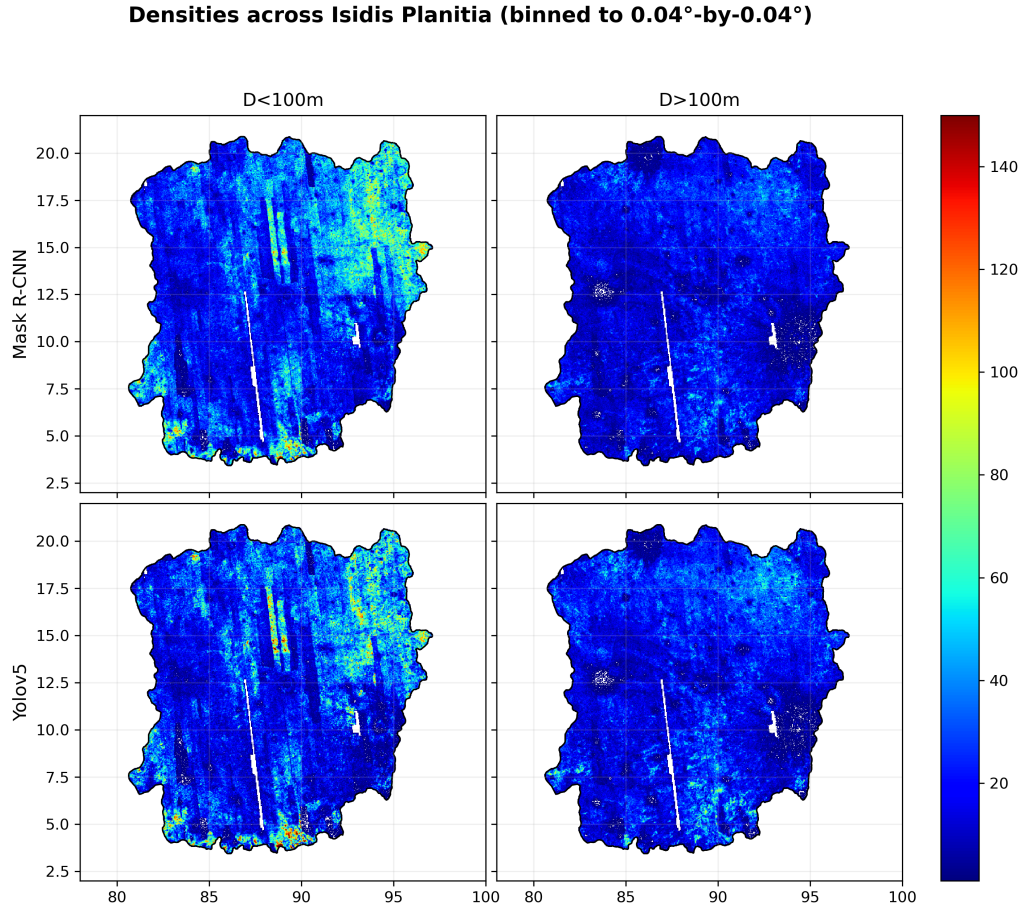


Figure 9.5: Crater detections at  $D < 100m$  (left column) and  $D > 100m$  (right column) for Mask R-CNN (top row) and YOLOv5 (bottom row). The heatmap’s scale is clipped at 150, to allow for better visualisation. The smaller size range is strongly affected by image quality, with distinct artifacts arising from the boundaries between images used in the mosaic. YOLOv5’s detections seem slightly more susceptible to this, with more detections in high density areas, and less detections in low density ones when compared to Mask R-CNN’s. At the higher diameter range, both models are less affected by image noise, because larger craters are more likely to still be resolvable despite radiometric noise.

### 9.3.3 Density over Isidis

As well as CSFDs, we can also consider the density of crater detections by Mask R-CNN and YOLOv5 across space. Whilst an in-depth analysis of their spatial statistics is left for Section 9.4, an initial look at the way the detection densities vary across the surface of Isidis can tell us about the ways in which the model is interacting with the data, and inform the decision as to which model results are more suitable.

Figure 9.5 shows how the detection rate varies across Isidis. The CTX image mosaic’s structure can be seen clearly in the density of detections, with some CTX image strips having many more detections than others, at lower diameter ranges. Looking at some examples of the image boundaries in the mosaic it becomes apparent that the lower quality images are too noisy for very small craters to be detected by the model. Unfortunately,

this is not easily rectifiable. However, the relative sensitivity of the models to the transitions between low and high noise images hints at which more model is robust. Visually, based on Figure 9.5 it seems that Mask R-CNN finds craters in a more even distribution across the image boundaries, suggesting it is more resistant to CTX radiometric noise.

Using both CSFD analysis, and density heatmaps, Mask R-CNN's and YOLOv5's behaviour can be examined in an application-dependent context. In this real-world context, Mask R-CNN is better equipped to find very small craters, and more resilient to radiometric noise in CTX data, making it a more reliable model to use for the study. Therefore, from here onwards in the study, the detections from Mask R-CNN are used.

Originally, investigating the densities of craters using heatmaps seemed a promising way to analyse the spatial distribution of craters. For example, rays of secondaries may have been visible around large primaries. However, inspection of Figure 9.5 suggests that simple density heatmaps are not sufficient, given their coarse resolution and noise, and so other spatio-statistical methods are needed to study the nature of the crater distribution.

## 9.4 Statistical Analysis

### 9.4.1 Annular density variations

To ascertain the effect of large craters on the small crater populations in their surroundings, we must measure some quantity as a function of the radius from that large crater. An annulus is defined as the area bounded by two circles of different radii, both centred at the crater centre. By plotting statistical metrics across binned distances from large craters, their impact can be seen as a function of distance from their centres. The dataset from [11] is used to locate all of these primary craters, excluding those which were found to be too faint by visual inspection, as detailed in Section 9.3.

The size of the annuli around the large impacts is proportional to the diameter, and can overlap with other craters' annuli (Figure 9.6). Where overlapping, small craters are allowed to be used multiple times in the calculations, as they are in some sense more affected by the primaries than those craters which are only close to a single large primary impact. We measure the density of craters in each of these annuli, and inspect its ratio with the average density for that diameter across the whole of Isidis, allowing us to isolate the impact of the large craters on the distribution.

Figure 9.7 shows how the densities of small craters change as a function of distance from larger ones. Across all primary crater sizes, the annuli close to the crater's rim are



highly depopulated, suggesting obliteration of craters extends up to around 3-4 crater diameters from the centre. Beyond this distance, densities return to the average of Isidis, or at least to a constant rate that is not too dissimilar from the average density of detections over Isidis.

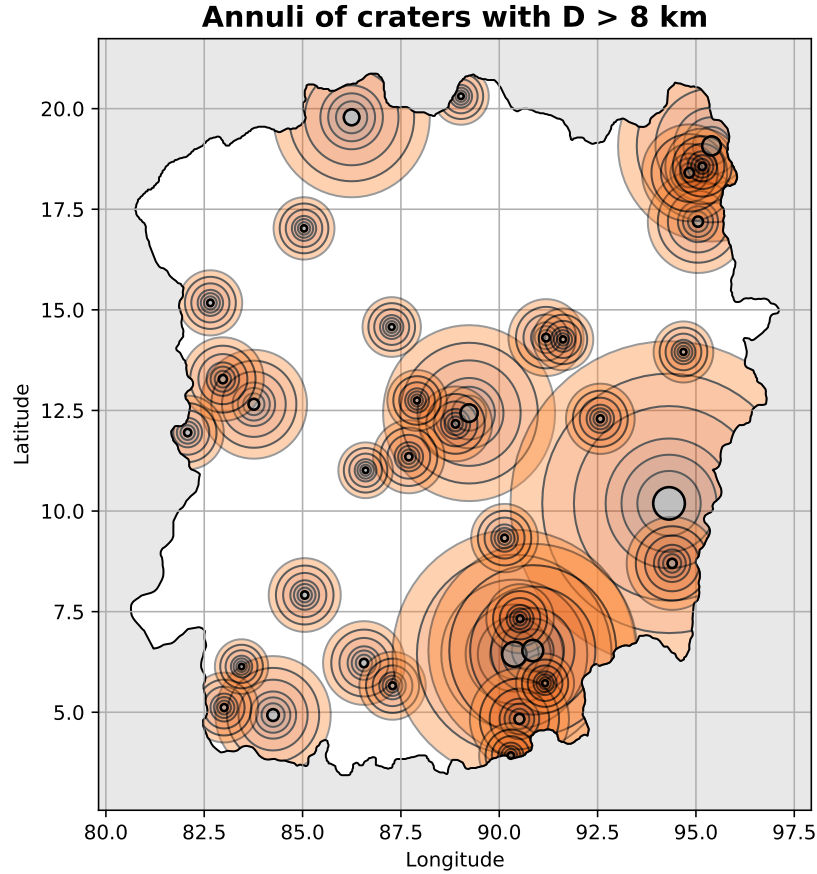


Figure 9.6: Concentric annuli used in the analysis of the areas around large craters. Craters shown to accurate size with the innermost grey circles. Annuli extend out to 5 crater diameters, or 10 radii, away from the centre. Many craters' annuli overlap, which is unavoidable, especially for these larger diameter craters. This shows the difficulty in analysing individual craters, and underlines the advantage of pooling the statistics from many annuli.

However, there is a noticeable divergence from this otherwise monotonic trend for annuli of craters with  $D > 8\text{km}$ . Starting at the crater rim and extending out to about  $1.5D$  from the centre, there is a region with higher densities than for annuli around other diameter ranges (Figure 9.7, lower-right). The additional density in this region peaks at roughly 20% of Isidis' average density for all three small crater bins. A similar analysis is conducted in Robbins et al. [43], who use hand-labelled secondaries from 24 primary impacts spread across Mars, to produce plots of annular densities. Their method targets only those secondaries which are visually identifiable as such, otherwise they would not be marked. Their value for the peak of secondaries as a function of distance was 2.4 crater radii from the rim, which is roughly  $1.5D$  from the centre, whilst our possible secondaries peak at a significantly shorter distance of around  $0.8D$ . Whether the peak

seen in Figure 9.7 is caused by secondaries—or by something else—is unclear, given the discrepancy in the peak distances measured.

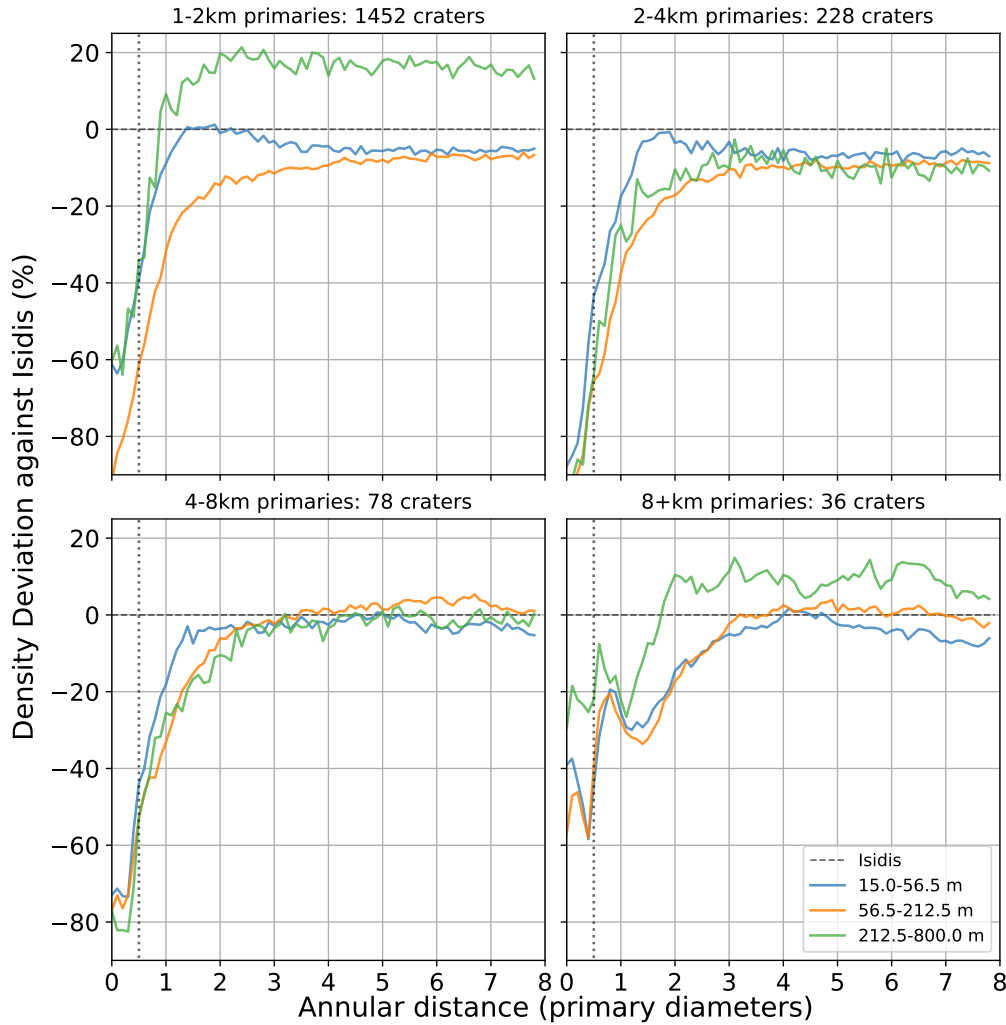


Figure 9.7: Density deviation as a function of annular distance from primary craters, binned to  $D/10$ . The vertical dashed line indicates the large craters' rim, at  $D/2$ . All primary crater size ranges show a decrease in densities at distances within 3–4 diameters. The largest primary craters ( $D > 8\text{km}$ ) are also surrounded by a small region with more craters than is found around others, at roughly  $0.8D$ .

#### 9.4.2 Nearest Similar Neighbours

The previous statistical tests have focused on the density of craters across space and size. Now, a micro-statistical measure is introduced which tells us something about the relationships between individual craters. Nearest neighbour analysis is commonly used to quantify the level of clustering in a distribution. This assumes that all of the instances (in this case individual craters) are equally valid to consider interactions between. However, in the case of craters, it is clear that the distance between a very large crater and its

immediate nearest neighbour (probably a small crater within it's rim) is not a physically important measurement. Instead, we would like to consider the closest *similar* crater.

Therefore, I introduce the NSN metric, defined as the closest crater within a factor of 2 of the diameter of the target crater. This is quite like the measure used in [207], which used nearest neighbour analysis over binned diameter ranges. However, craters are not restricted to nearest neighbours within bins, instead allowing for each crater to pair with any crater within a factor of 2 of its diameter, either half as small or twice as big. This is calculated by successive nearest neighbour searches on an R-Tree [210], with rejection of those neighbours that are not within the correct size window for each crater. Figure 9.8 visualises a small selection of the detections on Isidis, and their NSNs.

Nearest neighbour analysis relies on comparing the distances measured in the distribution, to an idealised random distribution: a Poisson point process, which describes a completely random spatial process, in which there is no dependence between the locations of samples. The expected mean nearest neighbour distance,  $R_{NN}^{expected}$ , for a Poisson point process, with density  $\rho$ , is

$$R_{NN}^{expected} = \frac{1}{2\sqrt{\rho}} \quad (9.1)$$

However, for the NSN,  $\rho$  is a function of diameter, expressed here as  $\rho_{eff}$ , as for each crater there is a different *effective* density, which only takes those craters of a similar size into account. This can be expressed as the total CSFD (normalised by area) between  $D/2$  and  $2D$ ,

$$\rho_{eff}(D) = CSFD(D/2) - CSFD(2D) \quad (9.2)$$

Therefore, the expected NSN distance,  $R_{NSN}^{expected}(D)$ , also becomes a function of diameter,  $D$ :

$$\begin{aligned} R_{NSN}^{expected}(D) &= \frac{1}{2\sqrt{\rho_{eff}(D)}} \\ R_{NSN}^{expected}(D) &= \frac{1}{2\sqrt{CSFD(D/2) - CSFD(2D)}} \end{aligned} \quad (9.3)$$

With Equation 9.3, then, we have a measure of an idealised random point process' mean NSN distance. The ensuing analysis of Mask R-CNN's detections is focused on their deviations from this expectation, in different circumstances. Firstly, though, it is worthwhile to discuss briefly some of the things which may cause the distribution to deviate from a Poisson point process.

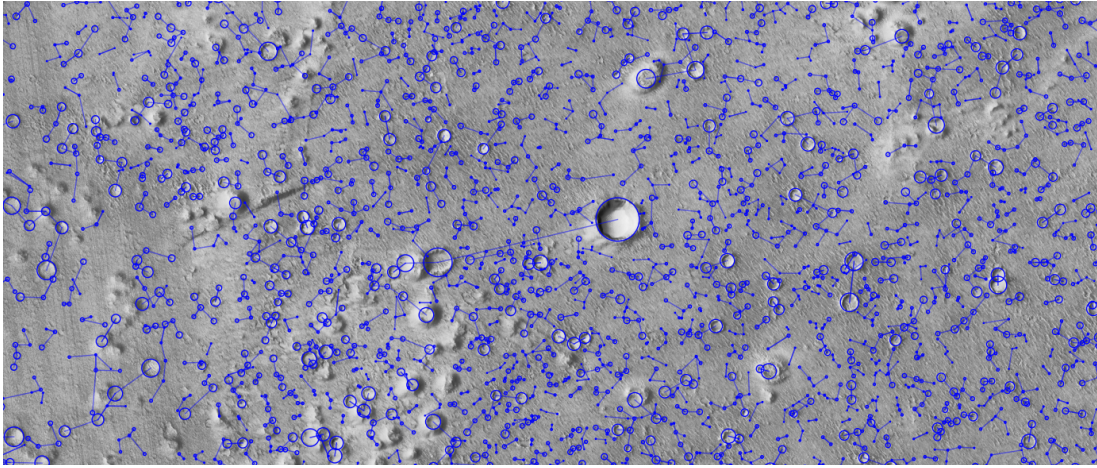


Figure 9.8: Visualisation of a small portion of Mask R-CNN’s detections over Isidis. The lines between detections join NSNs. Larger craters must find partners of a similar diameter, forcing them to ignore small craters close by.

Clearly, craters cannot be infinitesimally close to one another, and are likely unresolvable if the centres are less than about a radius apart from one another. This will lead to a positive pressure on the mean NSN distance, as craters create a small unpopulated halo through obliteration. Additionally, it is thought that secondaries are generally more clustered than their primary cousins. This is because the trajectories of particles emanating from a secondary-producing event will fly in highly correlated angles, leading to a shotgunning effect (this can also happen to primaries, as meteorites break up in the atmosphere, leading to a clustered pattern of craters, however most agree that clustering is more associated with secondaries [195]).

There are also effects which decrease the mean distance whilst not suggesting any clustering in the actual production of craters. For instance, if erosion of craters happens heterogeneously (e.g. through obliteration by a larger crater), this leaves an artificially clustered area around it, with craters more densely packed in the surroundings of the eroded region, than on it. This same affect can also be seen due to limitations in the crater detection model, and noisy images. For instance, several areas of Isidis have much lower small crater densities than others, due to noisy CTX images included in the Murray Labs mosaic. Whilst unavoidable in the scope of this study, this again creates an artificial clustering effect on the craters, especially at lower sizes where the detection rate is more dependent on image quality.

Taken together, then, we see there are a complex ensemble of factors which affect  $R_{NSN}$ . These factors cannot be easily separated from one another. Instead, we look at local variations, with the assumption that many of the effects (declustering through obliteration of very close neighbours, clustering of primaries due to atmospheric break up,

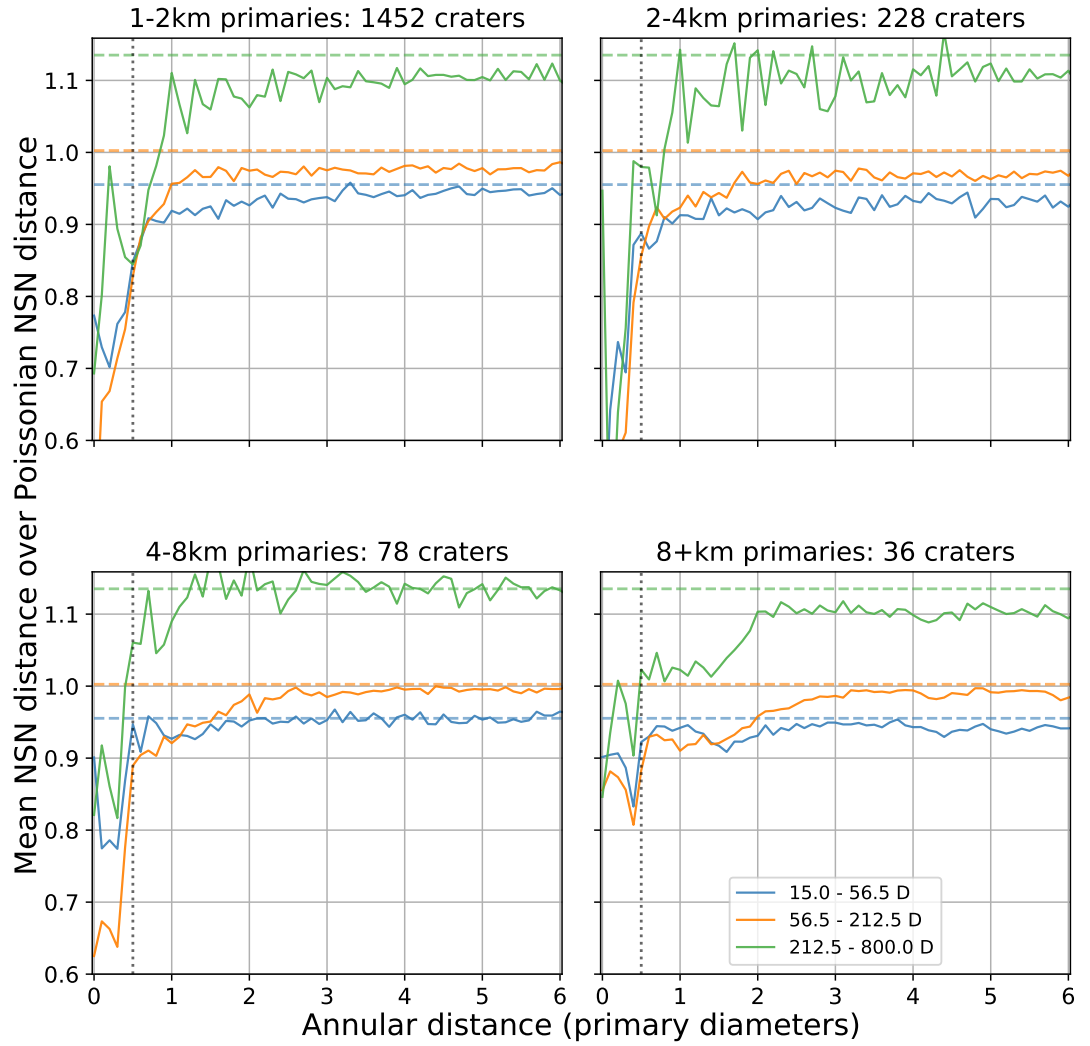


Figure 9.9: Ratio of  $R_{NSN}$  against  $R_{NSN}^{expected}$  for small craters in the vicinity of larger ones. Horizontal dotted lines indicate the average of this ratio for craters within that diameter range over Isidis. Vertical dotted line marks the craters' rims.

and artificial clustering of distributions due to heterogenous erosion or missed detections) are roughly constant across our data. Whilst they can affect small areas very differently, we assume their effect is roughly constant over the length scale of Isidis. By treating these affects as a background, we can focus more on the key parameter we wish to study: the impact of large craters on the distribution of small craters in their vicinity.

To do this, we again turn to plotting the variation across the annuli around large impacts. This time, we find the ratio of the measured  $R_{NSN}$  against the value for a Poisson point process with the same density,  $R_{NSN}^{expected}$ . Figure 9.9 plots this ratio for 3 diameter ranges (the same as those used in Figure 9.7).

Figure 9.9 has some similarities with the trends in density over annular distance seen in 9.7. For all four size ranges of primaries, the maximum area of effect extends to around 2-3 diameters. At distances within this, the  $R_{NSN}$  is smaller than the average over Isidis, showing that small craters close to the primary impacts are *more* clustered. The clustering of the smallest surrounding craters ( $15m < D < 56.5m$ ) seems to be less affected, with  $R_{NSN}$  only dropping by around 0.05–0.1 just outside the primary craters’ rims, compared to drops of around 0.1–0.15 for craters in the range  $56.5m < D < 212.5m$ . The largest bin ( $212.5m < D < 800m$ ) is quite noisy, but also seems to be affected in roughly the same way as the other two size ranges within three primary diameters.

Whereas the ratio of local  $R_{NSN}$  to  $R_{NSN}^{expected}$  increases consistently from the craters’ rim outwards for primary craters of diameters less than 8 km, those bigger than 8 km across (Figure 9.9, bottom right) seem to plateau from 0.5-1.5 diameter annuli, and then begin to increase. This change in pattern is in the same location as the small increase in density seen in the bottom right of Figure 9.7, and suggests that craters in this range are more clustered than they would otherwise be. Given that secondary craters are thought to be strongly clustered, this is further evidence that there is an observable population of secondary craters at distances within around 1.5 diameters, or 3 radii, from the centre of 8km+ impacts.

### 9.4.3 Phase Angle Analysis

Having explored trends in the distances between NSNs, we can also look at how the angles between them vary. Measuring the prevalence of different angles against cardinal direction (see Figure 9.10) hints at processes which affect crater production, such as the interplay between orbital dynamics and atmospheric break-up. If there is a prevailing angle from which meteorites hit the surface, and those meteorites tend to break up in the atmosphere, we would expect some NSNs to show a slight preference for this direction.

As well as the angle against cardinal directions, we can also look at the relative differences between angles. In this case, the angle made by the line spanning two small craters, with a line going to a nearby large crater, the annuli of which the two small craters are within. If pairs of small NSNs tend to be aligned with the radial line from the large crater, then it is more likely that these are part of a ray, which have been observed for several craters on Mars. If, however, the distribution of angles is random, then it suggests that rays are not present, or that they account for an undetectably small number of craters.

Figure 9.11 shows that craters at the smallest sizes are systematically different to those at larger diameters. Those above the median diameter value (74.6 m) show very strong preferences for certain angles, with deviations of  $\pm 2\%$ . This suggests that many impactors are producing craters in such a way that there is a prevailing direction to their nearest neighbours. The most straightforward explanation seems to be that these impactors are arriving at Isidis from certain angles, and then atmospheric break up of the impactor causes lines of craters to form. The two sets of maxima and minima in 9.11 suggests that there are two distinct modes in the direction of impactors, if atmospheric break up is responsible for this pattern. Meanwhile, at smaller sizes ( $D < 43.4m$ ), there is a less prominent trend of only around  $\pm 1\%$  and in a different direction, which suggests that this has a different origin to those  $D > 43.4m$ .

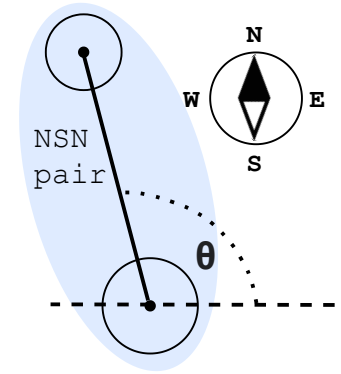


Figure 9.10: Geometry of NSN phase angle. The angle is measured as the difference between the easterly line with the line spanning from a crater to its NSN.

The impactor flux from deep space seems like the first obvious place to look for a preference in the impactors' direction. As [211] shows, the flux of asteroids at a planet's surface varies with zenith angle. The modulation we observe in Figure 9.11, however, is across azimuthal angles, and thus requires there to be a prevailing direction for primaries to arrive from. Impactors are travelling at a mean relative velocity of around 9 km/s [212] when they impact Mars, and the planet's diurnal rotation speed is 241.17 m/s at the equator. Therefore, it seems unreasonable that the diurnal rotation would be responsible for any noticeable directionality in the impactors. Moreover, whilst one of the lobes seen in Figure 9.11 is somewhat equatorial in direction, there is another sharp lobe at around  $115^\circ$  from the equatorial line, which does not correlate with diurnal rotation.

Another possible explanation for the peaks observed in NSN angular distribution, is that the cones which form curvilinear ridges over Isidis (also known as "thumbprint terrain") are being falsely detected as craters, and tend to align themselves along certain



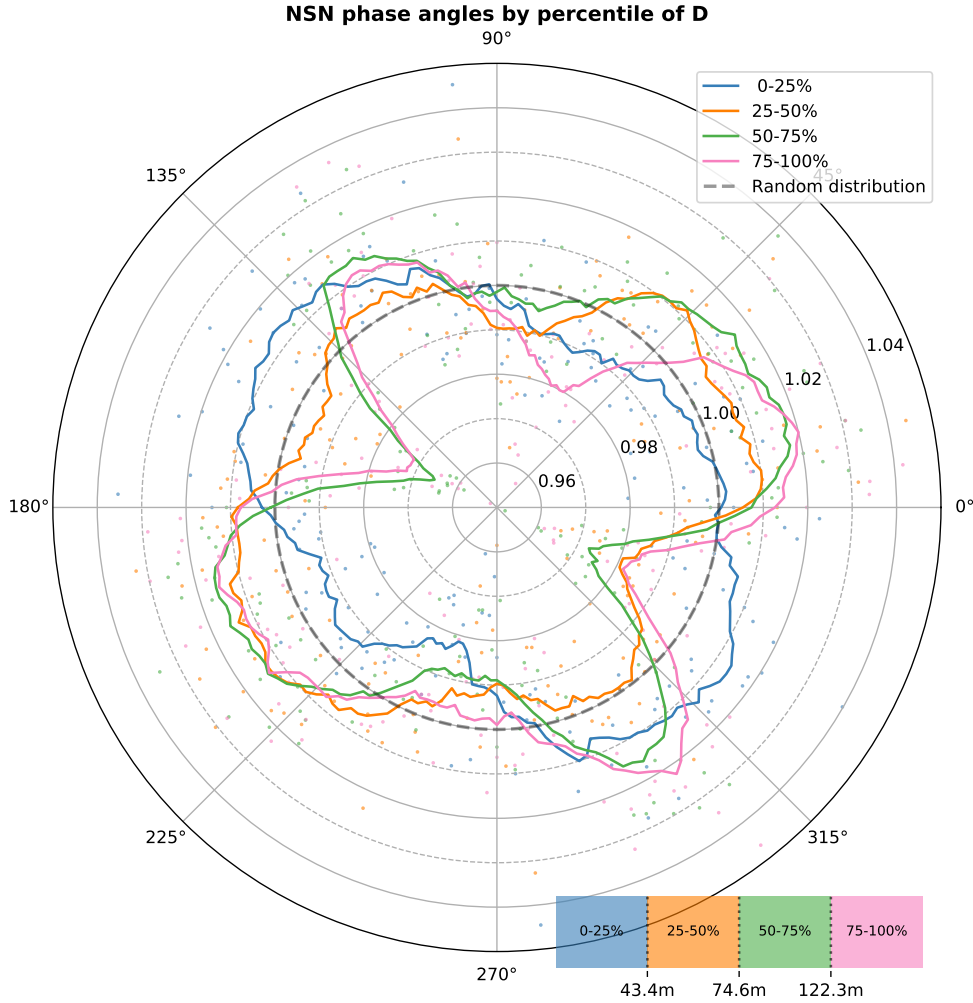


Figure 9.11: Relative prevalence of NSN phase angles, for different diameter ranges, split into quartiles. Angles are binned into  $2^\circ$  (shown by the points), and  $0^\circ$  is an easterly direction. The lines represent smoothed averages over a  $\pm 15^\circ$  range. As one would expect, the distribution is rotationally symmetrical on  $180^\circ$ , because NSNs often pair together in both directions, making roughly equal numbers of lines in opposite directions. All three quartiles above 25% have similar patterns, with two lobes at around  $0-45^\circ$  and  $100-125^\circ$ , although the 25–50% quartile has a less pronounced structure. Meanwhile, the lowest quartile follows a different distribution, with one smoother lobe at around  $100-170^\circ$ .

directions. This explanation is inconsistent with our results for three reasons, though. First, [143] note that the cones over Isidis are not distributed over the whole basin, with very few in the South East quadrant of Isidis (see Figure 1 from [143]). As we will see later, the distribution we find holds over all 4 quadrants of Isidis, unlike the distribution of thumbprint terrain. Second, whilst the direction of the curvilinear ridges are correlated locally, over the whole of Isidis there are no clear prevailing directions on which they lie. Third, the different peaks we find in small and larger detections (Figure 9.11) are not easily explained if they are caused by thumbprint terrain, given that there aren't distinct populations of cones at different sizes, aligned along different directions.



If we discount the effects of orbital mechanics, and the distribution of the cones over Isidis, then, we are left with another hypothesis. A significant portion of the detected craters could have been created in isolated events which arrived at Isidis from specific directions: a massive number of secondary craters from a singular moments in Mars' history, when large impacts produced huge showers of ejecta.

These craters are very close to one another, often becoming NSNs, so the particles which created them must have entered Mars' atmosphere in formation, requiring extremely similar initial ballistic velocities and angles at ejection from primary. This could be due to highly correlated beams of ejecta, initially in liquid or vapour phase during ejection, that rapidly cool whilst in flight, coagulating to form groups of solid secondary impactors all travelling in very close formation [213, 214]. Evidence for this has been found on Phobos: the straight grooves running over its surface are thought to be caused by streams of vaporised ejecta from impacts on Mars [214, 215].

Upon re-entry into the Martian atmosphere, particles in the clusters would have experienced different amounts of atmospheric drag, due to their surface areas and masses. This atmospheric drag would act to further separate the trajectories (along the flight's arc, but with very little impact on the relative lateral velocities of the particles), leading to impacts arranged in lines along their incoming azimuthal angle. The three maxima in Figure 9.11 suggest at least three events produced the population seen on Isidis, although there may be more overlapping with each other.

To link the peaks seen in Figure 9.11 with possible events, their angles need to be estimated with some measure. To this end, we define the three peaks by different symbols. The peak seen in the lowest percentile (blue line, Figure 9.11) at around  $140^\circ$ , is defined as  $\alpha$ , and the corresponding peak in a symmetrical position around  $320^\circ$  as  $\alpha'$ . The sharp peak seen at around  $120^\circ$  in percentiles above 25% is defined as  $\beta$ , and its corresponding opposite as  $\beta'$ . Also in those higher percentiles, the peak at around  $30^\circ$  and  $210^\circ$  as  $\gamma$  and  $\gamma'$ . Now the

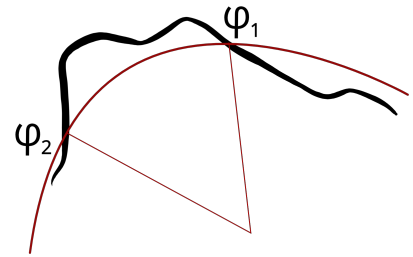


Figure 9.12: Schematic diagram showing how the two angles,  $\varphi_1$  and  $\varphi_2$ , are calculated for a peak in NSN phase angle, defined as the angles at which the smoothed prevalence (black curve) crosses the mean value (red arc). These are averaged to find the peak's direction. The same holds for the corresponding peak found  $180^\circ$  apart.

three peaks are defined separately, we measure the crossing points over the average value on either side of the peak, denoting their angles as  $\alpha_1$  and  $\alpha_2$  and so on (Figure 9.12). The mid-point of these two angles,  $\bar{\alpha}$ , is then used as the approximate direction of the peak. Table 9.1 gives the angles for each peak's crossing points, and their averages.

Angle	0-25% D	25-50% D	50-75% D	75-100% D
$\alpha_1$	101	-	-	-
$\alpha_2$	184	-	-	-
$\bar{\alpha}$	<b>142.5</b>	-	-	-
$\alpha'_1 - 180^\circ$	100	-	-	-
$\alpha'_2 - 180^\circ$	178	-	-	-
$\bar{\alpha}' - 180^\circ$	<b>139</b>	-	-	-
$(\bar{\alpha} + \bar{\alpha}')/2$	<b>140.75</b>	-	-	-
$\beta_1$	-	104	101	98
$\beta_2$	-	140	135	132
$\bar{\beta}$	-	<b>122</b>	<b>118</b>	<b>115</b>
$\beta'_1 - 180^\circ$	-	-	103	97
$\beta'_2 - 180^\circ$	-	-	135	140
$\bar{\beta}' - 180^\circ$	-	-	<b>119</b>	<b>118.5</b>
$(\bar{\beta} + \bar{\beta}')/2$	-	<b>122</b>	<b>118.5</b>	<b>116.75</b>
$\gamma_1$	-	-3	-4	-9
$\gamma_2$	-	63	60	40
$\bar{\gamma}$	-	<b>30</b>	<b>30</b>	<b>15.5</b>
$\gamma'_1 - 180^\circ$	-	-9	0	-4
$\gamma'_2 - 180^\circ$	-	64	58	59
$\bar{\gamma}' - 180^\circ$	-	<b>27.5</b>	<b>29</b>	<b>27.5</b>
$(\bar{\gamma} + \bar{\gamma}')/2$	-	<b>28.75</b>	<b>29.5</b>	<b>21.5</b>

Table 9.1: Crossing angles for the three observed peaks,  $\alpha$ ,  $\beta$  and  $\gamma$ . There is fairly good agreement in direction between the different percentiles where they share peaks, with a range of 6-7° over the mean  $\beta$  and  $\gamma$  angles. There is also good agreement between each peak and its corresponding opposite, except for  $\gamma$  in the 75-100% diameter percentile, which disagrees by 12°.  $\beta'$  was not calculable for the 25-50% percentile as it did not peak strongly enough.

This method can be extended, to ascertain whether the direction of the peaks change over Isidis, which may hint at their point of origin. The detections over Isidis are broken into 4 roughly equal quadrants, on either side of 88.75°E and 12.5°N. Then, we plot the NSN phase angle prevalence for each of these quadrants, both for  $D < 50m$  (Figure 9.13), and for  $D > 50m$  (Figure 9.14).

As expected, the  $\alpha$  peak is consistently visible over all 4 quadrants at  $D < 50m$  (Figure 9.13), however given that the body of craters has now been split into 4, it is significantly more noisy than when looking at the whole of Isidis. Therefore, it has been smoothed by  $\pm 30^\circ$  rather than the  $\pm 15^\circ$  used elsewhere. In craters of  $D > 50m$ , the  $\beta$  and  $\gamma$  peaks are visible in all 4 quadrants (Figure 9.14).

With these plots, we can estimate angles for  $\alpha$ ,  $\beta$  and  $\gamma$  for each of the 4 quadrants (Table 9.2). Using spherical trigonometry, and our values of the latitude, longitude and phase angle at each of the quadrants' centroids (simply defined as the centre of gravity of the polygon of the quadrant), we can then calculate the ground track for a circular orbit of Mars which crosses at this point. The path of this circular orbit is a rough approximation

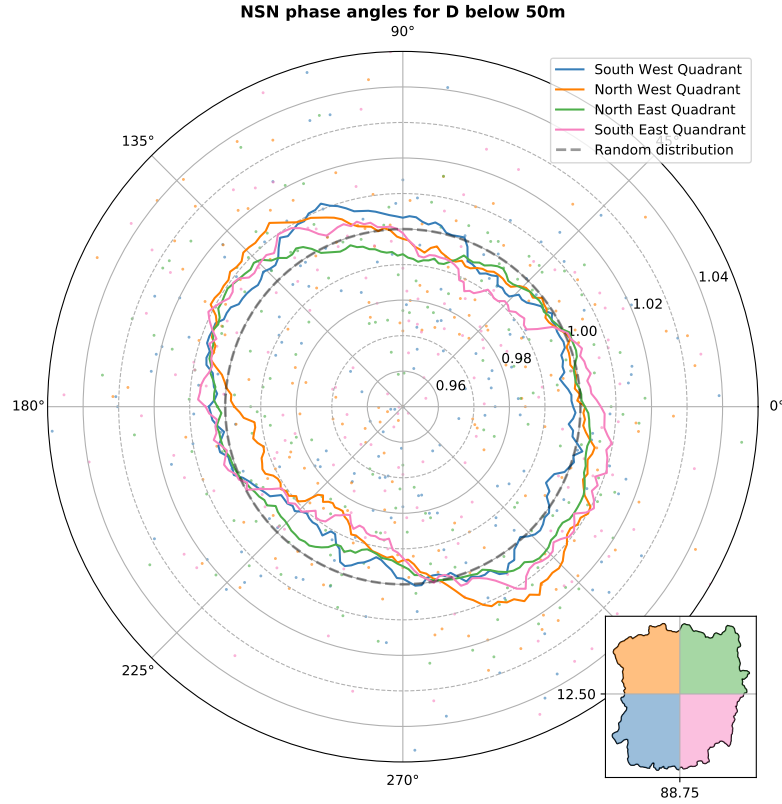


Figure 9.13: Prevalence of NSN phase angles for small craters in the four quadrants of Isidis Planitia. At this size range,  $\alpha$  is consistently visible, with no sign of  $\beta$  or  $\gamma$ . Given that there are many fewer craters per quadrant, the data is averaged across  $\pm 30^\circ$  rather than the  $\pm 15^\circ$  used elsewhere. All quadrants show a similar trend, which is parameterised in Table 9.2.

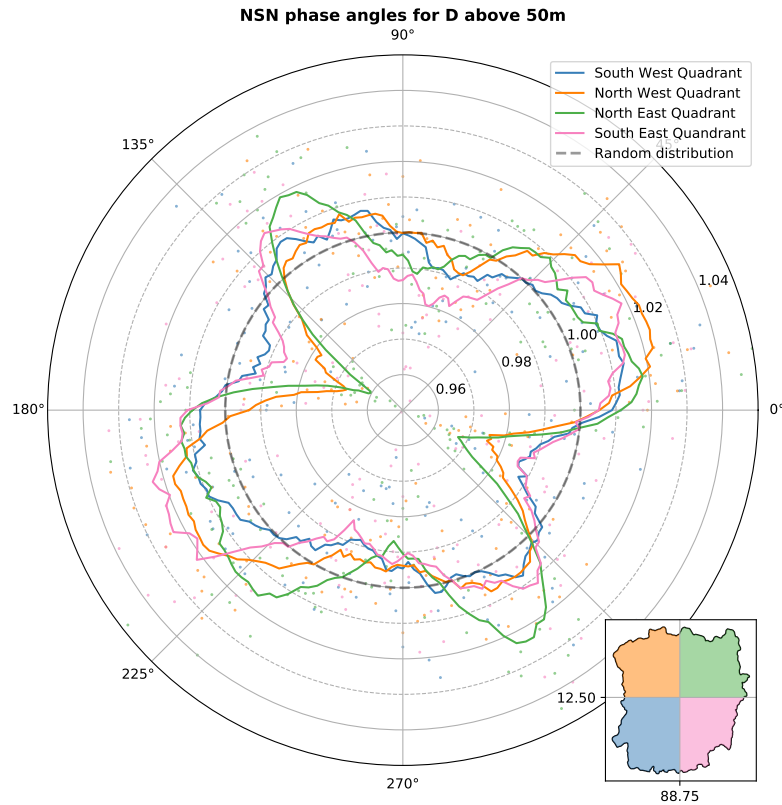


Figure 9.14: NSN phase angles for larger ( $D > 50m$ ) craters in the four quadrants of Isidis Planitia. The  $\beta$  and  $\gamma$  peaks are easily resolved in all four quadrants, with some small differences between them, which are measured in Table 9.2.

of the areas on Mars where a primary impact could have created secondaries which arrive at Isidis from the measured direction.

Recall, from spherical trigonometry, the spherical triangle sine and cosine rules. For a triangle with side lengths  $a$ ,  $b$ ,  $c$  and angles  $A$ ,  $B$ ,  $C$ ,

$$\frac{\sin A}{\sin a} = \frac{\sin B}{\sin b} = \frac{\sin C}{\sin c} \quad (9.4)$$

$$\cos a = \cos b \cos c + \sin b \sin c \cos A \quad (9.5)$$

The inclination of the orbit,  $i$ , and the longitude of the ascending node,  $\Omega$ , are calculated with respect to the longitude,  $lon$ , latitude,  $lat$ , and phase angle,  $\varphi$ , at the centroid (all in units of radians, for ease of calculation), using Equations 9.4 and 9.5,

$$i = \arccos(\cos(lat) \cos(\varphi)) \quad (9.6)$$

$$\Omega = \begin{cases} lon - \arccos\left(\frac{\cos\left(\arcsin\left(\frac{\sin(lat)}{\sin(i)}\right)\right)}{\cos(lat)}\right) & \text{if } \varphi \leq \frac{\pi}{2}, \\ lon + \arccos\left(\frac{\cos\left(\arcsin\left(\frac{\sin(lat)}{\sin(i)}\right)\right)}{\cos(lat)}\right) & \text{if } \varphi > \frac{\pi}{2} \end{cases} \quad (9.7)$$

These orbital tracks indicate a rough direction from which streams of secondary craters could have emanated, to produce these populations. Figure 9.15 visualises these orbits over the globe. None of the three peaks show much evidence of their paths focusing on a single point. Given the uncertainties in the method for measuring the peaks' angles, this simply proves that—if any of the three peaks do originate from a specific location—the impact they came from is far enough away from Isidis that the precision of angular measurements needed for triangulation is not available.

#### 9.4.4 Relative Angle Analysis

In the previous section, the angles between NSN pairs were used to uncover patterns in the directionality and distribution of craters. Now, rather than measuring angles against compass directions, this next experiment looks at how these angles relate to large craters in the vicinity of the NSN pairs.

Angle	NW	NE	SW	SE
$\alpha_1$	95	120	75	95
$\alpha_2$	170	200	200	200
$\bar{\alpha}$	<b>132.5</b>	<b>160</b>	<b>137.5</b>	<b>147.5</b>
$\alpha'_1 - 180^\circ$	100	110	-	105
$\alpha'_2 - 180^\circ$	180	190	-	205
$\bar{\alpha}' - 180^\circ$	<b>140</b>	<b>150</b>	-	<b>155</b>
$(\bar{\alpha} + \bar{\alpha}')/2$	<b>136.25</b>	<b>155</b>	<b>137.5</b>	<b>151.25</b>
$\beta_1$	90	101	93	106
$\beta_2$	131	132	140	140
$\bar{\beta}$	<b>110.5</b>	<b>116.5</b>	<b>116.5</b>	<b>123</b>
$\beta'_1 - 180^\circ$	105	103	100	106
$\beta'_2 - 180^\circ$	135	135	140	138
$\bar{\beta}' - 180^\circ$	<b>120</b>	<b>119</b>	<b>120</b>	<b>122</b>
$(\bar{\beta} + \bar{\beta}')/2$	<b>115.25</b>	<b>117.75</b>	<b>118.25</b>	<b>122.5</b>
$\gamma_1$	-4	-6	-5	-3
$\gamma_2$	57	59	45	47
$\bar{\gamma}$	<b>26.5</b>	<b>26.5</b>	<b>20</b>	<b>22</b>
$\gamma'_1 - 180^\circ$	4	-5	-8	-7
$\gamma'_2 - 180^\circ$	60	70	44	49
$\bar{\gamma}' - 180^\circ$	<b>32</b>	<b>32.5</b>	<b>18</b>	<b>21</b>
$(\bar{\gamma} + \bar{\gamma}')/2$	<b>29.25</b>	<b>29.5</b>	<b>19</b>	<b>21.5</b>

Table 9.2: Peak crossing angles for craters in each of the 4 quadrants of Isidis. These were derived from Figure 9.13 in the case of  $\alpha$ , and Figure 9.14 for  $\beta$  and  $\gamma$ . Bold values are averages from those above them, bold italicised values are the final values for each quadrant, taken as the average of the bold values.

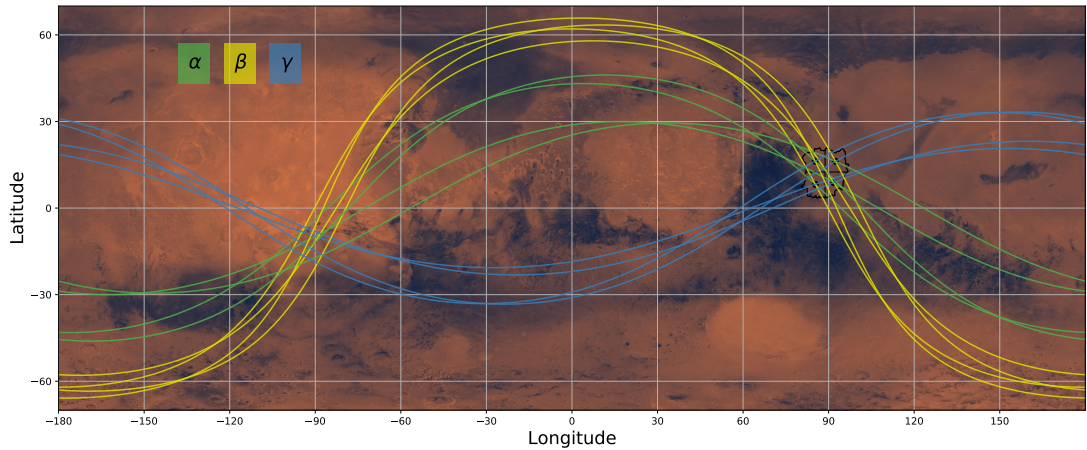


Figure 9.15: Global map of Mars between 70°S–70°N. The directions of each of the three events,  $\alpha$ ,  $\beta$  and  $\gamma$  are extrapolated into circular orbits crossing the centroid of the quadrangles. This estimates the line of possible points on which a primary impact may have created the secondaries responsible for the peak in NSN angles. No obvious focal point can be seen in any of the three sets of lines, because the angular differences are too small over Isidis to resolve the origin points. This figure is overlaid on the Viking orbiter mosaic, courtesy of NASA and the USGS.

Figure 9.16 sets out the geometry of the relative angle between NSN pairs. We define a *radial* NSN pair to be one which goes in the direction of a radial line from the large primary ( $\theta \leq 45^\circ$ ), and an *axial* pair as one which is perpendicular to a radial line ( $\theta > 45^\circ$ ).

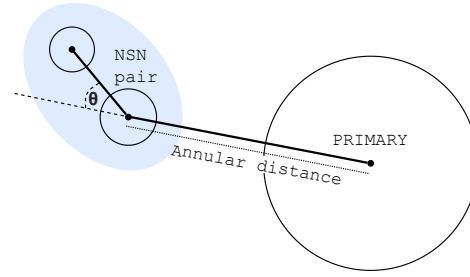


Figure 9.16: Geometry of relative NSN angle,  $\theta$ . The angle is always taken as the smallest positive angle between the two bisecting lines, making it lie between  $0^\circ$  and  $90^\circ$ . A *radial* pair is one where  $\theta \leq 45^\circ$ , and *axial* where  $\theta > 45^\circ$ .

The rationale for studying the angles in this way is the finding by several studies (e.g. [199, 200]) that impact craters can create very large populations of secondary craters arranged in *rays*, emanating from the impact

location. These rays have been observed to extend very far from the primary impact, in the case of Zunil crater, up to 1600 km away of the 10 km diameter crater [199].

These studies identified secondary crater rays visually. Instead, this work considers the entire population of crater detections, but can look for signs of rays within that. If a significant number of craters are aligned along rays centering on primary impacts, there should be an uptick in radial vs. axial NSN angles, indicating some of the NSN pairs are aligned in rays.

The ratio of radial to axial angles as a function of annular distance is plotted in Figure 9.17. There is no noticeable trend at any primary diameter scale or annular distance, suggesting that if secondaries are in rayed formations, they are not a large enough population to affect this metric's value, or are far enough away not to be captured by this analysis. Zunil crater was found to produce rays very far from the crater (up to 160 times the diameter), which we do not have the ability to test for, given that our detections are bounded by Isidis Planitia borders. Given that Sections 9.4.1 and 9.4.2 pointed to the main population of secondaries existing at around  $0.8D$ , it seems that—at least for those secondaries—rays are not evident. It may be that secondaries further out than this exist, but are not resolvable in the annular variations with respect to any of the three variables measured: density, NSN distance, or relative NSN phase angles.

## 9.5 Interim Conclusions

This study sought to better our understanding of small crater populations. This was done in a novel way, by detecting a massive number of craters automatically across a large region, and then employing statistical techniques to uncover patterns in the way those craters are distributed. Isidis was chosen as the study site for its flat topography

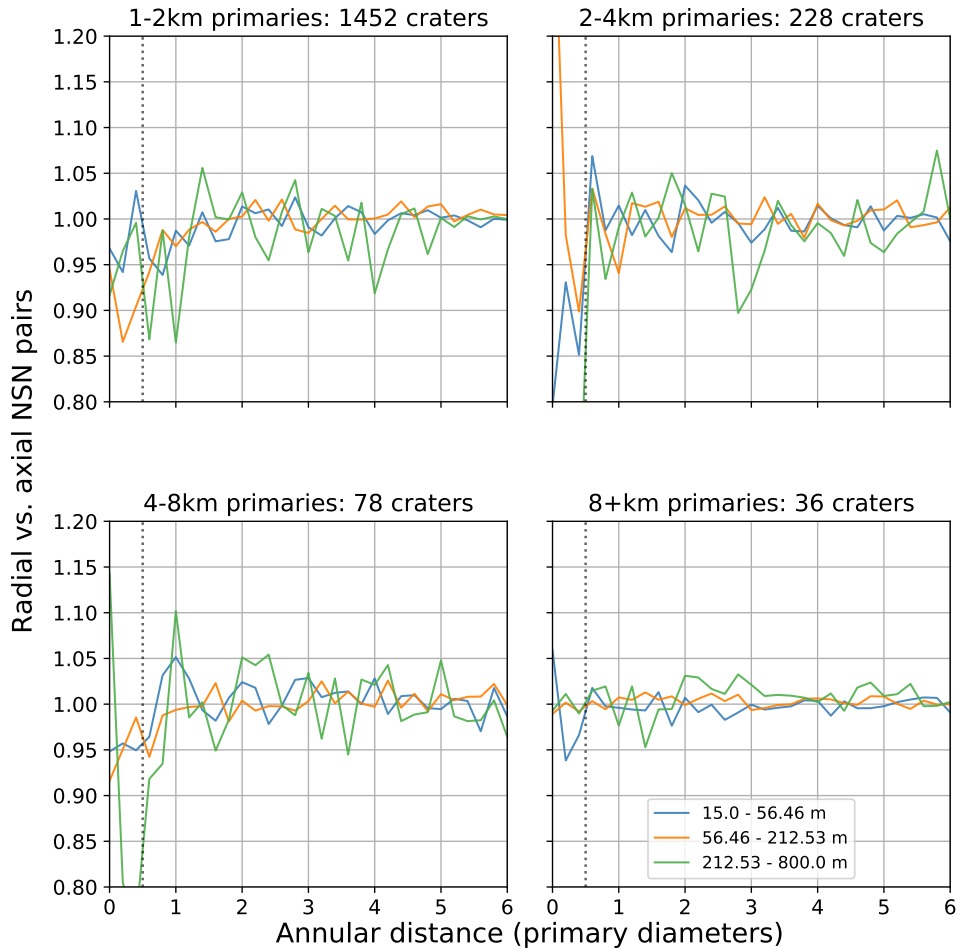


Figure 9.17: Ratio of number of radial to axial NSN pairs, as a function of annular distance. This statistic has no trend, fluctuating around a value of 1 for all three diameter bins, for all annuli. This means there is no evidence of rayed cratering.

and geological consistency, which made it easier for the crater detection model, and made it more straightforward to interpret the results.

Two models were used to detect craters over Isidis, having both shown roughly equal performance in the tests of Chapter 8. Further context-specific considerations, including the nature of their CSFDs, and the spatial distribution of results, suggested that Mask R-CNN's detections were preferable to YOLOv5's. This was because Mask R-CNN showed higher detection rates at very small diameters, and less difference in density between areas of the CTX mosaic with low and high levels of noise.

Obliteration was found to be the most significant impact of large craters ( $D > 1km$ ) on smaller crater populations within their immediate surroundings. It was shown that small crater populations are up to 50% less dense close to the large crater rim, with a noticeable diminishing of density out to around 4 diameters. This finding tells us that counting of small craters within 4 diameters, or 8 radii, of a  $D > 1km$  crater should either be corrected for, or discarded, when trying to date surfaces.

For craters with  $D > 8\text{km}$ , some evidence of secondary cratering was found, however these were closer than previous studies have measured (with densities peaking within a diameter of the crater centre) and did not appear to have a steeper CSFD than the primaries in their surroundings, with craters at  $212.5\text{m} < D < 800\text{m}$  showing roughly the same increase in density as  $D < 56.5\text{m}$ . Despite the noticeable bump in densities that may be due to secondaries, it cannot be claimed—based on these experiments—that secondaries close to their parent impacts dominate at any size range studied ( $D > 15\text{m}$ ).

Meanwhile, no evidence was found that these secondary craters were forming rays close to primary impacts. This may be because rays do not form within the distances measured from the large primaries. If both these results and those of others are correct, this points to the existence of two distinct secondary crater populations, with distant secondaries forming in rays, and ones close to the primary impact not doing so.

The hypothesis of distant rays is strengthened by the tentative evidence for large populations of craters arriving at Isidis from specific directions. An experiment attempting to triangulate the origin of these secondaries was not successful, given the uncertainty in the measurements of the angles. They did confirm that if these are of secondary origin, then their parent primary is far from Isidis, which is an important result in itself. Future work could take measurements from several sites around Mars, leading to a more successful triangulation. If primary impacts can be found which are responsible for this trend in NSN angle, then this would be strong evidence that a huge proportion of craters  $D < 1\text{km}$  are of secondary origin, and are far from their parent primary.

Whilst this method shows potential for pinpointing primaries which are responsible for large populations of secondary craters in Mars’ recent history, it will be highly non-trivial to calculate the percentage of total craters that are of distant secondary origin using this approach. This is because the NSN pairings are highly noisy, as many of those distant secondary craters will find a pairing with uncorrelated primary impacts, weakening the signal. In addition, it is not clear how well-aligned the hypothetical streams of ejecta emanating from the primary are, or how far apart the particles tend to impact the surface. In the future, stochastic modelling of this process—where physical parameters are tuned to fit real-world NSN angle and distance distributions—would allow us to estimate the number of craters which are distant secondaries.

This study is the first to apply deep learning to crater detection at scale with new scientific outcomes. Many of the statistical techniques which arose in the analysis of the detections were also novel, such as the definition of NSN, and the ensuing interpretation of NSN phase angles. The methods of this work can be easily extended to other areas on Mars, or indeed other bodies such as the Moon.



NSN analysis of large, automatically detected crater surveys could add significant detail to our knowledge of the recent history of planetary bodies. If distinct source primaries can be successfully triangulated using the NSN phase angles, then comparisons of where their resulting secondaries are and are not present in different geological units would resolve the chronological order of different primary impacts and resurfacing events, offering a narrative, stepwise view of a planet's recent history.

When scaling up the automated detection of craters from the experiments in Chapter 8 to a real-world application such as this, the specificities and peculiarities of the study site had some impact on the model's behaviour. One prominent example of this are the false positive detections over the cones in parts of Isidis Planitia. This presents an avenue for further work, in order to better ascertain the prevalence and effect of these errors, and to use the cone features as hard negatives during training, to further improve real-world performance of the CDA.

Another facet of real-world application which this chapter showed the importance of is the relationship between crater diameter and detection performance (both in terms of precision and recall). Given the noticeable decrease in detections against the expected inverse power law for craters below 200 m in diameter, some formulation of the performance statistics against diameter would allow users of the data (and the CDA itself) to better interpret and use the crater surveys. These performance estimates could also be tied to image quality, given that—as demonstrated in Figure 9.5—different products have very different densities of detections.

Adaptation and extension of crater detection models to also estimate properties of the craters (e.g. depth-to-diameter ratio, ellipticity, freshness) would provide a massively rich dataset for more targeted statistical tests. The concept of similarity for NSNs could be broadened to include these parameters, meaning craters could be paired to other craters with not only similar sizes, but also morphologies and appearances. This may help to disaggregate the total crater population into subcategories that each have their own trends and histories, telling us more and more about the processes which govern their formation and evolution.



## 10 | Discussion

This thesis tackled two distinct problems—cloud masking and crater detection. Chapters 3-5 covered the technical details of cloud masking, whilst Chapters 6-9 dealt with crater detection. In this chapter, I note some of the similarities and differences in the methods and findings of these two branches of work (summarised in Table 10.1), and explore the implications for sensor independence and remote sensing.

	Cloud Masking	Crater Detection
<b>Planet</b>	Earth	Mars
<b>Data type</b>	Multispectral	Panchromatic, visible/IR
<b>Sensors used</b>	6: Carbonite 2, Landsat 7/8, Sentinel-2, PerúSat-1, Sentinel-3 SLSTR	3: HRSC, CTX, THEMIS Daytime IR
<b>Resolution</b>	1 m – 500 m	5 m – 1 km
<b>Task</b>	Segmentation	Object detection
<b>Sensor independence from...</b>	SEnSel (customised model)	No alteration needed
<b>Best model</b>	DeepLabv3+	Mask R-CNN
<b>Effect of sensor independence on performance</b>	Slight increase in some settings	Large increase
<b>Effect of sensor independence on utility</b>	Large - now possible to develop deep learning cloud masks without training data on target satellite	Large - enabled model to get close to human-level, making scientific application possible
<b>Dataset created</b>	Yes - 513 scenes	Yes - 5000 craters
<b>Dataset annotation strategy</b>	Semi-automated, 2 annotators	Manual, 16 annotators
<b>Novel architecture successful</b>	Partly - CloudFCN worked well, just not as well as DeepLabv3+	No - Model developed from scratch was not suitable
<b>Scientific application pursued</b>	No, but cloud masks have value as a technology for many satellite users, which includes scientists	Yes - crater detections analysed over Isidis Planitia

Table 10.1: Overview of differences in methodology, data, and outcomes between the two projects.

## 10.1 Data

In both problems studied in this thesis, the existing data offered limited possibilities for training and testing sensor independent models, and so new datasets had to be created through manual, or semi-automated, labelling. Even with the crater dataset from the ORBYTS project, and the Sentinel-2 cloud mask catalogue, the number of datasets available to test concepts in sensor independence was limited. With more datasets, more general models could have been trained, and then tested in a wider range of scenarios.

So, whilst model designs such as SEnSeI can open the door to sensor independence, our understanding is still—like most supervised machine learning topics—limited by the available labelled data. Nevertheless, both projects had results which reinforce the assertion that sensor independence can unlock greater value in datasets than they would have when used on their own. Moreover, as deep learning practitioners continue to work on remote sensing tasks, the quantity of labelled data can only grow, making sensor independence more important as time goes on.

Remote sensing research—of Earth and elsewhere—rests on huge amounts of funding and technical work to make satellite manufacture and operation possible. By comparison, labelling data is relatively cheap, and yet it is often left to individual scientists to find the funding and time required to do this. If the institutions and companies which operate satellites want to realise deep learning’s potential, then they should look at ways to increase support of large dataset creation activities.

As shown in Chapter 7, dataset creation can be used to broaden science’s appeal, and to work with the wider community. Public institutions involved in space research should see this as an opportunity to simultaneously support original research, channel funding into community projects, and improve the public perception of science. Especially for projects relating to topics like climate change and environmental collapse, which have vital consequences for everyone. Celebrating the involvement of people outside of academia could help to build trust, understanding and support for the conclusions and policy recommendations that that research arrives at.

For crater detection, sensor independence was possible without modification to models. This is because crater detection does not require multispectral data, meaning all datasets are panchromatic, and craters look visually similar at different scales (or at least, similar enough that a model learns information from large craters which is useful in small crater detection). Meanwhile, pursuing sensor independent cloud masking models required a new architecture, SEnSeI, to be developed.

So, the availability of labelled data, and the structure of the data (in this instance its spectral composition), seem to be key drivers of the work and methods in both projects. In some instances—like in the creation of new datasets—those influences pushed both projects in a similar direction. Conversely, the different spectral structure of datasets led to a divergence in the two projects regarding model architectures like SEnSeI.

## 10.2 Model Selection

In both projects, custom models were developed from scratch. CloudFCN (Chapter 3) showed state-of-the-art performance on Landsat 8, when compared to other non-deep learning methods measured by Foga et al. [106]. However, in later experiments in Chapter 5, it did not match the performance of DeepLabv3+. Additionally, when combined with SEnSeI, training became unstable and so a sensor independent version of it could not be developed.

The initial CDA proposed in Chapter 6 was first prototyped as a classification algorithm, and worked well in this setting when compared to other methods. However, this did not translate into an effective or efficient *detection* algorithm. Eventually, models like YOLOv5 and Mask R-CNN—developed for general object detection tasks and pre-trained on large datasets like COCO and PASCAL VOC—were shown to be more accurate and much more computationally efficient in Chapter 8.

As we know from the wealth of applications that state-of-the-art models succeed in, they are highly adaptable to a wide range of problems, which means that the best models for one task are likely to also be good in other tasks of a similar nature. Given that dozens of models are designed and tested on large computer vision datasets, it is unsurprising that some of these are more effective than those designed by any individual researcher such as myself. Additionally, as was seen in the increased performance from pre-training with an autoencoder in Chapter 5 and 6, initialising a model with pre-trained weights boosts performance. The state-of-the-art CNNs used which out-performed my own efforts—DeepLabv3+, YOLOv5, and Mask R-CNN—all benefitted from extensive pre-training on large datasets with hundreds of thousands of images.

Sensor independence added a very significant boost to performance of both YOLOv5 and Mask R-CNN. Without this boost, the scientific merit of the study on Isidis Planitia would have been weakened due to added noise. Meanwhile, for cloud masking, whilst utility increased, SEnSeI did occasionally lower performance, especially when the model it was compared to had access to a large training set on a single sensor anyway. This is understandable, given that a model with SEnSeI attached to it is already doing something

harder than the model without, even if only trained on data from one sensor, because it is making cloud masks from any subsetting combination of that sensor's bands.

## 10.3 Science and Technology

In Section 1.3, it was emphasised that the kind of value which automating a given task provides can be technological or scientific. It was asserted that sensor independence benefitted both technological and scientific problems. Now, given the results in the two projects, we can see that this assertion holds, at least for the problems tackled here. Not only was there a difference in the kinds of outcome from the two projects, but these outcomes also encouraged a different emphasis in the structure of the experimental set-ups.

Sensor independence unlocked technological value in cloud masking, by showing that satellite operators and instrument teams could create deep learning models which perform better than other methods, even when there is no training data for their satellite. This means deep learning algorithms can be designed pre-launch, and be ready to use as soon as data downlinking begins. The experiments conducted on cloud masking tested performance across multiple satellites, and show through inter-comparison of models that sensor independence worked. Because of the technological emphasis of the project, numerical performance and breadth of testing scenarios was important, as sensor independent cloud masking will be most useful when used widely and in many settings. It is a technological improvement to existing approaches, and as such, adoption by many users and in many situations is how it will ultimately generate value. Whilst technological, it benefits science because those users will improve the cloud masks they use in their work, and as such produce higher quality science.

On the other hand, crater detection took a more focused approach, targeting a specific scientific aim: the detection of craters over a region of the Martian surface, Isidis Planitia. As such, experiments in Chapter 8 used a single test set over Isidis Planitia. This test set was used because it was a sample of precisely the task for which the models were being designed. The experiments could have, for example, included results of models trained on CTX and HRSC data, and tested on the THEMIS Daytime IR data. But this would be scientifically irrelevant—we already know where all the craters are in the THEMIS Daytime IR mosaic. So, whilst it would demonstrate sensor independence, it would not aid in reaching the scientific goal that was the focus of the study.

# 11 | Conclusions

## 11.1 Findings & Contributions

This thesis demonstrates sensor independent deep learning applied to two case studies using optical satellite imagery: cloud masking and crater detection. The results show that sensor independence is worthwhile pursuing, and that the methods proposed to realise it are effective. This section summarises the key findings from the thesis.

CloudFCN [1], a novel convolutional segmentation model, was developed specifically for cloud masking, and performed well when tested on Carbonite-2 and Landsat 8 in Chapter 3. By using a fully convolutional architecture, CloudFCN could be used on images of varying size, and its U-Net style design allowed for fusion between low- and high-level features, from the individual pixel reflectances, to large spatial features. When compared with popular single-pixel methods on Landsat 8, it was shown to significantly outperform them, with a *Quality* of 6.7% higher than the best performing model from [106]. Experiments into the robustness of CloudFCN against sources of noise were a first hint at deep learning’s ability to work in a sensor independent manner, by simulating sensors with different radiometric characteristics.

In Chapter 4, the largest labelled dataset for cloud masking to date (with respect to the number of scenes) was created for Sentinel-2 [3]. This dataset, comprising 513 subscenes, was annotated using a new semi-automated segmentation labelling tool, IRIS. IRIS—developed by John Mrziglod and myself—making painting of scenes much faster, by in-filling areas not yet painted with predictions from a random forest. This tool was designed to be application-agnostic, and its source code is available and easily configurable for a wide range of segmentation labelling tasks <sup>1</sup>.

The Sentinel-2 cloud mask catalogue generated in Chapter 4 was used to train and test various models in Chapter 5. SEnSeI, a novel PINN, made both a single-pixel neural network, and a large CNN, DeepLabv3+ into sensor independent models. In doing so, it was shown that sensor independence could be achieved whilst maintaining or even exceeding the performance of specialised models in some instances. The sensor independent version of DeepLabv3+ was tested on five satellites, 3 of which it had not seen during training, and consistently outputted accurate cloud masks across them all, except for some issues on Landsat 7. This was all achieved with only a small computational

---

<sup>1</sup><https://github.com/ESA-PhiLab/iris>

overhead from SEnSeI (see Appendix A). These results demonstrated that sensor independent models can be applied to a wide range of multispectral satellites, without any labelled training data.

Chapter 6 detailed a CNN which was initially designed to classify small craters in images from HRSC. By pre-training the model using an unsupervised autoencoder set-up, performance was boosted and the classifier achieved an  $F_1$  of 90.4%, 1% higher than the  $F_1$  of the CNN proposed by Cohen et al. [151]. When this classifier was converted into a detection model, performance suffered and it was computationally inefficient. Whilst training with hard negatives boosted performance, and parallel processing techniques made predictions much faster, the model was still not sufficiently accurate or fast to merit its use.

The limitations of the model in Chapter 6 led to two changes in approach. First, in Chapter 7, a dataset of craters was created with CTX imagery [2], which expanded the available labelled data for training and testing models, and widened the number of high resolution sensors that that data came from. This dataset was created as part of the ORBYTS scheme, with 16 students aged 16-18 participating in the project. Six annotators were used for each of the 12 scenes in the dataset, with 15,000 individual crater annotations made, and 5,000 final craters. The multi-annotator approach was validated using a measure of inter-annotator agreement, and it was shown that the use of six volunteer labellers led to annotations in close agreement with an expert's (my own).

The second change of approach—documented in Chapter 8—was to use previously published object detection models, rather than developing one from scratch. Both Mask R-CNN and YOLOv5 demonstrated impressive performance, whilst being far more computationally efficient than the initial model design (39 and 3,800 times faster, respectively). By fine-tuning the model's parameters, and altering the level of data augmentation used, Mask R-CNN's and YOLOv5's mAP@50% was increased slightly. More significantly, using a sensor independent training set-up—in which data from HRSC, CTX, and THEMIS Daytime IR was combined—significantly boosted the mAP@50%.

Mask R-CNN and YOLOv5 were used to detect over 5 million craters over Isidis Planitia in Chapter 9. In total, roughly 5.5 million craters were detected by each model. Both the performance on the test set in Chapter 8 and inspection of the CSFDs, confirmed that Mask R-CNN had produced the detections most suitable for further analysis, and so were focused on in a set of statistical experiments. Several novel geospatial analysis approaches were taken, which would not have been possible to use without automated detection, because of the number of craters used. The NSN was defined to investigate the clustering of craters of similar sizes, and evidence was found for a small but noticeable



population of secondary craters close to large primaries. The angular distribution of NSN pairs was used to show distinct angles of arrival for a large number of impactors, which strengthens the hypothesis that a substantial percentage of small craters are distant secondaries created in isolated, planet-wide events in Mars' recent past.

## 11.2 Future Work

This thesis has focused on applying sensor independence to imagery from optical satellites, with one band (panchromatic) or a few bands (multispectral). Beyond this, though, there are a huge variety of sensors for which sensor independence may be useful. The most obvious of these is hyperspectral imagery. Uptake of hyperspectral imagery is increasing, with the recent launch of several missions like PRISMA and HyperScout-2, and with upcoming missions such as CHIME. By expanding sensor independence to hyperspectral imaging, the density and complexity of spectral information would increase, so that a much wider variety of band combinations could be used, thus allowing for models to be used across an even wider variety of sensors—both hyperspectral and multispectral. A labelled hyperspectral dataset could be used to simulate the bands of all manner of instruments, meaning one could train models from the hyperspectral dataset for any target multispectral spacecraft. Whilst the structure of descriptor vectors used in Chapter 5 enabled the model to distinguish between and use the multispectral bands, hyperspectral imagery may require more detailed descriptor vectors, perhaps encoding the actual transmission curves of each band, rather than just their minima, centres, and maxima.

As well as expanding the spectral capabilities of sensor independent models like SEnSeI, geometrical flexibility would also be desirable. SEnSeI expects all channels to have the same resolution and size. A more general approach would be to design a version of SEnSeI which was given information about each channel's resolution, and which processed it accordingly, such that the output of SEnSeI contained encoded information about the channels' spatial resolutions. Building on this, other characteristics of the sensor and other contextual information could be included in the descriptors given to SEnSeI. For example, noise characteristics, illumination and viewing angles, and point spread functions. This would allow models to make predictions based on this wider body of variables, and may improve performance and the model's ability to adapt to new sensors.

SEnSeI was designed deliberately in a way completely independent of the specific nature of cloud masking. Nor is it specific to segmentation problems. To prove SEnSeI's capabilities as a general-purpose tool for achieving sensor independence, future work

could apply it to other computer vision tasks. This could include classification, object detection, tracking in video or time-series, among others.

In this thesis, cloud masking has been framed as a task which primarily provides value to science *through* technological means, by improving the quality of cloud predictions. However, high resolution cloud masking may also have direct scientific applications. For example, the effect of clouds on the climate system (and vice-versa) represents a significant uncertainty in our understanding of climate change, with the Intergovernmental Panel on Climate Change (IPCC) stating that “Confidence in the representation of processes involving clouds and aerosols remains low”<sup>2</sup>. Clouds at different heights and thicknesses are thought to have varying effects on the climate system [216]. High cirrus clouds push temperatures up by trapping longwave thermal radiation, whilst stratocumuli (and other cloud types found lower in the atmosphere) cause net cooling, by reflecting shortwave solar radiation. Repurposing high-resolution satellite data to obtain empirical measurements of cloud processes with high spatial resolution could constrain cloud-climate interactions from a previously unexplored angle, given that the data currently used in climate modelling is at orders of magnitude coarser resolution than a satellite such as Sentinel-2 provides.

Crater counting is scientifically useful because of its use in age-dating for planetary surfaces. Small craters, however, are less well understood than populations of larger ones, and as such there is a need to better understand their production and erosion processes. As a case-study in automated detection’s scientific usefulness, Isidis Planitia was surveyed for small craters. Whilst this is the largest single crater survey ever conducted, extending it to the rest of the planet would produce an even more valuable resource. The attempted triangulation of large secondary-producing impacts (Figure 9.15) was limited by the size of the study region. By surveying sites across Mars and using the same methods, it may have been clear whether or not the angular trends were indeed the result of large streams of distant secondaries. The creation of a large online database of global crater detections would allow others to use this resource for their own work. If Isidis’ crater density were extrapolated across the planet’s surface then such a global survey would contain roughly 1 billion detections, or 540 million between 30°S–30°N.

This work has focused on cratering on Mars, but many other bodies in the solar system also have substantial populations of craters. For example, the Moon, Mercury, Phobos, and icy moons like Europa, are all currently studied through cratering. Given the success of Mask R-CNN in detecting craters across quite different sensors, both high resolution visible (HRSC and CTX), and lower resolution thermal (THEMIS), it seems that extension to other planetary bodies would be potentially successful too.

---

<sup>2</sup>IPCC AR5 report, p. 56 <https://www.ipcc.ch/report/ar5/syr/>

# Bibliography

- [1] Alistair Francis, Panagiotis Sidiropoulos, and Jan-Peter Muller. Cloudfcn: Accurate and robust cloud detection for satellite imagery with deep learning. *Remote Sensing*, 11(19):2312, 2019.
- [2] Alistair Francis, Jonathan Brown, Thomas Cameron, Reuben Crawford Clarke, Romilly Dodd, Jennifer Hurdle, Matthew Neave, Jasmine Nowakowska, Viran Patel, Arianne Puttock, et al. A multi-annotator survey of sub-km craters on mars. *Data*, 5(3):70, 2020.
- [3] Alistair Francis, John Mrziglod, Panagiotis Sidiropoulos, and Jan-Peter Muller. Sentinel-2 cloud mask catalogue. *Zenodo*, November 2020.
- [4] Martin Sudmanns, Dirk Tiede, Hannah Augustin, and Stefan Lang. Assessing global sentinel-2 coverage dynamics and data availability for operational earth observation (eo) applications using the eo-compass. *International Journal of Digital Earth*, 0(0):1–17, 2019.
- [5] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.
- [6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.
- [7] Scott Mayer McKinney, Marcin Sieniek, Varun Godbole, Jonathan Godwin, Natasha Antropova, Hutan Ashrafian, Trevor Back, Mary Chesus, Greg S Corrado, Ara Darzi, et al. International evaluation of an ai system for breast cancer screening. *Nature*, 577(7788):89–94, 2020.
- [8] S. Dodge and L. Karam. A study and comparison of human and deep learning recognition performance under visual distortions. In *2017 26th International Conference on Computer Communication and Networks (ICCCN)*, pages 1–7, 2017.
- [9] Ethan Fast and Eric Horvitz. Long-term trends in the public perception of artificial intelligence. In *Thirty-first AAAI conference on artificial intelligence*, 2017.

- [10] Jayshree Pandya. The Troubling Trajectory Of Technological Singularity. *Forbes*, Apr 2019.
- [11] Stuart J Robbins and Brian M Hynek. A new global database of mars impact craters  $\geq 1$  km: 1. database creation, properties, and parameters. *Journal of Geophysical Research: Planets*, 117(E5), 2012.
- [12] Roberto Bugiolacchi, Steven Bamford, Paul Tar, Neil Thacker, Ian A Crawford, Katherine H Joy, Peter M Grindrod, and Chris Lintott. The moon zoo citizen science project: Preliminary results for the apollo 17 landing site. *Icarus*, 271:30–48, 2016.
- [13] Manuel A Aguilar, María del Mar Saldana, and Fernando J Aguilar. Assessing geometric accuracy of the orthorectification process from geoeye-1 and worldview-2 panchromatic images. *International Journal of Applied Earth Observation and Geoinformation*, 21:427–435, 2013.
- [14] Jasper Poort, Florian Raudies, Aurel Wannig, Victor AF Lamme, Heiko Neumann, and Pieter R Roelfsema. The role of attention in figure-ground segregation in areas v1 and v4 of the visual cortex. *Neuron*, 75(1):143–156, 2012.
- [15] Ulf H Schnabel, Christophe Bossens, Jeannette AM Lorteije, Matthew W Self, Hans Op de Beeck, and Pieter R Roelfsema. Figure-ground perception in the awake mouse and neuronal activity elicited by figure-ground stimuli in primary visual cortex. *Scientific reports*, 8(1):1–14, 2018.
- [16] Yuji Roh, Geon Heo, and Steven Euijong Whang. A survey on data collection for machine learning: a big data-ai integration perspective. *IEEE Transactions on Knowledge and Data Engineering*, 2019.
- [17] Gerhard Neukum, Boris A Ivanov, and William K Hartmann. Cratering records in the inner solar system in relation to the lunar reference system. *Chronology and evolution of Mars*, pages 55–86, 2001.
- [18] Charlotte Revel, Vincent Lonjou, Sébastien Marcq, Camille Desjardins, Bertrand Fougne, Céline Coppolani-Delle Luche, Nicolas Guilleminot, Anne-Sophie Lacamp, Emmanuel Lourme, Christine Miquel, et al. Sentinel-2a and 2b absolute calibration monitoring. *European Journal of Remote Sensing*, 52(1):122–137, 2019.
- [19] ESA. Multispectral instrument (msi) overview. <https://sentinel.esa.int/web/sentinel/technical-guides/sentinel-2-msi/msi-instrument>. [Online; Accessed on 14/05/2021].

- [20] ESA. Sentinel-2. <https://sentinels.copernicus.eu/web/sentinel/missions/sentinel-2>. [Online; Accessed on 14/05/2021].
- [21] USGS. Landsat 8. <https://www.usgs.gov/core-science-systems/nli/landsat/landsat-8>. [Online; accessed 15/05/2021].
- [22] USGS. Landsat 7. <https://www.usgs.gov/core-science-systems/nli/landsat/landsat-7>. [Online; accessed 19/05/2021].
- [23] ESA. Sentinel-3 olci. <https://sentinel.esa.int/web/sentinel/technical-guides/sentinel-3-olci>. [Online; Accessed on 20/06/2021].
- [24] ESA. Sentinel-3 slstr. <https://sentinel.esa.int/web/sentinel/technical-guides/sentinel-3-slstr>. [Online; Accessed on 20/06/2021].
- [25] ESA. Sentinel-3 olci coverage. <https://sentinel.esa.int/web/sentinel/user-guides/sentinel-3-olci/coverage>. [Online; Accessed on 21/06/2021].
- [26] ESA. Sentinel-3 slstr coverage. <https://sentinel.esa.int/web/sentinel/user-guides/sentinel-3-slstr/coverage>. [Online; Accessed on 21/06/2021].
- [27] NASA. Landsat 8 Overview. <https://landsat.gsfc.nasa.gov/landsat-8/landsat-8-overview>. [Online; accessed 15/05/2021].
- [28] Anders Kose Nervold, Joshua Berk, Jeremy Straub, and David Whalen. A pathway to small satellite market growth. *Advances in Aerospace Science and Technology*, 1(1):14–20, 2016.
- [29] Satellite Imaging Corporation. Skysat-c generation satellite sensors. <https://www.satimagingcorp.com/satellite-sensors/skysat-1/>. [Online; accessed 02/07/2020].
- [30] Ralf Jaumann, Gerhard Neukum, Thomas Behnke, Thomas C Duxbury, Karin Eichentopf, Joachim Flohrer, SV Gasselt, Bernd Giese, Klaus Gwinner, Ernst Hauber, et al. The high-resolution stereo camera (hrsc) experiment on mars express: Instrument aspects and experiment conduct from interplanetary cruise through the nominal mission. *Planetary and Space Science*, 55(7-8):928–952, 2007.
- [31] Christian Heipke, Jürgen Oberst, Jörg Alibertz, Maria Attwenger, Peter Dorninger, Egon Dorrer, Markus Ewe, Stephan Gehrke, Klaus Gwinner, Heiko Hirschmüller,

- et al. Evaluating planetary digital terrain models—the hrsc dtm test. *Planetary and Space Science*, 55(14):2173–2191, 2007.
- [32] P Sidiropoulos and J-P Muller. On the status of orbital high-resolution repeat imaging of mars for the observation of dynamic surface processes. *Planetary and Space Science*, 117:207–222, 2015.
- [33] Richard W Zurek and Suzanne E Smrekar. An overview of the mars reconnaissance orbiter (mro) science mission. *Journal of Geophysical Research: Planets*, 112(E5), 2007.
- [34] Michael C. Malin, James F. Bell III, Bruce A. Cantor, Michael A. Caplinger, Wendy M. Calvin, R. Todd Clancy, Kenneth S. Edgett, Lawrence Edwards, Robert M. Haberle, Philip B. James, Steven W. Lee, Michael A. Ravine, Peter C. Thomas, and Michael J. Wolff. Context camera investigation on board the mars reconnaissance orbiter. *Journal of Geophysical Research: Planets*, 112(E5), 2007.
- [35] NASA. Mars global coverage by context camera on mro - nasa’s mars exploration program. <https://mars.nasa.gov/resources/8334/mars-global-coverage-by-context-camera-on-mro>. [Online; accessed 15/06/2020].
- [36] NASA. Hirise. <https://mars.nasa.gov/mro/mission/instruments/hirise/>. [Online; Accessed on 18/06/2020].
- [37] Alfred S McEwen, Eric M Eliason, James W Bergstrom, Nathan T Bridges, Candice J Hansen, W Alan Delamere, John A Grant, Virginia C Gulick, Kenneth E Herkenhoff, Laszlo Keszthelyi, et al. Mars reconnaissance orbiter’s high resolution imaging science experiment (hirise). *Journal of Geophysical Research: Planets*, 112(E5), 2007.
- [38] The University of Arizona. Hirise: 45,000 mars orbits and counting. <https://uanews.arizona.edu/story/hirise-45000-mars-orbits-and-counting>, Mar 2016. [Online; Accessed on 18/06/2020].
- [39] Lujendra Ojha, Alfred McEwen, Colin Dundas, Shane Byrne, Sarah Mattson, James Wray, Marion Masse, and Ethan Schaefer. Hirise observations of recurring slope lineae (rsl) during southern summer on mars. *Icarus*, 231:365–376, 2014.
- [40] Kathryn M Stack, Nathan R Williams, Fred Calef, Vivian Z Sun, Kenneth H Williford, Kenneth A Farley, Sigurd Eide, David Flannery, Cory Hughes, Samantha R

- Jacob, et al. Photogeologic map of the perseverance rover field site in jezero crater constructed by the mars 2020 science team. *Space science reviews*, 216(8):1–47, 2020.
- [41] RS Saunders, RE Arvidson, GD Badhwar, WV Boynton, PR Christensen, FA Cucinotta, WC Feldman, RG Gibbs, C Kloss, MR Landano, et al. 2001 mars odyssey mission summary. *Space Science Reviews*, 110(1):1–36, 2004.
- [42] CS Edwards, KJ Nowicki, PR Christensen, J Hill, N Gorelick, and K Murray. Mosaicking of global planetary image datasets: 1. techniques and data processing for thermal emission imaging system (themis) multi-spectral data. *Journal of Geophysical Research: Planets*, 116(E10), 2011.
- [43] Stuart J Robbins and Brian M Hynek. Secondary crater fields from 24 large primary craters on mars: Insights into nearby secondary crater production. *Journal of Geophysical Research: Planets*, 116(E10), 2011.
- [44] Andreas Holzinger, Georg Langs, Helmut Denk, Kurt Zatloukal, and Heimo Müller. Causability and explainability of artificial intelligence in medicine. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 9(4):e1312, 2019.
- [45] Warren S McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, 1943.
- [46] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.
- [47] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Icml*, 2010.
- [48] Bing Xu, Naiyan Wang, Tianqi Chen, and Mu Li. Empirical evaluation of rectified activations in convolutional network. *arXiv preprint arXiv:1505.00853*, 2015.
- [49] Chigozie Nwankpa, Winifred Ijomah, Anthony Gachagan, and Stephen Marshall. Activation functions: Comparison of trends in practice and research for deep learning. *arXiv preprint arXiv:1811.03378*, 2018.
- [50] R Duncan Luce. The choice axiom after twenty years. *Journal of mathematical psychology*, 15(3):215–233, 1977.
- [51] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

- [52] Yuxin Wu and Kaiming He. Group normalization. In *Proceedings of the European conference on computer vision (ECCV)*, pages 3–19, 2018.
- [53] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.
- [54] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. The MIT Press, 2016.
- [55] Pieter-Tjerk De Boer, Dirk P Kroese, Shie Mannor, and Reuven Y Rubinstein. A tutorial on the cross-entropy method. *Annals of operations research*, 134(1):19–67, 2005.
- [56] Matthew D. Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 8689 LNCS(PART 1):818–833, 2014.
- [57] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31, 2017.
- [58] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: Convolutional Networks for Biomedical Image Segmentation. may 2015.
- [59] Zhengxin Zhang, Qingjie Liu, and Yunhong Wang. Road extraction by deep residual u-net. *IEEE Geoscience and Remote Sensing Letters*, 15(5):749–753, 2018.
- [60] Hao Dong, Guang Yang, Fangde Liu, Yuanhan Mo, and Yike Guo. Automatic brain tumor detection and segmentation using u-net based fully convolutional networks. In *annual conference on medical image understanding and analysis*, pages 506–517. Springer, 2017.
- [61] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Semantic image segmentation with deep convolutional nets and fully connected crfs. *arXiv preprint arXiv:1412.7062*, 2014.
- [62] Philipp Krähenbühl and Vladlen Koltun. Efficient inference in fully connected crfs with gaussian edge potentials. *Advances in neural information processing systems*, 24:109–117, 2011.



- 
- [63] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE transactions on pattern analysis and machine intelligence*, 40(4):834–848, 2017.
- [64] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking atrous convolution for semantic image segmentation. *arXiv preprint arXiv:1706.05587*, 2017.
- [65] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018.
- [66] François Chollet. Xception: Deep learning with depthwise separable convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1251–1258, 2017.
- [67] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.
- [68] Jasper RR Uijlings, Koen EA Van De Sande, Theo Gevers, and Arnold WM Smeulders. Selective search for object recognition. *International journal of computer vision*, 104(2):154–171, 2013.
- [69] Marti A. Hearst, Susan T Dumais, Edgar Osuna, John Platt, and Bernhard Scholkopf. Support vector machines. *IEEE Intelligent Systems and their applications*, 13(4):18–28, 1998.
- [70] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.
- [71] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.
- [72] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.

- [73] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.
- [74] Joseph Redmon and Ali Farhadi. Yolo9000: better, faster, stronger. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7263–7271, 2017.
- [75] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.
- [76] Charles Ruizhongtai Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. corr abs/1612.00593 (2016). *arXiv preprint arXiv:1612.00593*, 2016.
- [77] Charles R Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *arXiv preprint arXiv:1706.02413*, 2017.
- [78] Nicholas Guttenberg, Nathaniel Virgo, Olaf Witkowski, Hidetoshi Aoki, and Ryota Kanai. Permutation-equivariant neural networks applied to dynamics prediction. *arXiv preprint arXiv:1612.04530*, 2016.
- [79] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2):303–338, 2010.
- [80] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.
- [81] Keinosuke Fukunaga and Larry Hostetler. The estimation of the gradient of a density function, with applications in pattern recognition. *IEEE Transactions on information theory*, 21(1):32–40, 1975.
- [82] Marcel R Ackermann, Johannes Blömer, Daniel Kuntze, and Christian Sohler. Analysis of agglomerative clustering. *Algorithmica*, 69(1):184–215, 2014.
- [83] M. D. King, S. Platnick, W. P. Menzel, S. A. Ackerman, and P.A. Hubanks. Spatial and Temporal Distribution of Clouds Observed by MODIS Onboard the Terra and Aqua Satellites. *IEEE Transactions on Geoscience and Remote Sensing*, 51(7):3826–3852, 2013.
- [84] Jinru Xue and Baofeng Su. Significant remote sensing vegetation indices: A review of developments and applications. *Journal of Sensors*, 2017, 2017.

- [85] Siri Larsen, Hans Koren, and Rune Solberg. Traffic monitoring using very high resolution satellite imagery. *Photogrammetric Engineering & Remote Sensing*, 75:859–869, 07 2009.
- [86] Larry Di Girolamo and Roger Davies. Cloud fraction errors caused by finite resolution measurements. *Journal of Geophysical Research: Atmospheres*, 102(D2):1739–1756, 1997.
- [87] Qing Zhang and Chunxia Xiao. Cloud detection of rgb color aerial photographs by progressive refinement scheme. *IEEE Transactions on Geoscience and Remote Sensing*, 52(11):7264–7275, 2014.
- [88] Adrian Fisher. Cloud and cloud-shadow detection in spot5 hrg imagery with automated morphological feature extraction. *Remote Sensing*, 6(1):776–800, 2014.
- [89] Nima Pahlevan, Jean-Claude Roger, and Ziauddin Ahmad. Revisiting short-wave-infrared (swir) bands for atmospheric correction in coastal waters. *Optics express*, 25(6):6015–6035, 2017.
- [90] Petr Chylek, S Robinson, MK Dubey, MD King, Q Fu, and WB Clodius. Comparison of near-infrared and thermal infrared cloud phase detections. *Journal of Geophysical Research: Atmospheres*, 111(D20), 2006.
- [91] Ismail Gultepe, R Sharman, Paul D Williams, Binbin Zhou, G Ellrod, P Minnis, S Trier, S Griffin, Seong S Yum, B Gharabaghi, et al. A review of high impact weather for aviation meteorology. *Pure and Applied Geophysics*, 176(5):1869–1921, 2019.
- [92] Richard R. Irish, John L. Barker, Samuel Goward, and Terry Arvidson. Characterization of the landsat-7 etm+ automated cloud-cover assessment (acca) algorithm. *Photogrammetric Engineering & Remote Sensing*, 72:1179–1188, 10 2006.
- [93] Zhe Zhu and Curtis E. Woodcock. Object-based cloud and cloud shadow detection in Landsat imagery. *Remote Sensing of Environment*, 118:83–94, 2012.
- [94] S. Bley and H. Deneke. A threshold-based cloud mask for the high-resolution visible channel of Meteosat Second Generation SEVIRI. *Atmospheric Measurement Techniques*, 6(10):2713–2723, 2013.
- [95] Ann Henderson-Sellers. De-fogging cloud determination algorithms. *Nature*, 298(5873):419–420, jul 1982.

- [96] W. B. Rossow, F. Mosher, E. Kinsella, A. Arking, M. Desbois, E. Harrison, P. Minnis, E. Ruprecht, G. Seze, C. Simmer, and E. Smith. ISCCP cloud algorithm intercomparison. *J. Clim. Appl. Meteorol.*, 24:877–903, 1985.
- [97] R. W. Saunders and K. T. Kriebel. An improved method for detecting clear sky and cloudy radiances from AVHRR data. *International Journal of Remote Sensing*, 9(1):123–150, 1988.
- [98] Yi Luo, Alexander P. Trishchenko, and Konstantin V. Khlopenkov. Developing clear-sky, cloud and cloud shadow mask for producing clear-sky composites at 250-meter spatial resolution for the seven MODIS land bands over Canada and North America. *Remote Sensing of Environment*, 112(12):4167–4185, 2008.
- [99] Lazaros Oreopoulos, Michael J. Wilson, and Táms Várnai. Implementation on landsat data of a simple cloud-mask algorithm developed for MODIS land bands. *IEEE Geoscience and Remote Sensing Letters*, 8(4):597–601, 2011.
- [100] 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.
- [101] Bo-Cai Gao and Yoram J. Kaufman. Selection of the 1.375- $\mu\text{m}$  MODIS Channel for Remote Sensing of Cirrus Clouds and Stratospheric Aerosols from Space. *Journal of the Atmospheric Sciences*, 52(23):4231–4237, 1995.
- [102] André Hollstein, Karl Segl, Luis Guanter, Maximilian Brell, and Marta Enesco. Ready-to-Use Methods for the Detection of Clouds, Cirrus, Snow, Shadow, Water and Clear Sky Pixels in Sentinel-2 MSI Images. *Remote Sensing*, 8(8):666, aug 2016.
- [103] Cintia Carbajal Henken, Maurice J. Schmeits, Hartwig Deneke, and Rob A. Roebeling. Using MSG-SEVIRI cloud physical properties and weather radar observations for the detection of Cb/TCu clouds. *Journal of Applied Meteorology and Climatology*, 50(7):1587–1600, 2011.
- [104] M. Joseph Hughes and Daniel J. Hayes. Automated detection of cloud and cloud shadow in single-date Landsat imagery using neural networks and spatial post-processing. *Remote Sensing*, 6(6):4907–4926, 2014.
- [105] Sergii Skakun, Eric F Vermote, Jean-claude Roger, Christopher O Justice, and Jeffrey G Masek. Validation of the LaSRC Cloud Detection Algorithm for Landsat

- 8 Images. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, PP:1–8, 2019.
- [106] Steve Foga, Pat L. Scaramuzza, Song Guo, Zhe Zhu, Ronald D. Dilley, Tim Beckmann, Gail L. Schmidt, John L. Dwyer, M. Joseph Hughes, and Brady Laue. Cloud detection algorithm comparison and validation for operational Landsat data products. *Remote Sensing of Environment*, 194:379–390, 2017.
- [107] Seema Mahajan and Bhavin Fataniya. Cloud detection methodologies: Variants and development—a review. *Complex & Intelligent Systems*, pages 1–11, 2019.
- [108] Maithra Raghu, Jon Kleinberg, and Ben Poole. On the expressive power of deep neural networks. 2016.
- [109] Pengfei Li, Limin Dong, Huachao Xiao, and Mingliang Xu. A cloud image detection method based on SVM vector machine. *Neurocomputing*, 169:34–42, 2015.
- [110] Asmala Ahmad and Shaun Quegan. Cloud Masking for Remotely Sensed Data Using Spectral and Principal Components Analysis. *Technology & Applied Science Research*, 2(3):221–225, 2012.
- [111] Jonathan Lee, Ronald C. Weger, Salles K. Sengupta, and Ronald M. Welch. A Neural Network Approach to Cloud Classification. *IEEE Transactions on Geoscience and Remote Sensing*, 28(5):846–855, 1990.
- [112] Stephan R. Yhann and James J. Simpson. Application of Neural Networks to AVHRR Cloud Segmentation. *IEEE Transactions on Geoscience and Remote Sensing*, 33(3):590–604, 1995.
- [113] Bin Tian, Mukhtiar A Shaikh, Mahmood R Azimi-Sadjadi, Thomas H. Vonder Haar, and Donald L Reinke. A study of cloud classification with neural networks using spectral and textural features. *IEEE Transactions on Neural Networks*, 10(1):138–151, 1999.
- [114] Yuri Shendryk, Yannik Rist, Catherine Ticehurst, and Peter Thorburn. Deep learning for multi-modal classification of cloud, shadow and land cover scenes in planetscope and sentinel-2 imagery. *ISPRS Journal of Photogrammetry and Remote Sensing*, 157:124–136, 2019.
- [115] Radhakrishna Achanta, Appu Shaji, Kevin Smith, Aurelien Lucchi, Pascal Fua, and Sabine Süsstrunk. SLIC superpixels compared to state-of-the-art superpixel methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(11):2274–2281, 2012.

- [116] Mengyun Shi, Fengying Xie, Yue Zi, and Jihao Yin. Cloud detection of remote sensing images by deep learning. *International Geoscience and Remote Sensing Symposium (IGARSS)*, 2016-Novem:701–704, 2016.
- [117] Fengying Xie, Mengyun Shi, Jihao Yin, and Danpei Zhao. Multilevel Cloud Detection in Remote Sensing Images Based on Deep Learning. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 10(8):3631–3640, 2017.
- [118] Yue Zi, Fengying Xie, and Zhiguo Jiang. A cloud detection method for Landsat 8 images based on PCANet. *Remote Sensing*, 10(6):1–21, 2018.
- [119] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.
- [120] 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.
- [121] Sorour Mohajerani, Thomas A. Krammer, and Parvaneh Saeedi. Cloud Detection Algorithm for Remote Sensing Images Using Fully Convolutional Neural Networks. oct 2018.
- [122] Zhiwei Li, Huanfeng Shen, Yancong Wei, Qing Cheng, and Qiangqiang Yuan. Cloud detection by fusing multi-scale convolutional features. *Proc. ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Science*, pages 149–152, 2018.
- [123] Lei Ma, Yu Liu, Xueliang Zhang, Yuanxin Ye, Gaofei Yin, and Brian Alan Johnson. Deep learning in remote sensing applications: A meta-analysis and review. *ISPRS journal of photogrammetry and remote sensing*, 152:166–177, 2019.
- [124] Zhiwei Li, Huanfeng Shen, Qing Cheng, Yuhao Liu, Shucheng You, and Zongyi He. Deep learning based cloud detection for medium and high resolution remote sensing images of different sensors. *ISPRS Journal of Photogrammetry and Remote Sensing*, 150:197–212, 2019.
- [125] Marc Wieland, Yu Li, and Sandro Martinis. Multi-sensor cloud and cloud shadow segmentation with a convolutional neural network. *Remote Sensing of Environment*, 230:111203, 2019.

- [126] Joseph M. Hughes. L8 sparcs cloud validation masks. <https://www.usgs.gov/core-science-systems/nli/landsat/spatial-procedures-automated-removal-cloud-and-shadow-sparcs>, 2016.
- [127] Gerhard Neukum and DU Wise. Mars- a standard crater curve and possible new time scale. *Science*, 194(4272):1381–1387, 1976.
- [128] Robert G Strom, Renu Malhotra, Takashi Ito, Fumi Yoshida, and David A Kring. The origin of planetary impactors in the inner solar system. *Science*, 309(5742):1847–1850, 2005.
- [129] Boris A Ivanov. Mars/moon cratering rate ratio estimates. *Space Science Reviews*, 96(1):87–104, 2001.
- [130] Caleb I Fassett and James W Head III. The timing of martian valley network activity: Constraints from buffered crater counting. *Icarus*, 195(1):61–89, 2008.
- [131] Paul B Niles, David C Catling, Gilles Berger, Eric Chassefière, Bethany L Ehlmann, Joseph R Michalski, Richard Morris, Steven W Ruff, and Brad Sutter. Geochemistry of carbonates on mars: implications for climate history and nature of aqueous environments. *Space Science Reviews*, 174(1):301–328, 2013.
- [132] Jean-Pierre Bibring, Yves Langevin, John F Mustard, François Poulet, Raymond Arvidson, Aline Gendrin, Brigitte Gondet, Nicolas Mangold, P Pinet, F Forget, et al. Global mineralogical and aqueous mars history derived from omega/mars express data. *science*, 312(5772):400–404, 2006.
- [133] Gerhard Neukum and Konrad Hiller. Martian ages. *Journal of Geophysical Research: Solid Earth*, 86(B4):3097–3121, 1981.
- [134] William K Hartmann and Daniel C Berman. Elysium planitia lava flows: Crater count chronology and geological implications. *Journal of Geophysical Research: Planets*, 105(E6):15011–15025, 2000.
- [135] William K Hartmann and Gerhard Neukum. Cratering chronology and the evolution of mars. *Chronology and evolution of Mars*, pages 165–194, 2001.
- [136] GW Wetherill. Late heavy bombardment of the moon and terrestrial planets. In *Lunar and Planetary Science Conference Proceedings*, volume 6, pages 1539–1561, 1975.

- [137] Alessandro Morbidelli, J-M Petit, B Gladman, and J Chambers. A plausible cause of the late heavy bombardment. *Meteoritics & Planetary Science*, 36(3):371–380, 2001.
- [138] Rodney Gomes, Harold F Levison, Kleomenis Tsiganis, and Alessandro Morbidelli. Origin of the cataclysmic late heavy bombardment period of the terrestrial planets. *Nature*, 435(7041):466–469, 2005.
- [139] Harold F Levison, Luke Dones, Clark R Chapman, S Alan Stern, Martin J Duncan, and Kevin Zahnle. Could the lunar “late heavy bombardment” have been triggered by the formation of uranus and neptune? *Icarus*, 151(2):286–306, 2001.
- [140] Jean-Pierre Williams, Carolyn H van der Bogert, Asmin V Pathare, Gregory G Michael, Michelle R Kirchoff, and Harald Hiesinger. Dating very young planetary surfaces from crater statistics: A review of issues and challenges. *Meteoritics & Planetary Science*, 53(4):554–582, 2018.
- [141] David A Minton, Caleb I Fassett, Masatoshi Hirabayashi, Bryan A Howl, and James E Richardson. The equilibrium size-frequency distribution of small craters reveals the effects of distal ejecta on lunar landscape morphology. *Icarus*, 326:63–87, 2019.
- [142] PA Davis and KL Tanaka. Curvilinear ridges in isidis planitia, mars—the result of mud volcanism. In *Lunar and Planetary Science Conference*, volume 26, 1995.
- [143] Rebecca R Ghent, Steven W Anderson, and Taronish M Pithawala. The formation of small cones in isidis planitia, mars through mobilization of pyroclastic surge deposits. *Icarus*, 217(1):169–183, 2012.
- [144] MA Ivanov, H Hiesinger, G Erkeling, FJ Hielscher, and D Reiss. Major episodes of geologic history of isidis planitia on mars. *Icarus*, 218(1):24–46, 2012.
- [145] Goran Salamunićar and Sven Lončarić. Open framework for objective evaluation of crater detection algorithms with first test-field subsystem based on mola data. *Advances in Space Research*, 42(1):6–19, 2008.
- [146] Erik R Urbach and Tomasz F Stepinski. Automatic detection of sub-km craters in high resolution planetary images. *Planetary and Space Science*, 57(7):880–887, 2009.
- [147] Andy Liaw, Matthew Wiener, et al. Classification and regression by randomforest. *R news*, 2(3):18–22, 2002.



- 
- [148] Lourenço Bandeira, Wei Ding, and Tomasz F Stepinski. Detection of sub-kilometer craters in high resolution planetary images using shape and texture features. *Advances in Space Research*, 49(1):64–74, 2012.
- [149] Paul Viola and Michael Jones. Rapid object detection using a boosted cascade of simple features. In *Proceedings of the 2001 IEEE computer society conference on computer vision and pattern recognition. CVPR 2001*, volume 1, pages I–I. Ieee, 2001.
- [150] Wei Ding, Tomasz F Stepinski, Yang Mu, Lourenco Bandeira, Ricardo Ricardo, Youxi Wu, Zhenyu Lu, Tianyu Cao, and Xindong Wu. Subkilometer crater discovery with boosting and transfer learning. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2(4):1–22, 2011.
- [151] Joseph Paul Cohen, Henry Z Lo, Tingting Lu, and Wei Ding. Crater detection via convolutional neural networks. *arXiv preprint arXiv:1601.00978*, 2016.
- [152] Danielle M DeLatte, Sarah T Crites, Nicholas Guttenberg, Elizabeth J Tasker, and Takehisa Yairi. Segmentation convolutional neural networks for automatic crater detection on mars. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 12(8):2944–2957, 2019.
- [153] Haibo Li, Bei Jiang, Yuyuan Li, and Le Cao. A combined method of crater detection and recognition based on deep learning. *Systems Science & Control Engineering*, 9(sup2):132–140, 2021.
- [154] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 07-12-June:1–9, 2015.
- [155] A M Francis, P Sidiropoulos, E Vazquez, and Mullard Space. Real-Time Cloud Detection in High-Resolution Videos: Challenges and Solutions. In *6th International Workshop on On-Board Payload Data Compression*, 2018.
- [156] Augustus Odena, Vincent Dumoulin, and Chris Olah. Deconvolution and checkerboard artifacts. *Distill*, 2016.
- [157] Andrew L Maas, Awni Y Hannun, and Andrew Y Ng. Rectifier nonlinearities improve neural network acoustic models. In *Proc. icml*, volume 30, page 3, 2013.

- [158] Léon Bottou. Large-scale machine learning with stochastic gradient descent. In Yves Lechevallier and Gilbert Saporta, editors, *Proceedings of COMPSTAT'2010*, pages 177–186, Heidelberg, 2010. Physica-Verlag HD.
- [159] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
- [160] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159, 2011.
- [161] Dang Ha The Hien. A guide to receptive field arithmetic for Convolutional Neural Networks, 2017.
- [162] Wenjie Luo, Yujia Li, Raquel Urtasun, and Richard Zemel. Understanding the effective receptive field in deep convolutional neural networks. In *Advances in neural information processing systems*, pages 4898–4906, 2016.
- [163] L8 biome cloud validation masks. <https://landsat.usgs.gov/landsat-8-cloud-cover-assessment-validation-data>, 2016.
- [164] Ron Morfitt, Julia Barsi, Raviv Levy, Brian Markham, Esad Micijevic, Lawrence Ong, Pat Scaramuzza, and Kelly Vanderwerff. Landsat-8 operational land imager (oli) radiometric performance on-orbit. *Remote Sensing*, 7(2):2208–2237, 2015.
- [165] L7 irish cloud validation masks. <https://landsat.usgs.gov/landsat-7-cloud-cover-assessment-validation-data>, 2016.
- [166] L Baetens and O Hagolle. Sentinel-2 reference cloud masks generated by an active learning method. *Type: Dataset. Available online: https://zenodo.org/record/1460961 (accessed on 19 February 2019)*, 2018.
- [167] Sorour Mohajerani and Parvaneh Saeedi. Cloud-net+: A cloud segmentation cnn for landsat 8 remote sensing imagery optimized with filtered jaccard loss function. *arXiv preprint arXiv:2001.08768*, 2020.
- [168] Giorgio Morales, Alejandro Ramírez, and Joel Telles. End-to-end cloud segmentation in high-resolution multispectral satellite imagery using deep learning. In *2019 IEEE XXVI International Conference on Electronics, Electrical Engineering and Computing (INTERCON)*, pages 1–4. IEEE, 2019.
- [169] Alistair Francis and John Mrziglod. Sensei. <https://github.com/aliFrancis/SEnSeI>, 2021.

- [170] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*, 2016.
- [171] Jenia Golbstein. Keras deeplab v3.1+. <https://github.com/Golbstein/Keras-segmentation-deeplab-v3.1/commit/e3f0daaa79a729c022da658fc86eef82a6c7ceeb>.
- [172] Derek Hoiem, Santosh K Divvala, and James H Hays. Pascal voc 2008 challenge. *World Literature Today*, 2009.
- [173] David A Van Dyk and Xiao-Li Meng. The art of data augmentation. *Journal of Computational and Graphical Statistics*, 10(1):1–50, 2001.
- [174] Anze Zupanc. Improving cloud detection with machine learning. *Availabe online: <https://medium.com/sentinel-hub/improving-cloud-detection-with-machine-learning-c09dc5d7cf13> (accessed on 13 May 2020)*, 2017.
- [175] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794, 2016.
- [176] Rosa Coluzzi, Vito Imbrenda, Maria Lanfredi, and Tiziana Simoniello. A first assessment of the sentinel-2 level 1-c cloud mask product to support informed surface analyses. *Remote sensing of environment*, 217:426–443, 2018.
- [177] Jacob Høxbroe Jeppesen, Rune Hylsberg Jacobsen, Fadil Inceoglu, and Thomas Skjødeberg Toftegaard. A cloud detection algorithm for satellite imagery based on deep learning. *Remote sensing of environment*, 229:247–259, 2019.
- [178] Pasquale L Scaramuzza, Michelle A Bouchard, and John L Dwyer. Development of the landsat data continuity mission cloud-cover assessment algorithms. *IEEE Transactions on Geoscience and Remote Sensing*, 50(4):1140–1154, 2011.
- [179] Sorour Mohajerani and Parvaneh Saeedi. Cloud-net: An end-to-end cloud detection algorithm for landsat 8 imagery. In *IGARSS 2019-2019 IEEE International Geoscience and Remote Sensing Symposium*, pages 1029–1032. IEEE, 2019.
- [180] Giorgio Morales, Samuel G Huamán, and Joel Telles. Cloud detection for perusat-1 imagery using spectral and texture descriptors, ann, and panchromatic fusion. In *Brazilian Technology Symposium*, pages 1–7. Springer, 2017.

- [181] David Thompson and Philip Brodrick. Realizing machine learning’s promise in geoscience remote sensing. *Eos*, 102, July 2021.
- [182] L Bandeira, W Ding, and TF Stepinski. Automatic detection of sub-km craters using shape and texture information. In *Lunar and Planetary Science Conference*, number 1533, page 1144, 2010.
- [183] BA Ivanov, G Neukum, and R Wagner. Size-frequency distributions of planetary impact craters and asteroids. In *Collisional processes in the solar system*, pages 1–34. Springer, 2001.
- [184] IJ Daubar, Alfred S McEwen, Shane Byrne, MR Kennedy, and B Ivanov. The current martian cratering rate. *Icarus*, 225(1):506–516, 2013.
- [185] Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, Pierre-Antoine Manzagol, and Léon Bottou. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of machine learning research*, 11(12), 2010.
- [186] Original research by young twinkle students (orbyts): when can students start performing original research? *Physics Education*, 53(1), 2017.
- [187] Louis L McQuitty. Elementary linkage analysis for isolating orthogonal and oblique types and typal relevancies. *Educational and Psychological Measurement*, 17(2):207–229, 1957.
- [188] GG Michael and Gerhard Neukum. Planetary surface dating from crater size–frequency distribution measurements: Partial resurfacing events and statistical age uncertainty. *Earth and Planetary Science Letters*, 294(3-4):223–229, 2010.
- [189] GG Michael, SHG Walter, T Kneissl, W Zuschneid, C Gross, PC McGuire, A Dumke, Björn Schreiner, Stephan van Gasselt, Klaus Gwinner, et al. Systematic processing of mars express hrsc panchromatic and colour image mosaics: Image equalisation using an external brightness reference. *Planetary and Space Science*, 121:18–26, 2016.
- [190] Panagiotis Sidiropoulos, Jan-Peter Muller, Gillian Watson, Gregory Michael, and Sebastian Walter. Automatic coregistration and orthorectification (acro) and subsequent mosaicing of nasa high-resolution imagery over the mars mc11 quadrangle, using hrsc as a baseline. *Planetary and Space Science*, 151:33–42, 2018.

- [191] Klaus Gwinner, Ralf Jaumann, Ernst Hauber, Harald Hoffmann, Christian Heipke, Jürgen Oberst, Gerhard Neukum, Veronique Ansan, J Bostelmann, Alexander Dumke, et al. The high resolution stereo camera (hrsc) of mars express and its approach to science analysis and mapping for mars and its satellites. *Planetary and Space Science*, 126:93–138, 2016.
- [192] JL Dickson, LA Kerber, CI Fassett, and BL Ehlmann. A global, blended ctx mosaic of mars with vectorized seam mapping: A new mosaicking pipeline using principles of non-destructive image editing. In *Lunar and planetary science conference*, volume 49, pages 1–2, 2018.
- [193] Danielle DeLatte; Sarah Crites; Nicholas Guttenberg; Elizabeth Tasker; Takehisa Yairi. Mars crater segmentation dataset, 2019.
- [194] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.
- [195] EB Bierhaus, Alfred S McEwen, SJ Robbins, KN Singer, L Dones, MR Kirchoff, and J-P Williams. Secondary craters and ejecta across the solar system: Populations and effects on impact-crater-based chronologies. *Meteoritics & Planetary Science*, 53(4):638–671, 2018.
- [196] Wayne A Roberts. Secondary craters. *Icarus*, 3(4):348–364, 1964.
- [197] Eugene M Shoemaker. Preliminary analysis of the fine structure of the lunar surface in mare cognitum. *The nature of the lunar surface*, (65):23, 1965.
- [198] EM Shoemaker, RM Batson, HE Holt, EC Morris, JJ Rennilson, and EA Whitaker. Television observations from surveyor 3. *Journal of Geophysical Research*, 73(12):3989–4043, 1968.
- [199] Alfred S McEwen, Brandon S Preblich, Elizabeth P Turtle, Natalia A Artemieva, Matthew P Golombek, Michelle Hurst, Randolph L Kirk, Devon M Burr, and Philip R Christensen. The rayed crater zunil and interpretations of small impact craters on mars. *Icarus*, 176(2):351–381, 2005.
- [200] Colin M Dundas and Alfred S McEwen. Rays and secondary craters of tycho. *Icarus*, 186(1):31–40, 2007.
- [201] Michael H Carr. *The surface of Mars*, volume 6. Cambridge University Press, 2007.

- [202] Philip J Clark and Francis C Evans. Distance to nearest neighbor as a measure of spatial relationships in populations. *Ecology*, 35(4):445–453, 1954.
- [203] Kenneth L Tanaka, James A Skinner Jr, James M Dohm, Rossman P Irwin III, Eric J Kolb, Corey M Fortezzo, Thomas Platz, Gregory G Michael, and Trent M Hare. Geologic map of mars. 2014.
- [204] Crater Analysis Techniques Working Group et al. Standard techniques for presentation and analysis of crater size-frequency data. *Icarus*, 37(2):467–474, 1979.
- [205] Nadine G Barlow. Crater size-frequency distributions and a revised martian relative chronology. *Icarus*, 75(2):285–305, 1988.
- [206] Gregory Michael and Andrew Annex. craterstats. <https://github.com/ggmichael/craterstats>, 2021.
- [207] GG Michael, Thomas Platz, Thomas Kneissl, and Nico Schmedemann. Planetary surface dating from crater size–frequency distribution measurements: Spatial randomness and clustering. *Icarus*, 218(1):169–177, 2012.
- [208] GG Michael. Planetary surface dating from crater size–frequency distribution measurements: Multiple resurfacing episodes and differential isochron fitting. *Icarus*, 226(1):885–890, 2013.
- [209] GG Michael, T Kneissl, and As Neesemann. Planetary surface dating from crater size-frequency distribution measurements: Poisson timing analysis. *Icarus*, 277:279–285, 2016.
- [210] Antonin Guttman. R-trees: A dynamic index structure for spatial searching. In *Proceedings of the 1984 ACM SIGMOD international conference on Management of data*, pages 47–57, 1984.
- [211] DW Hughes. Meteorite incidence angles. *Journal of the British Astronomical Association*, 103:123–126, 1993.
- [212] Duncan Steel. Distributions and moments of asteroid and comet impact speeds upon the earth and mars. *Planetary and Space Science*, 46(5):473–478, 1998.
- [213] Mark J Cintala and Richard AF Grieve. Scaling impact melting and crater dimensions: Implications for the lunar cratering record. *Meteoritics & Planetary Science*, 33(4):889–912, 1998.
- [214] JB Murray and DC Heggie. Character and origin of phobos’ grooves. *Planetary and Space Science*, 102:119–143, 2014.

- [215] JB Murray, Jonathan C Iliffe, J.-P. Muller, Gerhard Neukum, Stephanie Werner, and MR Balme. New evidence on the origin of phobos' parallel grooves from hrsc mars express. In *37th Annual Lunar and Planetary Science Conference*, page 2195, 2006.
- [216] Paulo Ceppi, Florent Brient, Mark D Zelinka, and Dennis L Hartmann. Cloud feedback mechanisms and their representation in global climate models. *Wiley Interdisciplinary Reviews: Climate Change*, 8(4):e465, 2017.





## A | Computational Efficiency

An experiment was conducted to measure the computation time for the models used in Chapter 5, and to measure how the addition of SEnSel affected this computation. Each model was given a 257-by-257 pixel scene 1,600 times. For variants with SEnSel, different numbers of bands were given, but for models without SEnSel, 13 bands was used, as this is the number of bands which Sentinel-2 data contains. The number of parameters for each model, and the time taken per megapixel, is given in Table A.1. SEnSel adds a small amount of computation time to the models, although when only using a few bands, SEnSel has no noticeable affect.

Model	No. Parameters	Speed (ms/Megapixel)
NN	1154	36.8
SEnSel+NN (3 bands)	2.27e5	33.2
SEnSel+NN (8 bands)	2.27e5	51.6
SEnSel+NN (13 bands)	2.27e5	71.4
DLv3	2.11e6	94.6
SEnSel+DLv3 (3 bands)	2.34e6	88.4
SEnSel+DLv3 (8 bands)	2.34e6	103
SEnSel+DLv3 (13 bands)	2.34e6	121

Table A.1: Measured computational efficiency across both the 2-layer neural network and DeepLabv3+. SEnSel adds roughly 30 ms/Megapixel to either model, when 13 spectral bands are inputted.



## B | Detailed Results on Sentinel-2

As well as measuring the metrics over the total Sentinel-2 test sets, we also subdivided the test set based on the categorical tags assigned to each image (see Table B.1). Relating the various models' performance with the content of the images gives a more nuanced view of how they can be expected to behave in the real world.

Of the metrics used, OA and Recall are relatively unaffected by the ratio of pixels which are cloudy, whereas BA, Precision and  $F_1$  are all highly dependent on the class distribution. Therefore, when inspecting these results, it is also important to consider the percentage of cloudy pixels that are present in a given category of images, and only compare results across categories in metrics where it is reasonable to do so.

OA	Total test set	forest /jungle	snow /ice	agricultural	desert /barren	shrublands	open _ water	low	high	thin	thick	isolated	extended	cumulus	stratocumulus	cirrus	ice _ clouds
DLv3 - S2	94.04	92.52	86.05	92.22	94.88	95.31	94.14	<b>94.92</b>	92.97	91.10	95.26	89.54	93.92	<b>95.00</b>	96.42	91.93	95.30
DLv3 - S2&L8 [common bands]	95.19	93.28	<b>89.71</b>	<b>95.07</b>	95.85	<b>95.92</b>	93.83	94.82	<b>94.52</b>	92.28	95.93	<b>90.48</b>	<b>94.83</b>	94.93	96.63	<b>93.84</b>	96.85
SEnSel+DLv3 - S2 [fixed bands]	94.46	91.60	88.36	92.61	<b>95.94</b>	94.56	94.01	94.15	93.43	91.11	95.27	88.95	94.64	94.20	95.93	92.86	96.58
SEnSel+DLv3 - S2	93.08	90.92	87.14	91.35	93.16	93.81	93.61	93.69	92.07	90.43	94.31	89.15	92.99	93.86	95.78	91.33	93.66
SEnSel+DLv3 - S2&L8	93.73	90.70	84.87	90.20	93.77	93.78	<b>94.31</b>	94.66	92.49	89.97	95.68	88.90	94.23	94.74	<b>96.86</b>	91.79	<b>97.45</b>
<b>BA</b>																	
DLv3 - S2	94.03	91.35	85.94	91.60	96.43	94.31	94.12	94.56	91.05	91.16	94.05	88.58	89.99	<b>94.57</b>	94.08	89.42	74.35
DLv3 - S2&L8 [common bands]	95.18	91.94	<b>89.41</b>	<b>93.61</b>	<b>97.21</b>	<b>94.75</b>	93.84	94.09	93.33	<b>92.37</b>	93.82	<b>89.60</b>	<b>89.27</b>	94.12	93.67	<b>92.46</b>	73.15
SEnSel+DLv3 - S2 [fixed bands]	94.42	90.44	88.49	91.04	96.38	92.69	94.01	93.54	90.81	91.03	93.16	88.25	90.21	93.48	92.35	89.72	78.89
SEnSel+DLv3 - S2	93.10	89.10	87.20	89.21	93.07	90.57	93.60	93.37	91.00	90.67	93.10	87.89	89.93	93.46	93.05	89.99	80.03
SEnSel+DLv3 - S2&L8	93.73	88.43	85.04	87.75	92.31	89.87	<b>94.27</b>	<b>94.57</b>	91.30	90.26	<b>94.64</b>	87.23	91.27	94.51	<b>94.46</b>	90.06	<b>82.88</b>
<b>Precision</b>																	
DLv3 - S2	94.57	91.64	83.03	85.03	62.24	90.33	94.67	96.79	96.71	93.81	97.69	85.09	98.80	96.96	98.45	96.48	98.63
DLv3 - S2&L8 [common bands]	95.61	93.76	<b>88.83</b>	<b>93.44</b>	67.07	92.39	93.25	96.09	<b>97.66</b>	<b>94.84</b>	97.22	<b>86.43</b>	98.61	96.32	98.25	<b>97.66</b>	98.54
SEnSel+DLv3 - S2 [fixed bands]	94.58	90.00	84.55	88.23	<b>68.00</b>	90.61	93.78	95.95	96.37	93.20	97.02	83.33	98.79	96.10	97.88	96.36	98.87
SEnSel+DLv3 - S2	94.00	91.36	83.47	86.98	55.35	92.28	94.14	96.24	97.03	94.09	97.40	85.37	98.87	96.45	98.19	97.01	98.99
SEnSel+DLv3 - S2&L8	94.42	92.89	80.26	85.27	58.12	<b>94.23</b>	<b>95.97</b>	97.15	97.07	93.95	<b>97.94</b>	86.25	<b>99.00</b>	<b>97.16</b>	<b>98.51</b>	96.90	<b>99.08</b>
<b>Recall</b>																	
DLv3 - S2	94.17	87.18	85.18	<b>90.03</b>	98.30	92.18	93.30	95.60	94.34	90.81	96.20	85.31	94.59	95.72	97.35	93.45	96.51
DLv3 - S2&L8 [common bands]	95.31	87.15	87.19	89.95	<b>98.83</b>	92.24	94.24	<b>96.20</b>	<b>95.38</b>	<b>91.82</b>	<b>97.57</b>	<b>86.62</b>	<b>95.77</b>	<b>96.30</b>	97.81	94.67	98.22
SEnSel+DLv3 - S2 [fixed bands]	94.99	86.27	<b>89.46</b>	87.08	96.91	88.70	94.02	95.32	95.29	91.51	96.92	85.86	95.40	95.43	97.36	<b>94.77</b>	97.61
SEnSel+DLv3 - S2	92.89	82.60	87.65	83.84	92.95	83.67	92.75	94.30	92.83	89.28	95.26	83.60	93.50	94.54	96.87	92.14	94.45
SEnSel+DLv3 - S2&L8	93.73	80.32	86.27	81.58	90.56	81.54	92.28	94.84	93.34	88.59	96.49	81.56	94.74	95.13	<b>97.82</b>	92.84	<b>98.29</b>
<b>F<sub>1</sub></b>																	
DLv3 - S2	94.37	89.35	84.09	87.46	76.22	91.25	93.98	<b>96.20</b>	95.51	92.28	96.94	85.20	96.65	<b>96.33</b>	97.90	94.94	97.56
DLv3 - S2&L8 [common bands]	95.46	<b>90.33</b>	<b>88.00</b>	<b>91.66</b>	79.91	<b>92.32</b>	93.74	96.15	<b>96.50</b>	<b>93.30</b>	<b>97.40</b>	<b>86.53</b>	<b>97.17</b>	96.31	98.03	<b>96.14</b>	98.38
SEnSel+DLv3 - S2 [fixed bands]	94.79	88.10	86.94	87.65	<b>79.92</b>	89.64	93.90	95.64	95.83	92.35	96.97	84.58	97.06	95.76	97.62	95.56	98.23
SEnSel+DLv3 - S2	93.44	86.76	85.51	85.38	69.38	87.76	93.44	95.26	94.88	91.62	96.32	84.47	96.11	95.49	97.52	94.52	96.66
SEnSel+DLv3 - S2&L8	94.07	86.15	83.16	83.38	70.80	87.43	<b>94.09</b>	95.98	95.17	91.19	97.21	83.84	96.82	96.13	<b>98.16</b>	<b>94.83</b>	<b>98.68</b>
No. of scenes	257	42	41	41	34	40	66	129	126	110	156	90	137	140	92	104	51
Cloud %	53.05	36.02	43.29	30.14	8.34	26.53	49.06	67.21	79.22	58.60	78.09	35.30	92.74	68.64	85.74	81.09	97.27

Table B.1: DeepLabv3+ model results over the Sentinel-2 test set (257 scenes). We also subdivide the total test by the presence of several categories of surface type and cloud properties, described in Section 4. Categories with fewer than 30 subscenes are omitted. The best result for each metric and image category is bolded.

## C | Concerning YOLOv5

You Only Look Once v5 (YOLOv5) is an extension to YOLOv3, but confusingly, not YOLOv4. The first author of the YOLO model family ended involvement after YOLOv3, citing concerns about the ethics of object detection applications [75]. Development has since bifurcated, with two independent teams working on YOLOv4 and YOLOv5 separately. More information about the design and structure of the YOLO model family can be found in Section 2.2.5.

YOLOv5 is still being actively improved upon and, at the time of writing, has not yet been published as a paper. As a result, many in the computer vision community remain sceptical regarding the claimed advantages of YOLOv5 over the other versions. Nevertheless, given that in this thesis testing was carried out by myself on crater detection, the performance of the model in this specific application was trusted, and claims as to its efficacy in other domains are not vitally important.

Previous versions of YOLO were implemented using a custom machine learning framework, *Darknet*<sup>1</sup>. This made YOLOv5—which is instead written natively in the *PyTorch* framework—more practical to work with. This was a significant motivating factor when deciding which model to select for the study. In hindsight, using YOLOv3 would have provided a more reliable, understandable, and reproducible model, however, given that YOLOv5 was not carried forward into the analysis in Chapter 9, this is not such a concern.

---

<sup>1</sup><https://pjreddie.com/darknet/>

