Machine Learning Models for On-Orbit Detection of Temperature and Chlorophyll Ocean Fronts

by

Violet C. Felt

B.S. Computer Science and Engineering
Massachusetts Institute of Technology, 2022

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Engineering in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 2022

Author .....
Department of Electrical Engineering and Computer Science
May 6, 2022

Certified by .....
Kerri L. Cahoy
Associate Professor of Aeronautics and Astronautics
Thesis Supervisor

Accepted by .....
Katrina LaCurts
Chair, Master of Engineering Thesis Committee

Machine Learning Models for On-Orbit Detection of Temperature and Chlorophyll Ocean Fronts

by

Violet C. Felt

Submitted to the Department of Electrical Engineering and Computer Science
on May 6, 2022, in partial fulfillment of the
requirements for the degree of
Master of Engineering in Electrical Engineering and Computer Science

Abstract

Small-scale ocean fronts play a significant role in absorbing the excess heat and CO2 generated by climate change, yet their dynamics are not well understood. Existing in-situ and remote sensing measurements of the ocean are of inadequate spatial and temporal coverage to globally map small-scale ocean fronts, and existing algorithms to generate ocean front maps are computationally intensive. We propose machine learning (ML) models to detect temperature and chlorophyll ocean fronts from unprocessed satellite imagery, significantly reducing the standard resources and computational times needed for detecting ocean fronts. These models are developed with resource-constrained satellite imaging platforms like CubeSats in mind, as such platforms are able to address the spatial and temporal coverage challenges. The highest performing models achieve accuracies of 96% and make predictions in milliseconds using less than 100 MB of storage; these capabilities are well-suited for CubeSat deployment.

Thesis Supervisor: Kerri L. Cahoy
Title: Associate Professor of Aeronautics and Astronautics

Acknowledgments

This thesis is the product of 11 months of intense thinking and coding.

My deepest thanks to my advisor Kerri Cahoy, who took me under her wing and guided me through the world of research. I am grateful for her invaluable insights, her humor, and her dedication to every graduate student in her lab. She has contributed to my growth as an individual and my confidence as a researcher, and will forever be my role model of a powerful female professor.

I am also grateful to Joe Kusters and Shreeyam Kacker, who helped me define my research and make the tough technical decisions. This thesis would be much lower quality without them.

I would like to thank Alex, Hannah, and the rest of the STAR Lab team, for welcoming a Computer Science student into the Aero Astro department with open arms and explaining very basic concepts to me.

To Agnes, Chris, and Delia: thank you for embarking on the wild journey that is MIT grad school with me and keeping me sane.

To my immediate and extended family—thank you for listening to my research problems and celebrating my grad school journey!

Finally, I would like to acknowledge Ben. Your unwavering support has allowed me to achieve my impossible.

THIS PAGE INTENTIONALLY LEFT BLANK

Contents

THIS PAGE INTENTIONALLY LEFT BLANK

List of Figures

THIS PAGE INTENTIONALLY LEFT BLANK

List of Tables

Chapter 1

Introduction

1.1 The Importance of Ocean Fronts

The seemingly continuous ocean is full of discrete features such as ocean fronts. Ocean fronts are narrow zones of strong gradients of water properties (such as temperature, salinity, nutrients, etc.) that separate distinct, relatively uniform water masses. Ocean fronts are the main structural elements of the oceanic realm, impacting everything from marine trophic levels to the climate. They occur on a variety of spatial and temporal scales, from fronts a few meters long that last days to fronts thousands of kilometers long that last millions of years [6].

Ocean fronts play a key role in the ecology of marine life–providing habitats for activities such as foraging, reproduction, nursing, and migration. Fishermen have routinely tracked ocean fronts to monitor marine life [6], but scientists have recently also started tracking ocean fronts to monitor climate change. The ocean absorbs almost all of the excess heat and about a third of the CO_2 generated by fossil fuel emissions and land-use changes. While the dynamics through which heat and CO_2 are absorbed by the ocean are not fully understood, this absorption may be influenced by small-scale ocean fronts [16].

Although ocean fronts are typically stationary or seasonally periodic, recent ocean front tracking indicates irregular, erratic ocean front activity. Perhaps the most famous example of this is the Gulf Stream, a large-scale ocean front that has lately

weakened in intensity, and is likely approaching collapse [8]. The collapse of the Gulf Stream would have a disastrous impact on global weather systems, leading to greater cooling and more powerful storms across the Northern Hemisphere, as well as severe disruption to the rain that billions of people rely on for crops in Africa, South America, and India [17]. Ocean front tracking is essential for making timely, informed decisions about climate change.

1.2 Tracking Ocean Fronts

The two most commonly tracked types of ocean fronts are sea surface temperature (SST) fronts and chlorophyll-a concentration (CHL) fronts. Traditionally, CHL and SST measurements are captured in-situ using buoys, gliders, and science cruises. Local maps of CHL and SST are created from these measurements, then fronts are hand-drawn on these maps in areas of sharp gradients; this is currently a lengthy and unstandardized process [6].

More recently, remote sensing data from airborne sensors and satellites is being compiled to create global maps of CHL and SST, as shown in Figure 1-1 [6]. These maps are then automatically and objectively analyzed by an algorithm such as the Cayula-Cornillon Algorithm (CCA) or the Belkin-O'Reilly Algorithm (BOA), both developed by oceanographers to detect ocean fronts in remote sensing imagery [9, 7].

1.3 Spatial and Temporal Problems

While remote sensing data and algorithms have greatly decreased the computational barrier to tracking ocean fronts, there are still issues in the spatial and temporal coverage of ocean fronts, especially small-scale ocean fronts. Large satellite missions often have poor spatial resolution for detecting small-scale ocean fronts, or poor temporal resolution because their orbits require days to weeks of time to image the entire globe [26]. Many satellites do not even spend resources on ocean imaging, instead focusing on land mass imaging.

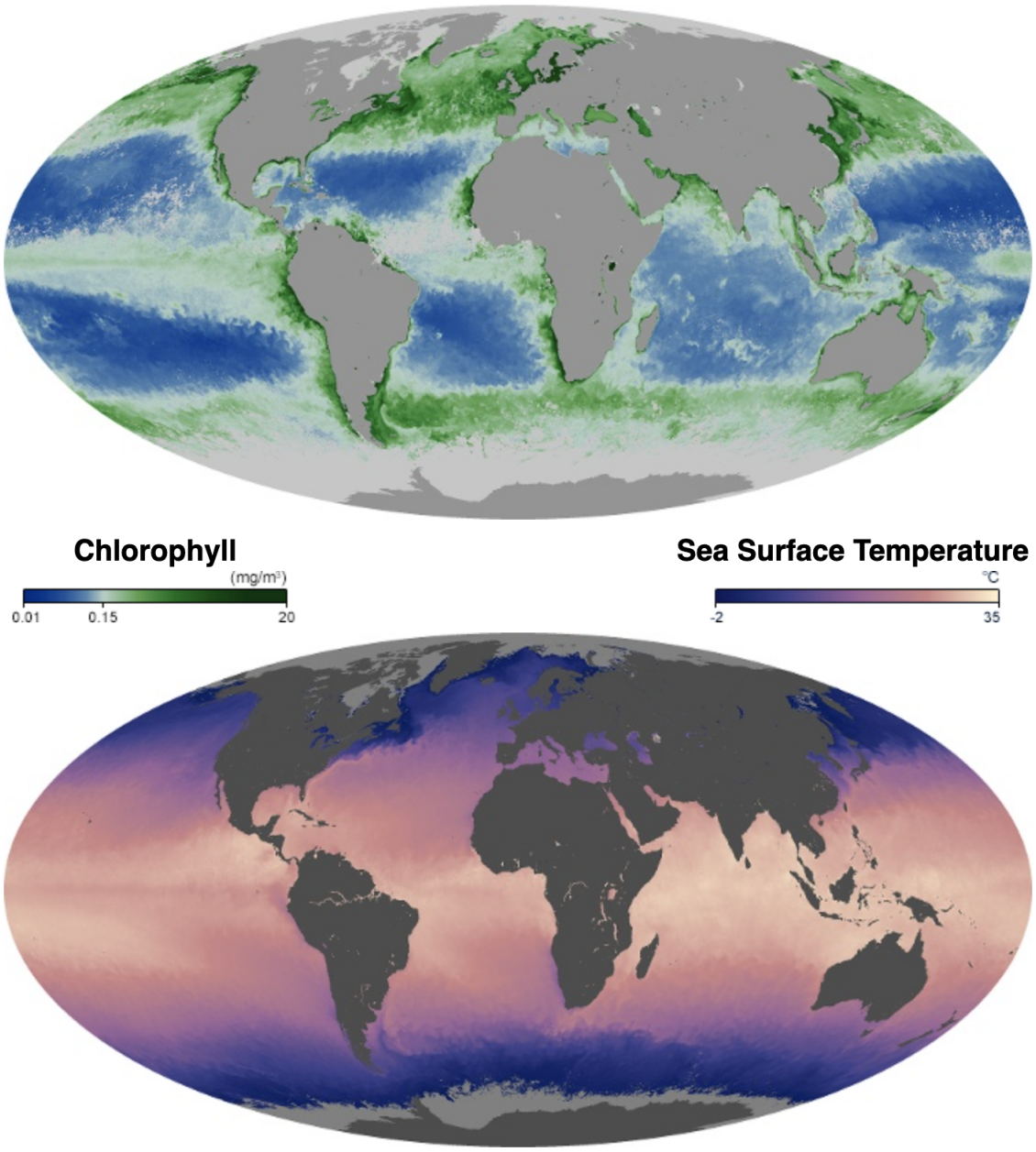

Figure 1-1: Remote sensing maps of CHL and SST, created from remote sensing data collected April 2020. Source: [38]

Additionally the time between when a satellite image is taken and when the ocean fronts in the image are identified can be weeks, due to many levels of intermediate processing (including incorporating data from other airborne and land sources) [58]. This lag prevents real-time tracking of ocean fronts, important for fishermen and scientists alike.

1.4 Tools for Solutions

1.4.1 CubeSats

During the past decade, CubeSats have gained momentum as a means to address targeted science questions (like the dynamics of ocean fronts) in a rapid and more affordable manner than traditional, complex satellites. CubeSats are miniaturized satellites made up of 10 cm x 10 cm x 10 cm cubic modules, and use off-the-shelf components for their electronics and structure, providing low-cost access to space. When proliferated, they combine the temporal resolution of Geostationary Earth Orbit missions with the spatial resolution of Low Earth Orbit missions, breaking the traditional trade-off in Earth observation missions [43].

However, their affordability leads to size, weight, and power (SWaP) constraints, including limited Earth downlink capabilities. Research CubeSats typically operate in an International Space Station or sun synchronous orbit with a single ground station, supporting only 1-2 ground passes per day [11]. Increasing the number of ground stations helps increase downlink capabilities, but also incurs additional cost. To maximize the scientific return of a CubeSat, remote sensing images should be analyzed on-orbit, so that the most scientifically valuable images can be prioritized for downlink.

Recent developments in commercial electronics components have increased the processing power and efficiency available for CubeSats, to the point where complex computations such as those required by some algorithms and even machine learning models can be completed on-orbit. For example, Nvidia and Xilinx are making parallel

processing available in embedded forms that can be used onboard small satellites like CubeSats [3].

1.4.2 Machine Learning

Traditional methods of ocean front detection involve running computationally intense (and sometimes recursive [7]) algorithms. These algorithms are impractical for on-orbit ocean front detection, even on CubeSats with state-of-the-art processing components [3]. Additionally, traditional methods of ocean front detection involve collecting data from multiple sources [13], impossible when detecting ocean fronts on-orbit. The task of on-orbit ocean front detection should be solved with a new tool: machine learning.

A machine learning (ML) model is a type of artificial intelligence (AI) that is able to adapt and learn a task (such as detecting ocean fronts from satellite imagery) without following explicit instructions. It learns the patterns between inputs (such as satellite images) and outputs (such as ocean front maps) using a vast quantity of training data, then is able to make new inferences/predictions from new data using the patterns it has learned.

While training an ML model is very computationally intensive, performing inference with an ML model is not as intensive, and thus ML models are suitable for the task of ocean front detection on a CubeSat [23]. Convolutional Neural Networks (CNNs) are a type of ML model well suited for ocean front detection. Translation- and scale-invariant, CNNs can detect ocean fronts in any part of an image, and detect ocean fronts of any size [12]. Additionally, CNNs accept variable size input images, which means they can detect ocean fronts in a 224 x 224 pixel image with the same level of accuracy as in a 4000 x 4000 pixel image. This flexibility is useful for SWaP-constrained on-orbit processing.

Since ocean fronts are narrow, high-gradient zones in satellite images, the task of front detection can be compared to the task of edge detection; a well-established problem in computer vision. Some of the most state-of-the-art edge detection methods are CNNs, trained on datasets of natural images with ground truth human edge

annotations [31]. These models approach human levels of accuracy and can provide starting architectures for ocean front detection models [30, 28].

1.5 Thesis Outline

The crux of this thesis is developing ML models to detect CHL and SST ocean fronts from a CubeSat. These models will integrate into a computer vision pipeline that determines which CubeSat images are downlinked to Earth, and in what order. These ML models require no external data and no intermediate processing, greatly decreasing the standard resources and computational time needed to detect ocean fronts.

Chapter 1 introduces the field: the importance of ocean front tracking, existing problems with ocean front tracking, and potential tools for improving ocean front tracking. Section 1.1 details the use of ocean front tracking in fields from fishing to climate change, while Section 1.2 details the types of ocean fronts tracked, and the tools currently used to track them. Section 1.3 describes the spatial, temporal, and computational problems with existing ocean front tracking, while Section 1.4 describes useful tools for solving these problems: CubeSats and ML.

Chapter 2 explains our approach: the existing relevant satellite missions, the existing algorithms used to detect ocean fronts, the existing models used to detect edges, and an example existing CubeSat mission these models could be deployed on. Section 2.1 details a relevant satellite mission for creating training data, while Section 2.2 details the traditional pipeline for detecting ocean fronts (and our pipeline for creating ground truth data). Section 2.3 describes existing edge detection datasets and model architectures, and Section 2.4 describes a specific CubeSat application and the associated image processing pipeline.

Chapter 3 explains our analysis: the decisions we made in creating training and ground truth data, and the decisions we made in creating and training the ML models. Section 3.1 details the specifics of creating a training dataset, while Section 3.2 details the specifics of creating the corresponding ground truth dataset, including

the introduction of a human source of ground truth data. Section 3.3 describes the architecture adjustments made to the edge detection models to address the task of on-orbit CubeSat ocean front detection, and Section 3.4 describes the parameters of the training process.

Chapter 4 explains our results: the performance of each ML model on the training datasets, and the resource utilization of each ML model. Section 4.1 details the qualitative and quantitative performance of the models on the training datasets, including a deep dive into the relationship between temperature and chlorophyll ocean front predictions. Section 4.2 describes the inference speed and memory requirements of each model, and compares these requirements to CubeSat constraints.

Chapter 5 summarizes our work: the scope of this thesis, general conclusions, and areas for future work. Section 5.1 details the main concepts covered in our work and the main results of our work, including a model recommendation for CubeSat deployment. Section 5.2 describes three areas for further research which could strengthen our understanding of ocean fronts and increase model performance.

The approach of this thesis is described graphically in Figures 1-2 and 1-3. Figure 1-2 shows the current pipeline for detecting ocean fronts, each step of which is explained in Chapters 2 and 3. Figure 1-3 shows our proposed pipeline for detecting ocean fronts, the details of which are explained in Chapters 2 and 3.

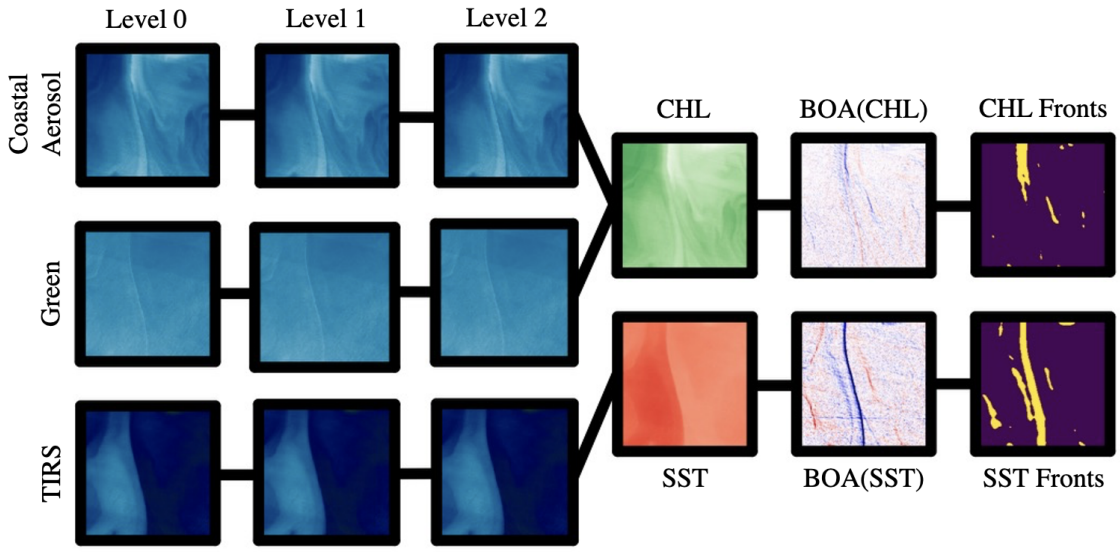

Figure 1-2: The traditional image processing pipeline for detecting ocean fronts. The inputs are unprocessed Thermal Infrared (TIRS), Green, and Coastal Aerosol bands. The intermediate processing steps include Level 1 and Level 2 preprocessing, CHL and SST calculations, and the Belkin O'Reilly Algorithm (BOA). The outputs are CHL and SST fronts. This image was constructed with a Landsat training scene, detailed in Appendix A.

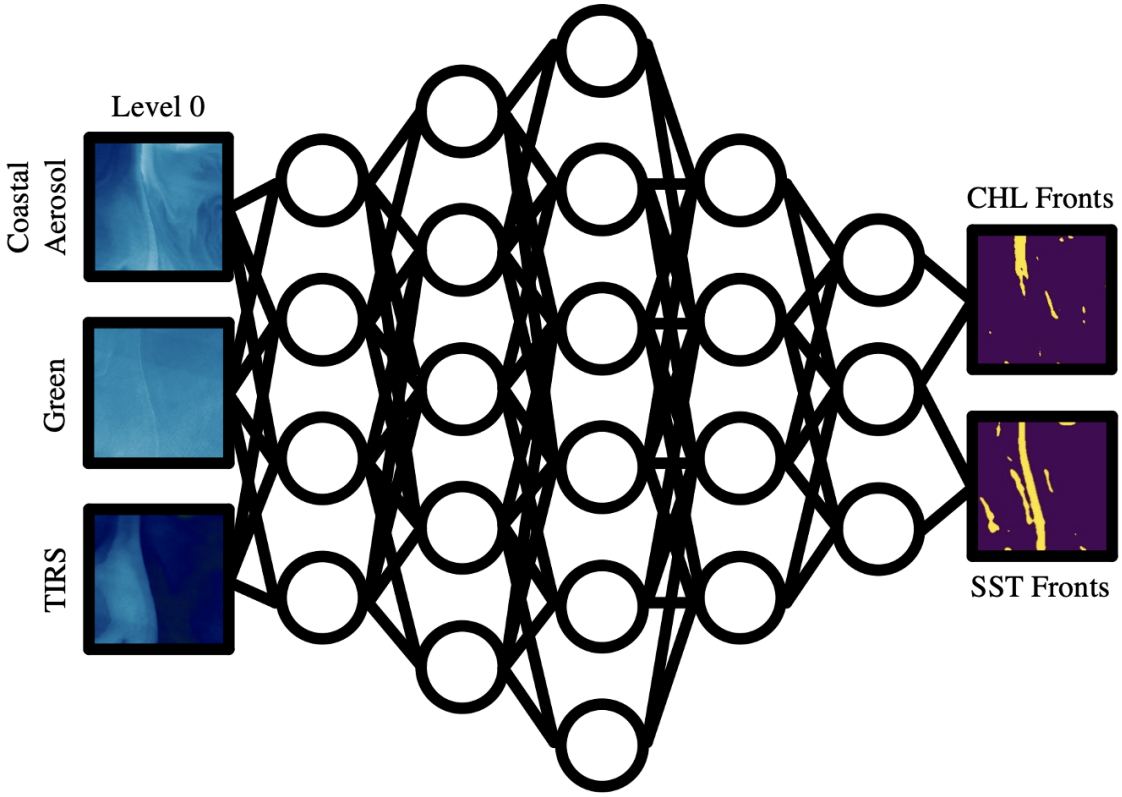

Figure 1-3: The proposed image processing pipeline for detecting ocean fronts, utilizing an ML model. The inputs are unprocessed Thermal Infrared (TIRS), Green, and Coastal Aerosol bands. The intermediate circles represent neurons in a convolutional neural network (CNN). The outputs are CHL and SST fronts. This image was constructed with a Landsat training scene, detailed in Appendix A.

THIS PAGE INTENTIONALLY LEFT BLANK

Chapter 2

Approach

2.1 Creating Training Data

In order to train an ML model, we must have many examples of inputs and outputs similar to the data we expect the model to process. Because our ML models are detecting ocean fronts from satellite data, we must use large quantities of existing satellite data to train the models.

There are multiple existing satellites that observe the ocean and capture both visible and infrared wavelengths, needed to measure CHL (0.44 micrometers) and SST (>8 micrometers) [5, 1, 37]. These satellites are listed in Table 2.1.

We choose Landsat 8 data to create the training dataset, primarily because Landsat has a much finer spatial resolution than the other satellites (necessary to capture small-scale ocean fronts) and Landsat stores data in an ML-friendly way using Google Earth Engine.

Table 2.1: Active satellite missions that observe the ocean and capture the range of wavelengths necessary to compute CHL and SST. Source: [5, 1, 37]

Agency	Operational Missions	Launch Years	Relevant Instruments	Band Count	Spectral Coverage (micrometers)
NASA	Terra, Aqua	1999, 2002	MODIS	36	0.41 - 14.2
NOAA	Suomi-NPP, JPSS-1	2011, 2017	VIIRS	22	0.41 - 12.0
USGS	Landsat 7, Landsat 8	1999, 2013	OLI, TIRS	11	0.43 - 12.5

Table 2.2: Landsat 8 spectral bands. Source: [37]

Instrument	Band	Wavelength (micrometers)	Spatial Resolution (meters)
OLI	Band 1 - Coastal Aerosol	0.43-0.45	30
OLI	Band 2 - Blue	0.45-0.51	30
OLI	Band 3 - Green	0.53-0.59	30
OLI	Band 4 - Red	0.64-0.67	30
OLI	Band 5 - Near Infrared (NIR)	0.85-0.88	30
OLI	Band 6 - Short-Wave Infrared (SWIR) 1	1.57-1.65	30
OLI	Band 7 - Short-Wave Infrared (SWIR) 2	2.11-2.29	30
OLI	Band 8 - Panchromatic	0.50-0.68	15
OLI	Band 9 - Cirrus	1.36-1.38	30
TIRS	Band 10 - Thermal Infrared (TIRS) 1	10.6-11.19	100
TIRS	Band 11 - Thermal Infrared (TIRS) 2	11.50-12.51	100

2.1.1 Landsat Program

The Landsat Program is a series of Earth-observing satellite missions, managed by NASA and the US Geological Survey. The first satellite (Landsat 1) was launched in 1972; the most recent satellite (Landsat 9) was launched in 2021. The remote sensing data acquired by the Landsat Program represents the longest continuous collection of space-based moderate-resolution remote sensing data. The ground resolution and spectral bands of the Landsat satellites were specifically chosen to track land use and document land change due to climate change, urbanization, drought, wildfire, and other natural and man-made changes [27].

2.1.2 Landsat Bands

The most recent satellites (Landsats 8 and 9) use two instruments–Operational Land Imager (OLI) and Thermal Infrared Sensor (TIRS)–to collect data in 11 bands detailed in Table 2.2. Band 1 is a new band added for coastal and aerosol studies, such as measuring chlorophyll concentrations in coastal regions. Band 1 reflects wavelengths scattered by dust, smoke, and water particles in the air, and therefore is also important for atmospheric correction of satellite imagery [37].

2.1.3 Landsat Preprocessing

Images taken by satellites are subject to distortion as a result of sensor, solar, atmospheric, and topographic effects [2]. The Landsat Program provides three levels of processed images for download: Levels 0, 1 and 2. Sample bands at these processing levels are shown in Figure 2-1. The Level 0 product consists of raw sensor data from each imaging band. The Level 1 Top of Atmosphere (TOA) Reflectance product consists of Level 0 data that is radiometrically calibrated and orthorectified using ground control points and a digital elevation model [21]. The Level 2 Surface Reflectance product consists of Level 1 data processed by the Land Surface Reflectance Code algorithm, which corrects for the temporally, spatially, and spectrally varying scattering and absorbing effects of atmospheric gases, aerosols, and water vapor. This algorithm also uses auxiliary atmospheric composition data collected by ground stations and other satellites. Consequently, it takes an average of 16 days to process Landsat data from Level 0 to Level 2 [58]. Our ML models accept Level 0 input data to circumvent this long processing time and need for auxiliary data.

2.2 Creating Ground Truth Data

In addition to the large quantities of satellite data we need as inputs to train our ML models, we also need corresponding ocean front data outputs (called ground truth data). We can computationally identify the ocean fronts present in our input data using a set of existing algorithms.

2.2.1 NASA Algorithms

NASA has standardized algorithms for calculating chlorophyll-a concentration and sea surface temperature from surface reflectance data (Landsat Level 2 data). A different set of coefficients are used for each satellite instrument.

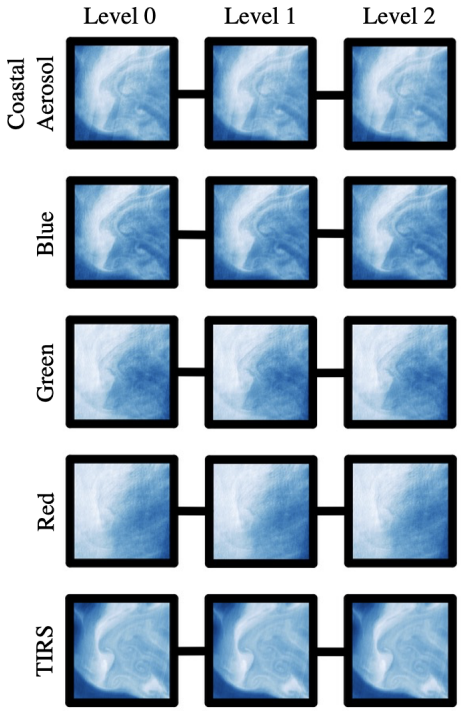

Figure 2-1: Five Landsat bands processed at Level 0, Level 1, and Level 2. This image was constructed with a Landsat training scene, detailed in Appendix A.

Chlorophyll-a (CHL) Concentration

To compute CHL, we utilize a fourth-order polynomial relationship between Landsat Bands 1 (Coastal Aerosol), 2 (Blue), 3 (Green) and chlorophyll-a concentration.

$$log_{10}(CHL) = a_0 + \sum_{i=1}^{4} a_i (log_{10}(\frac{\lambda_{BLUE}}{\lambda_{GREEN}}))^i$$

This algorithm returns the near-surface concentration of chlorophyll-a in units of $\frac{mg}{m^3}$, calculated using an empirical relationship derived from in-situ chlorophyll-a measurements and corresponding satellite imagery. λ_{BLUE} represents max(Band 1, Band 2) while λ_{GREEN} represents Band 3. a_0 through a_4 represent instrument-specific coefficients [18, 39].

An example CHL computation is shown in Figure 2-2.

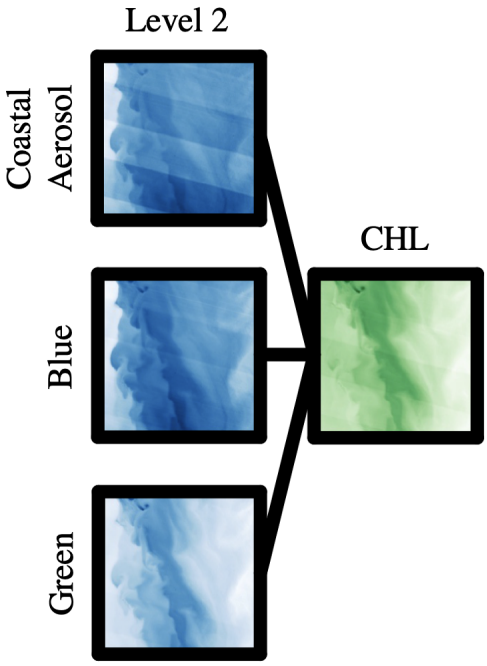

Figure 2-2: The CHL algorithm applied to Coastal Aerosol, Blue, and Green surface reflectance bands. This image was constructed with a Landsat training scene, detailed in Appendix A.

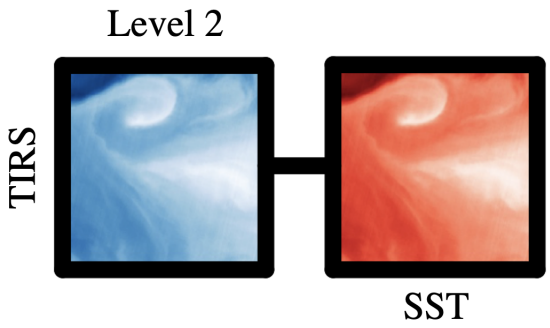

Figure 2-3: The SST algorithm applied to a TIRS surface reflectance band. This image was constructed with a Landsat training scene, detailed in Appendix A.

Sea Surface Temperature (SST)

To compute SST, we utilize a linear relationship between Landsat Band 10 (TIRS 1) and sea surface temperature.

$$SST = a_0 + a_1 \lambda_{TIRS}$$

This algorithm returns the skin sea surface temperature in units of $°C$, calculated using an empirical relationship derived from in-situ surface temperature measurements and corresponding satellite imagery. λ_{TIRS} represents Band 10. a_0 and a_1 represent instrument-specific coefficients [20].

An example SST computation is shown in Figure 2-3.

2.2.2 Belkin O'Reilly Algorithm

To detect ocean fronts, we use the Belkin O'Reilly Algorithm (BOA). BOA accepts SST and log-normalized CHL satellite imagery and produces gradient magnitude images, where gradient magnitude corresponds to oceanic front strength [7].

While there are many widely accepted methods for detecting SST fronts, there are few established methods for detecting CHL fronts. Important CHL spatial patterns such as blooms and ridges are often clipped by traditional noise-reducing median filters in early algorithmic stages [7].

BOA utilizes a recursive, shape-preserving, contextual median filter that removes noise while keeping CHL features. The filter operates on a 3 km x 3 km window while considering a larger 5 km x 5 km context, analyzing 1D slices of the window and context to determine whether or not to filter the window's central pixel [7].

This recursive median filter is applied until convergence (when the pixel values stop changing). Then, the Sobel operator is applied to calculate pixel-wise ocean front intensity. The Sobel operator is a standard edge detection filter in computer vision, utilizing two 3x3 kernels to calculate horizontal and vertical gradient approximations, then combining these two approximations to produce a gradient magnitude approximation [24, 7].

Detailed code for this algorithm can be found in Appendix B.

$$G_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

$$G_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

$$G = \sqrt{G_x^2 + G_y^2}$$

BOA is computationally intensive. The number of iterations until convergence is $O(N)$, where N is the number of pixels in the image [7]. Experimentally, it takes an average of 22 seconds to process a 224x224 pixel image with BOA using a Google Colab GPU.

Two example images processed with BOA are shown in Figure 2-4.

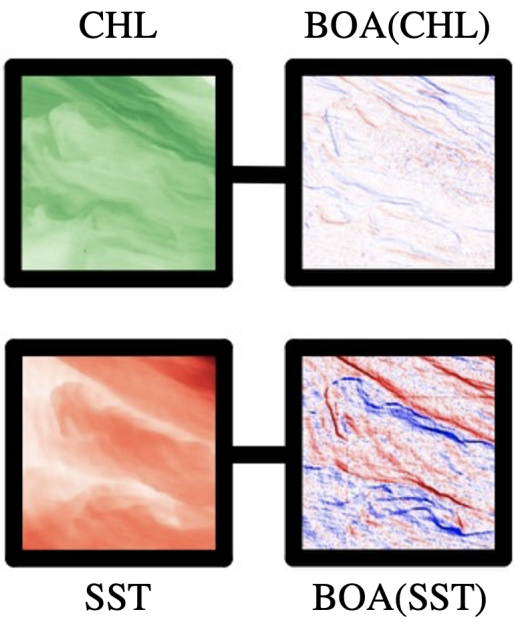

Figure 2-4: The Belkin O'Reilly algorithm applied to CHL and SST images. The red/blue color corresponds to the orientation of the ocean front, and a stronger color is equivalent to a more intense ocean front. This image was constructed with a Landsat training scene, detailed in Appendix A.

2.3 Existing Datasets and Model Architectures

In order to inform our ocean front detection model training pipeline, we explore related datasets and models in the field of edge detection.

2.3.1 Berkeley Segmentation Dataset 500

The Berkeley Segmentation Dataset 500 (BSDS500) is a popular dataset for training contour/edge detection models. It is composed of 500 images of natural scenes and their corresponding human-annotated edge maps. Each image is annotated by at least three humans, and edge maps are binary images of "edge" and "not edge" pixels [31]. BSDS500 is a benchmark dataset for edge detection models, and models are compared on this dataset using F_1 scores. An F_1 score is a way to measure the accuracy of binary classification models, discussed further in Section 4.1. The human F_1 score for the BSDS500 dataset (which measures how similar different human annotations of the

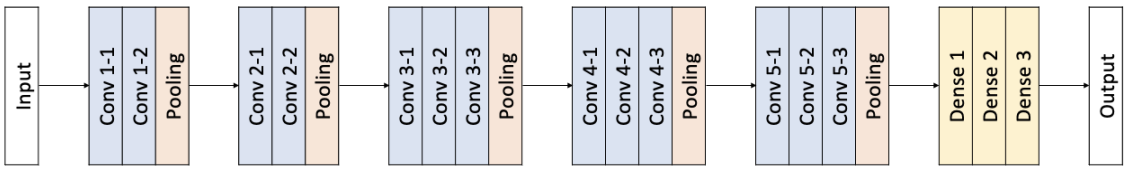

Figure 2-5: VGG-16 network architecture.

same image are) is 0.8. Therefore, any model that has an F_1 score around 0.8 is about as good as a human at detecting edges [31].

2.3.2 Visual Geometry Group

All of the ML models we implement are based on the Visual Geometry Group (VGG) CNN architecture. Originally designed to classify images, VGG has become one of the most famous architectures in the deep learning field. The architecture's success lies in its simplicity (largely convolutional layers with 3x3 kernels), its depth (16+ layers) and its generalizability to other ML tasks, even outside the realm of classification [45]. Figure 2-5 shows the original VGG-16 architecture.

Analyses of trained VGG models show that the convolutional layers are composed of small linear filters that create feature maps, which are useful in edge detection. The edge detection models detailed in 2.3.3 and 2.3.4 preserve the convolutional and pooling layers of this architecture [55, 57].

2.3.3 Holistically-Nested Edge Detection

The Holistically-Nested Edge Detection (HED) model adopts the five convolutional blocks of the VGG architecture, but adds a side-output layer at the end of each convolutional block. These five side output layers are intended to capture edges of varying scales in the image, and are fused together in one final convolutional layer to create the model output, as shown in Figure 2-6 [55].

HED is initialized with pre-trained VGG weights, then trains on the BSDS500 dataset. It outperforms existing ML and non-ML edge detection methods with an F_1

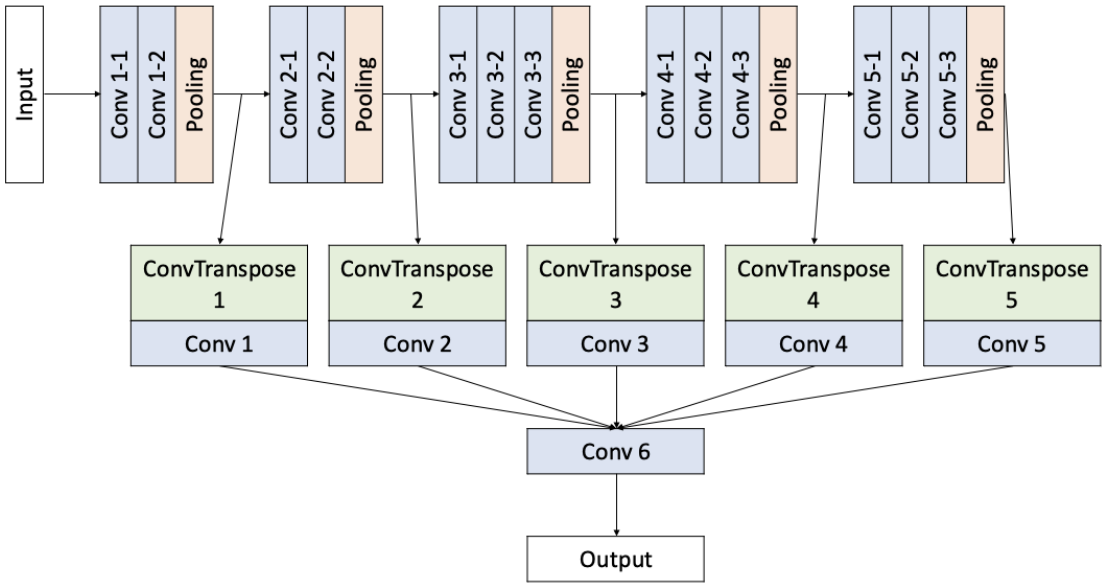

Figure 2-6: HED network architecture.

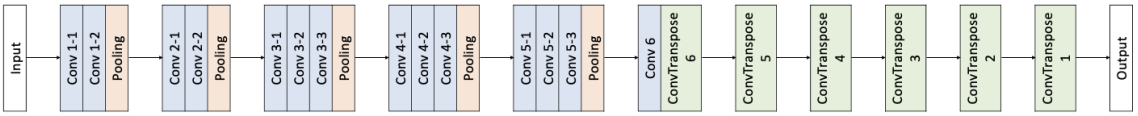

Figure 2-7: CEDN network architecture.

score of 0.79, and is notably more lightweight than previous edge detection architectures [55].

2.3.4 Convolutional Encoder-Decoder Network

The Convolutional Encoder-Decoder Network (CEDN) adopts the five convolutional blocks of of the VGG architecture, but adds a lightweight convolutional decoder to the end of the VGG encoder. This decoder is intended to capture the contours (the edges of objects) in the image, and the final decoding layer produces the model output, as shown in Figure 2-7 [57].

CEDN is initialized with pre-trained VGG weights, then trains on the BSDS500 dataset. It outperforms existing ML and non-ML edge detection methods with an F_1

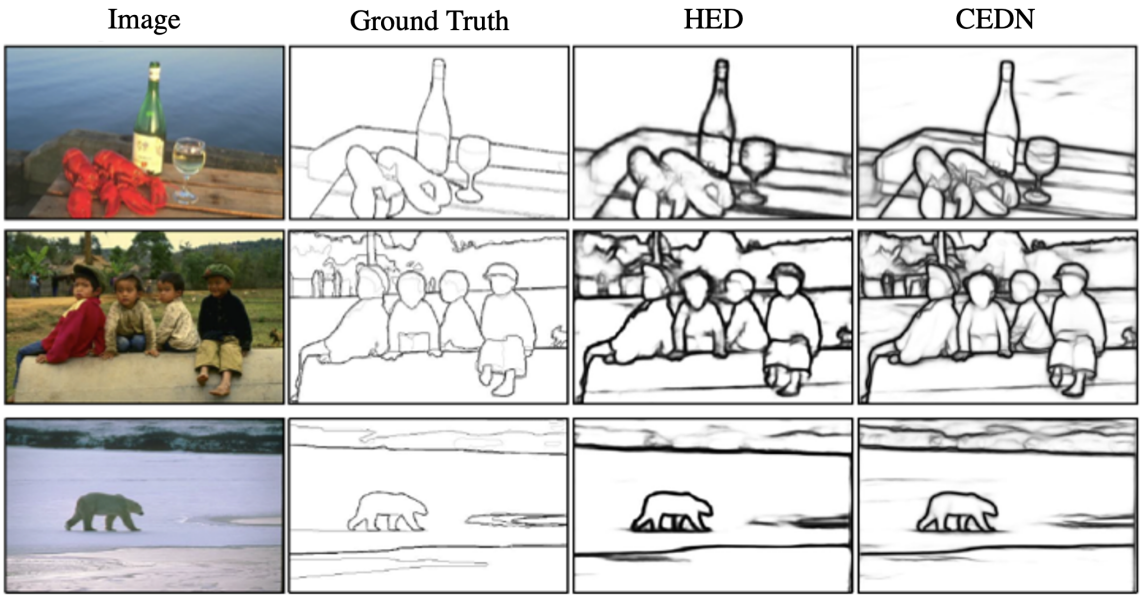

Figure 2-8: Sample HED and CEDN model predictions on images from the BSDS500 dataset. Source: [47]

score of 0.79, and generalizes notably well to other edge detection domains [57].

Figure 2-8 provides three examples of images from the BSDS500 dataset and their human annotated ground truths, as well as corresponding HED and CEDN model predictions.

2.4 Application: BeaverCube-2

A potential application for our ocean front detection models is BeaverCube-2 (BC2). BC2 is an Earth-imaging CubeSat designed by MIT's STAR Lab with guidance from Woods Hole Oceanographic Institution (WHOI), and sponsored by Northrop Grumman (NGC). Its primary mission is to demonstrate the computing capabilities of an AI Accelerator System on a Chip, which combines scalar and vector processing units with programmable logic for high-speed computations of ML models [3]. Its secondary mission is to obtain images of coastal waters for scientists at WHOI, who are studying the dynamics of small-scale ocean fronts in coastal waters. The BC2 mission parameters are an excellent match for our models, and BC2 is projected to

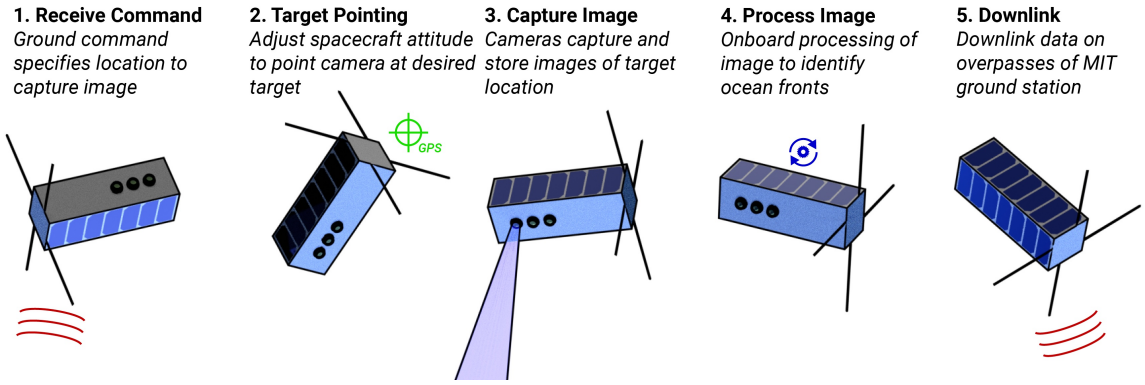

Figure 2-9: Concept of Operations for BeaverCube-2. Source: [22]

launch in 2023 [50].

2.4.1 Concept of Operations

BC2's concept of operations is detailed in Figure 2-9. First, BC2 receives a target location from the ground. Then, BC2 images the target location. Next, BC2 processes the image and computes relative scientific data, such as the locations of clouds and ocean fronts. Finally, if the image is deemed scientifically valuable, BC2 downlinks the image to Earth [23]. The ocean front detection models in this thesis fit under the "Process Image" phase of operations.

2.4.2 Cameras

BC2 has three cameras: one capturing thermal infrared (TIRS) wavelengths and two capturing visible (VIS) wavelengths. The TIRS camera is a Boson long-wave infrared (LWIR) camera, which uses an uncooled Vanadium Oxide microbolometer to measure infrared radiation. Both VIS cameras are Matrix Vision mvBlueFOX cameras, which are complementary metal-oxide-semiconductor cameras. One VIS camera has a monochrome filter on it to detect coastal/aerosol wavelengths [50]. The bands imaged by the cameras are detailed in Table 2.3. Note that BC2 images the wavelengths necessary to calculate CHL and SST at the spatial resolutions necessary to capture most small-scale ocean fronts.

Table 2.3: BeaverCube-2 spectral bands. Source: [50]

Camera	Band	Center Wavelength (micrometers)	Spatial Resolution (meters)	Image Size (pixels)
VIS + filter	Coastal Aerosol	0.44	150	960x1280
VIS	Blue	0.47	150	960x1280
VIS	Green	0.53	150	960x1280
VIS	Red	0.63	150	960x1280
TIR	Thermal Infrared (TIRS)	10.5	266	256x320

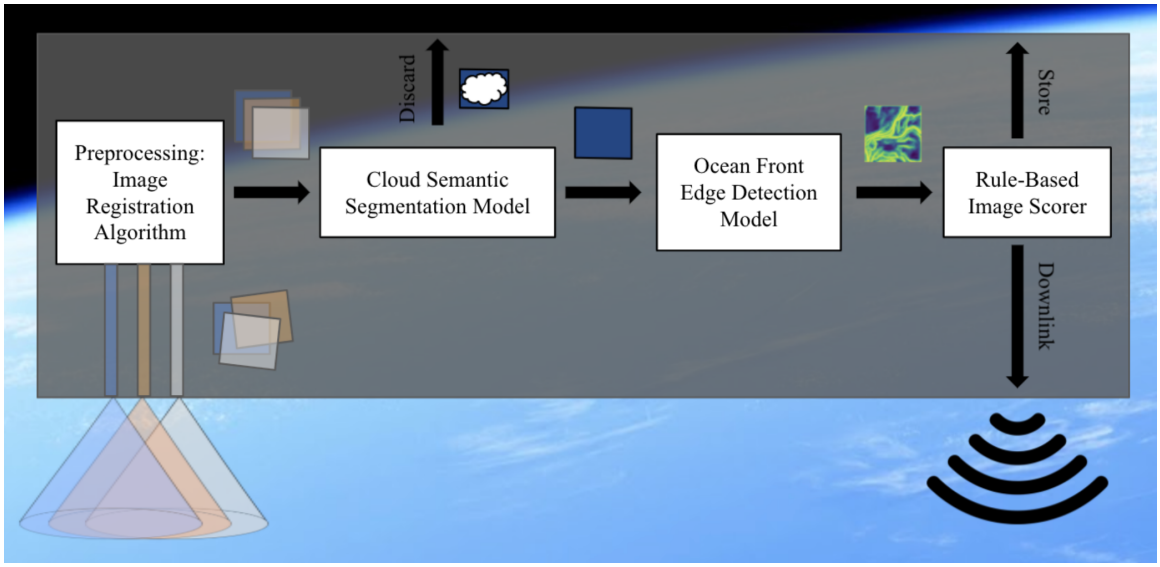

Figure 2-10: Image Processing Pipeline for BeaverCube-2.

2.4.3 Image Processing Pipeline

We develop an image processing pipeline demonstrating how each satellite image taken by BC2 will be processed before downlink. The four stages (represented by white rectangles in Figure 2-10) are detailed in the following sections. The ocean front detection models in this thesis fit under the "Ocean Front Detection" stage.

Image Registration

The first step is image registration: aligning the images taken by the three cameras into one coordinate plane. There are multiple existing strategies for automatic sub-pixel co-registration of satellite images [49]. There are also promising ML methods for automatic co-registration of medical images, although few have been applied to satellite imagery yet [14]. The image registration stage is outside the scope of this

thesis, and will be developed in future work.

Cloud Semantic Segmentation

The second step is cloud semantic segmentation: identifying clouds in registered images. Cloud identification is a well-established remote sensing problem, with the most recent research producing fairly accurate ML semantic segmentation models that designate each pixel as "cloud" or "not cloud" [36]. The cloud semantic segmentation stage is outside the scope of this thesis, and is being developed concurrently by fellow STAR Lab researchers [22].

Ocean Front Detection

The third step is ocean front detection: identifying ocean fronts in non-cloudy images. This problem is the primary focus of this thesis, and potential models for BC2 deployment are detailed in Chapters 2-4.

Image Scorer

The final step is image scoring: computing the scientific value of each image so it can be placed at the appropriate location in the downlink queue. The image scorer should be flexible, responding to priorities uplinked from the ground. The image scorer stage is outside of the scope of this thesis, and will be developed in future work.

Chapter 3

Analysis

3.1 Training Data

We download Landsat 8 scenes using Google Earth Engine. The JavaScript Earth Engine API and additional python preprocessing allow us to create scenes with the parameters listed in Table 3.1. Detailed code for this download process can be found in Appendix A.

We then select 225 of the highest-quality scenes from this set, emphasizing spatial variance as detailed in Figure 3-1. We ensure that we capture images of various frontal structures including large-scale water mass fronts, meso-scale fronts around eddies, and local chlorophyll blooms and ridges [6]. The 225 training scenes selected

Table 3.1: Parameters for Landsat training scenes.

Parameter	Value
Region of Interest	Ocean
Date Collected	2017-2021
Processing Levels	0,1,2
Data Quality	Tier 1
Cloud Cover	$<5\%$
Resolution	250m
Random Crop	224x224 pixels
Pixel Range	0-255

are listed in Appendix A.

We augment this set of 225 scenes by mirroring and rotating by 90, 180, and 270 degrees. We do not augment by scaling or cropping, as the 250 meter resolution and 224x224 pixel parameters approximate our BeaverCube-2 application [50]. This augmentation results in a final dataset of 1800 training scenes.

3.1.1 Bias in Training Data

We try to ensure that the distribution of frontal features in our training data reflects the distribution of frontal features in the oceans. This goal is hard to meet, because Landsat 8 predominantly images coastal ocean regions in first world countries [20], inherently biasing our training dataset towards those regions. Our intended application BeaverCube-2 will primarily image Cape Hatteras, which is a coastal ocean region in a first world country. However, for future applications, this bias in training data could result in decreased operational accuracy.

Additionally, because we select training scenes with little cloud cover and there is a relationship between atmospheric fronts and ocean fronts [15], our training data is biased towards detecting the types of fronts that appear in cloudless regions. Fortunately, our intended application BeaverCube-2 will also only detect fronts in cloud-free images, and there is no obvious future application where our models could be used to detect ocean fronts in cloud-heavy regions.

3.2 Ground Truth Data

We create ground truth ocean front data by processing Landsat Level 2 data using the NASA CHL and SST algorithms detailed in Section 2.2.1, then applying the Belkin-O'Reilly algorithm detailed in Section 2.2.2.

Figure 3-1: Locations of Landsat training data.

3.2.1 Thresholding Ocean Fronts

We threshold the Belkin-O'Reilly output, turning each gradient magnitude image into a binary image where pixels are labeled "ocean front" or "not ocean front." There are four main reasons behind this decision, which does result in a loss of ocean front intensity information:

1. ML edge detection models and other ocean front detection algorithms (like CCA) use binary images [9, 28, 30]; binarizing our ground truth makes it easier to use and compare to existing methods.

2. The BeaverCube-2 cloud detection model produces binary images; binarizing our ground truth aligns with this existing model and simplifies the job of the image scorer in the image processing pipeline.

3. Mild striping caused by inaccurate calibration of the Landsat pushbroom sensors is present in some training scenes (see Figure 3-2) [37]; binarizing our ground truth ensures the model does not think that these stripes are ocean fronts (because these stripes are below the ocean front binarization threshold).

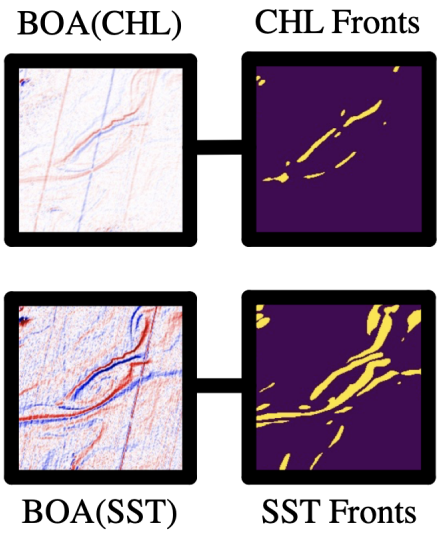

Figure 3-2: Thresholding applied to a BOA output. Note the mild striping in the BOA images, due to the Landsat pushbroom sensors. This image was constructed with a Landsat training scene, detailed in Appendix A.

4. We create human annotated binary ocean front images (described in Section 3.2.2) as an alternative ground truth; binarizing our Belkin O'Reilly ground truth makes it possible to compare these sources of ground truth data.

We select thresholds of 0.1°C/km for the SST ocean fronts and 0.05mg/m^3/km for the CHL ocean fronts. There are no universally established thresholds for deciding what constitutes an ocean front, but these thresholds are used in previous literature and are approved by consulting oceanographers [6, 32].

3.2.2 Human Annotated Ground Truth Data

In addition to creating the Belkin O'Reilly ground truth ocean front dataset, we use a subset of 46 training scenes (368 scenes after augmentation) to create human annotated ground truth ocean front data. Human subjects are given Level 0 Landsat inputs and Belkin O'Reilly un-thresholded outputs to assist in creating their binary output images.

There are three main reasons behind creating human annotated ground truth

data, which takes significant time and human resources.

1. ML edge detection models use human annotated data to train, because they are trying to make models that perform "better" (more human-like) than existing edge detection algorithms [31]. While not the primary goal, it would be advantageous if our models performed "better" than BOA.

2. The Belkin O'Reilly ground truth data is often noisy: detecting fronts only a few pixels wide or chopping fronts into disjointed sections. Humans have built-in image processing "filters" that intuitively detect long, smooth ocean fronts without worrying about numeric thresholds [33], as shown in Figure 3-3.

3. A set of human annotated data would allow us to do transfer learning–first training on the larger BOA ground truth dataset, then fine-tuning on the smaller human annotated ground truth dataset–to maximize model performance [12].

3.3 Model Architectures

The model architectures detailed in Sections 2.3.3 and 2.3.4 are state-of-the-art models for edge detection on a variety of natural images. However, our on-orbit ocean front detection task is narrower in scope and more SWaP-constrained than the previously developed edge detection models. We hypothesize that smaller models–which need less storage space and make faster inferences–can be as accurate as larger models for our application.

3.3.1 Reducing Model Breadth

While there are five spectral bands captured by BeaverCube-2, we use only three to detect ocean fronts: Coastal Aerosol, Green, and TIRS. It is not intuitive to remove potentially useful training data, but the NASA algorithm for computing SST uses only the TIRS band, and the NASA algorithm for computing CHL uses mostly the

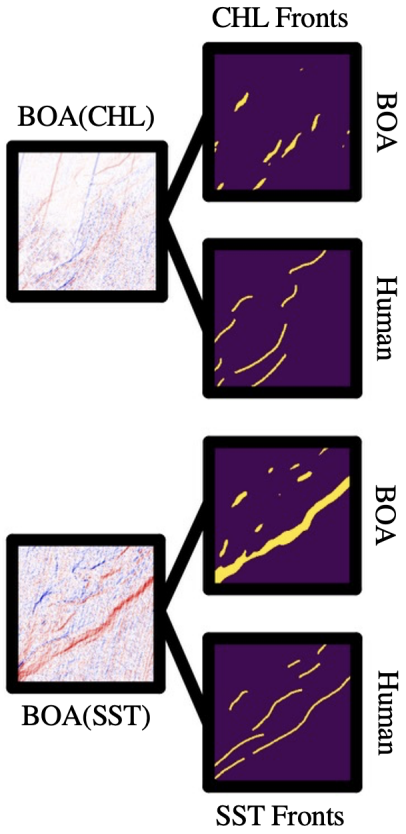

Figure 3-3: Comparing BOA ground truth and human ground truth. This image was constructed with a Landsat training scene, detailed in Appendix A.

Table 3.2: The impact that reducing model breadth has on the number of model parameters. Note that Large HED/Large CEDN are listed to show the impact of using all five BC2 input bands, but these models are not viable for BC2 deployment due to their large size, and thus are not explored further.

Model	Input Dimensions	Parameters (million)
Large HED	224x224x5	26.9
HED	224x224x3	16.1
Large CEDN	224x224x5	66.7
CEDN	224x224x3	40.0

Coastal Aerosol and Green bands (the Blue band is used 0.00001% of the time in creating our ground truth).

This three band input also aligns with standard computer vision models including VGG, HED, and CEDN, which accept RGB inputs [45, 55, 57]. By matching our input dimensions to theirs (the 224x224 pixel window was also selected for this reason), we can easily use pre-trained VGG model weights. Additionally, training with three bands instead of five means we have fewer model parameters (represented in Table 3.2); the model needs less storage space and can make faster inferences.

3.3.2 Reducing Model Depth

Edge detection happens in the first few layers of a computer vision model [45]. Therefore, it is feasible that decreasing the number of layers in our models could preserve the edge detection capabilities of the models, while reducing the storage space and inference time. We experiment with a smaller version of the HED model (Figure 3-4), and a smaller version of the CEDN model (Figure 3-5).

Training with a shallower network means we have fewer model parameters (represented in Table 3.3); the model needs less storage space and can make faster inferences.

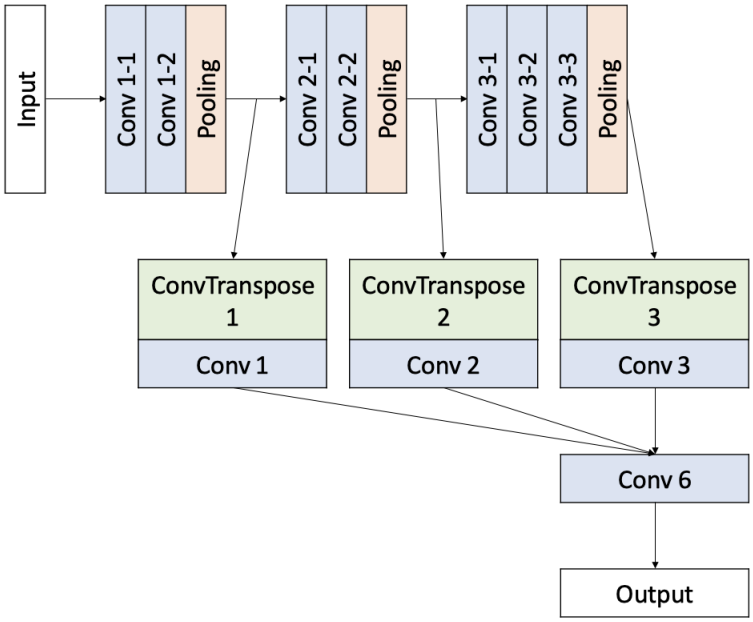

Figure 3-4: A smaller version of the HED network architecture.

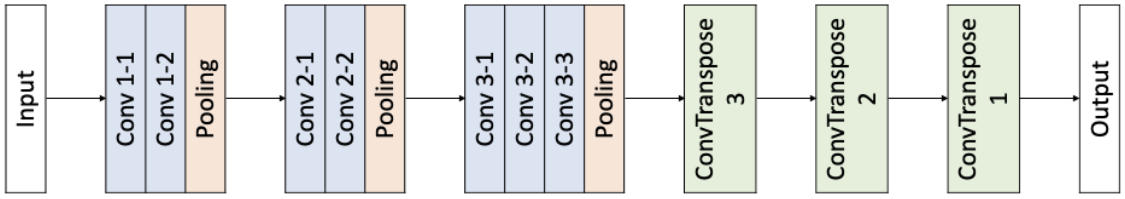

Figure 3-5: A smaller version of the CEDN network architecture.

Table 3.3: The impact that reducing model depth has on the number of model parameters.

Model	Architecture	Parameters (million)
HED	Figure 2-6	16.1
Small HED	Figure 3-4	1.9
CEDN	Figure 2-7	40.0
Small CEDN	Figure 3-5	2.7

3.4 Training Decisions

3.4.1 Focal Cross Entropy Loss

A loss function is a measure of how good a model is at predicting what it's supposed to predict, and is used in training a model. Selecting a good loss function is important, as it can change how well a model learns patterns in the data.

Our goal is to determine which pixels in an image are ocean front pixels, and which are not. This is a pixel-wise classification problem, where one class is "ocean front" and one class is "not ocean front."

A standard loss function for this type of task is binary cross entropy (BCE) loss. However, BCE performs best when both classes have an equal number of pixels. In our dataset, only 6% of pixels are labeled "ocean front."

To combat this class imbalance, we use focal cross entropy (FCE) loss. This loss function weights examples based on class imbalance, and also dynamically assigns more weight to hard-to-classify examples and less weight to easy-to-classify examples [29].

If y is the ground truth $y \in \{0, 1\}$ and p is the predicted probability $p \in [0, 1]$ then pixel-wise BCE and FCE are defined as:

$$BCE = -(y * log(p) + (1 - y) * log(1 - p))$$

$$FCE = -(\alpha * (1 - p)^{\gamma} * y * log(p) + (1 - \alpha) * p^{\gamma} * (1 - y) * log(1 - p))$$

Note that FCE has the same structure as BCE, but adds two additional scalars: α is a weighting factor based on the number of examples in each class, and γ is a focusing parameter designed to down-weight easily-classified examples [29]. For our application, $\alpha = 0.25$ and $\gamma = 2$.

Table 3.4: Parameters for training.

Parameter	Value
Train:Test:Val	80:10:10
Pre-Trained Weights	VGG
Batch Size	10
Epochs	100
Early Stopping	10
Loss Function	FCE
Optimizer	Adam
Learning Rate	1e-4

3.4.2 Training Parameters

To prepare our data for training, we subtract the per-channel mean pixel values calculated on the training dataset (the VGG preprocessing step [45]) and shuffle the data. We build four models in TensorFlow [4]: HED, Small HED, CEDN, and Small CEDN. Detailed code for these model architectures can be found in Appendix C.

We train these models on the MIT SuperCloud [40] using the parameters detailed in Table 3.4, first training on the BOA ground truth dataset then fine-tuning on the human annotated ground truth dataset.

Chapter 4

Results

4.1 Comparing Model Performance

We analyze the performance of our four model architectures across two testing datasets using qualitative and quantitative metrics. Qualitatively, we visually compare model outputs to ground truth data. Quantitatively, we calculate binary metrics, F_1 scores, and a precision-recall curve for each model.

All models produce "soft" outputs, meaning every pixel has a value between 0 and 1 representing the probability that pixel is an ocean front. Thresholding these outputs at a certain probability produces "hard" outputs, where every pixel is 0 (not an ocean front) or 1 (an ocean front). This allows each pixel to be assigned a value of true positive (the model predicts it is an ocean front and it is in fact an ocean front), false positive (the model predicts it is an ocean front but it is not in fact an ocean front), true negative (the model predicts it is not an ocean front and it is not in fact an ocean front), or false negative (the model predicts it is not an ocean front but it is in fact an ocean front).

We use four binary metrics to compare models: accuracy, precision, specificity, and recall. These performance metrics are based on the number of true positive (tp), false positive (fp), true negative (tn), and false negative (fn) pixels in each prediction. For a perfect model, these metrics would all be 1.0.

$$accuracy = \frac{tp + tn}{tp + fp + tn + fn}$$

$$precision = \frac{tp}{tp + fp}$$

$$specificity = \frac{tn}{tn + fp}$$

$$recall = \frac{tn}{tn + fp}$$

Changing the binarizing threshold changes the values of the binary metrics. We calculate the F_1 score for each threshold in the range [0.05,0.1,0.15...0.85,0.9,0.95], and select the maximal F_1 score to determine the "best" threshold for model performance. The F_1 score balances precision and recall. High precision ensures that the model only identifies ocean front pixels as ocean front pixels, while high recall ensures that the model identifies all ocean front pixels as ocean front pixels. For a perfect model, the maximal F_1 score would be 1.0.

$$F_1 = 2 \cdot \frac{precision \cdot recall}{precision + recall} = \frac{tp}{tp + \frac{1}{2}(fp + fn)}$$

We also plot precision vs. recall over the full range of thresholds and calculate the area under the precision-recall curve. These metrics provide more broad information about model performance and stability. For a perfect model, the precision-recall curve would look like a horizontal line at precision=1 for all values of recall, and the area under the curve (AUC) would be 1.0.

4.1.1 Training with Belkin O'Reilly Data

Figure 4-1 provides four examples of model input/output, displaying model performance across a variety of frontal types and intensities. The far left column is the Level 0 input data. Note the Green channel is also a model input, but is excluded from

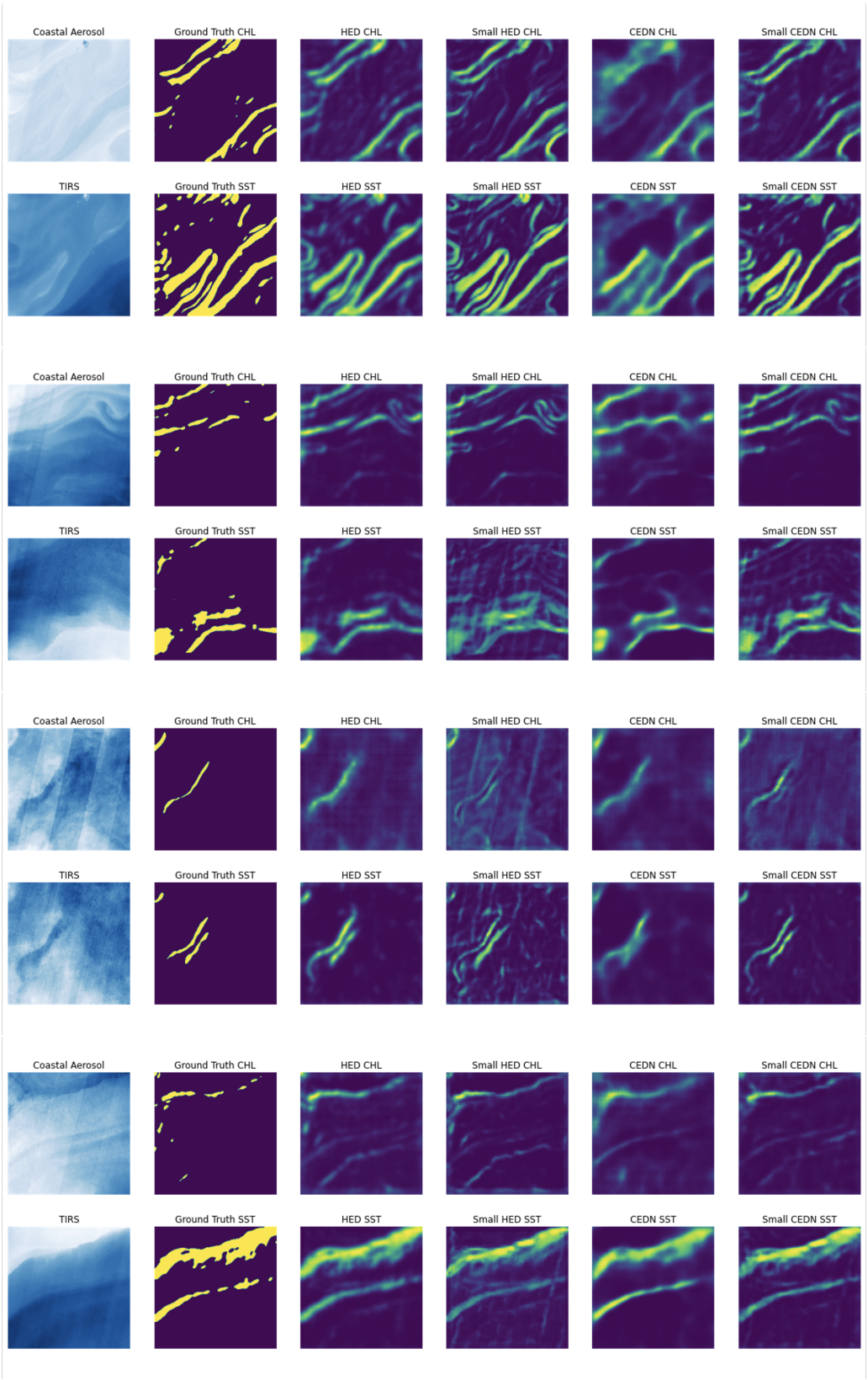

Figure 4-1: Sample HED, Small HED, CEDN, and Small CEDN model predictions on images from the BOA ground truth dataset.

this diagram for simplicity (it is visually similar to the Coastal Aerosol channel). The next column is the thresholded BOA ground truth data. The final four columns are unthresholded outputs from our four models: HED, Small HED, CEDN, and Small CEDN.

Qualitative Analysis

All four models do a reasonable job of looking visually similar to the ground truth.

Note that Small HED captures the mild striping caused by inaccurate calibration of the Landsat pushbroom sensors [37]. This is not present in the ground truth data, but also not a cause for great concern, as the BeaverCube-2 cameras do not have pushbroom sensors and consequently will not capture images with this striping [50].

Occasionally, HED has a checkered pattern in its output. This is particularly noticeable in the third input/output CHL prediction. This pattern implies that there is some instability in the model, emerging especially when there are few fronts in the image.

CEDN is blurrier than the other outputs, failing to delineate between separate fronts in the first input/output CHL and third input/output SST prediction. This fuzziness is due to the small size of the center convolutional layer in the encoder-decoder architecture, and could potentially be mitigated by adding skip connections across the encoding and decoding layers [57], but that is outside the scope of this thesis.

Qualitatively, Small HED captures the most frontal detail, followed by Small CEDN, followed by HED, followed by CEDN. This order is inversely related to the number of parameters in each model, supporting our hypothesis that the first few layers in the model are the ones most important for ocean front detection.

It is also important to acknowledge that the BOA ground truth data is not perfect, a clear example of which is shown in Figure 4-2. The ground truth only captures parts of the swirls evident in the input data. In contrast, most models capture all of the swirls evident in the input data.

This emphasizes the importance of our pretrained VGG weights; the models al-

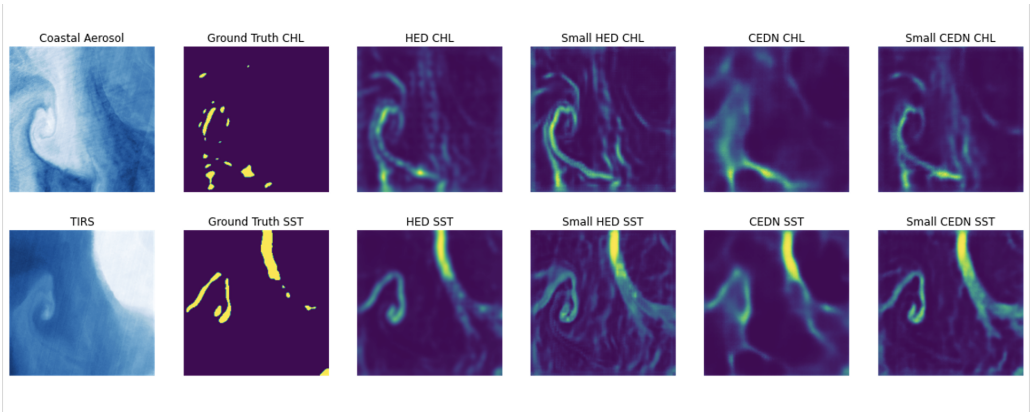

Figure 4-2: Sample HED, Small HED, CEDN, and Small CEDN model predictions on an image with a poor BOA ground truth.

ready have some robust concept of edge detection that only needs to be refined for ocean front detection. The disparity between ground truth data and model predictions–where the models seem to outperform the ground truth–underscores the generalization capabilities of the models.

However, this disparity also provides a cautious warning for our quantitative metrics. These numbers only reflect the models' abilities to match the BOA ground truth data, an imperfect approximation for the true ocean fronts in each input.

Quantitative Metrics for CHL

The precision-recall curve for CHL is shown in Figure 4-3 and describes model performance across all thresholds. HED has the best performance across the board (and the highest AUC), followed by Small CEDN, Small HED, and CEDN. The smooth curves show a stable trade-off between precision and recall.

The best threshold for each model and the corresponding F_1 score and binary metrics are detailed in Table 4.1. Best thresholds fluctuate between 0.3 and 0.35. HED performs better than all other models across the board, especially in the category of precision, meaning HED is better at identifying only "ocean front" pixels as "ocean front." This aligns with our qualitative analysis: HED captures the main ocean fronts but not smaller details, which matches our ground truth that captures the main ocean

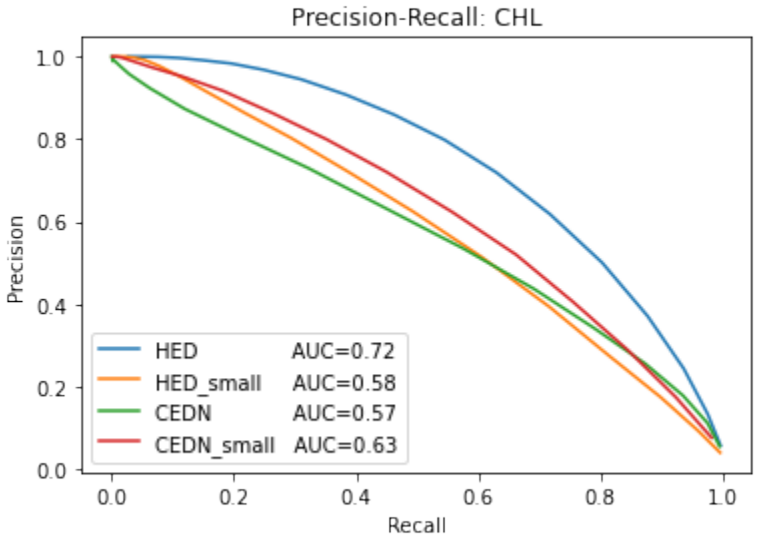

Figure 4-3: CHL precision-recall curve for models trained with BOA ground truth data.

Table 4.1: CHL metrics for models trained with BOA ground truth data. The bolding highlights the highest score for each metric.

Model	Best Threshold	F_1 score	Accuracy	Precision	Specificity	Recall
HED	0.35	**0.67**	**0.99**	**0.72**	**0.99**	**0.63**
Small HED	0.3	0.56	0.98	0.51	0.99	0.61
CEDN	0.35	0.55	0.98	0.54	0.99	0.57
Small CEDN	0.3	0.59	0.98	0.62	0.99	0.56

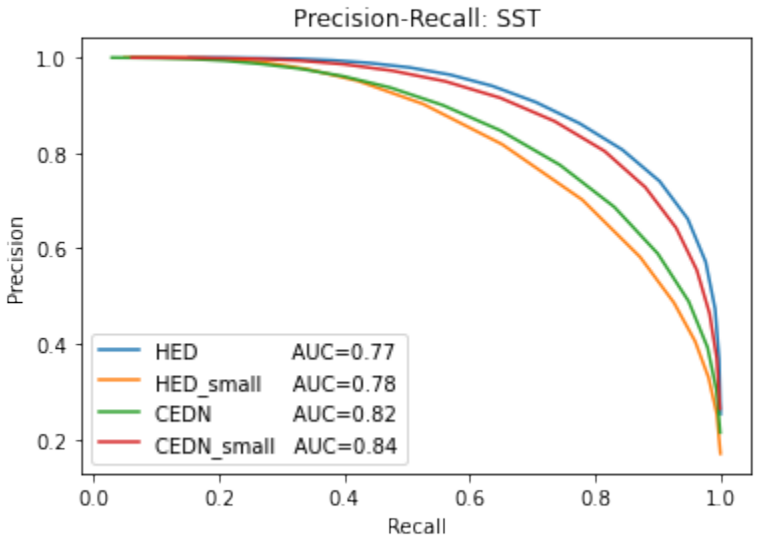

Figure 4-4: SST precision-recall curve for models trained with BOA ground truth data.

fronts but not smaller details.

Note the high accuracy and specificity across all models. While numerically impressive, these numbers are not very descriptive of model performance. High accuracy ensures the model identifies all pixels correctly, but in our application this number is dominated by the "not ocean front" pixels being classified correctly. High specificity ensures that the model identifies all "not ocean front" pixels as "not ocean front" pixels, which is secondary to our goal of detecting ocean front pixels.

Quantitative Metrics for SST

The precision-recall curve for SST is shown in Figure 4-4 and describes model performance across all thresholds. HED and Small CEDN have the best performance across the board (while the HED curve is slightly higher, Small CEDN has a higher AUC), followed by CEDN and Small HED. The smooth curves again show a stable trade-off between precision and recall.

The best threshold for each model and the corresponding F_1 score and binary metrics are detailed in Table 4.2. The best threshold is consistent at 0.35. HED again performs well across the board, but is less of an obvious front-runner in SST front

Table 4.2: SST metrics for models trained with BOA ground truth data. The bolding highlights the highest score for each metric.

Model	Best Threshold	F_1 score	Accuracy	Precision	Specificity	Recall
HED	0.35	**0.82**	**0.96**	**0.81**	0.97	**0.84**
Small HED	0.35	0.74	0.94	0.70	0.96	0.78
CEDN	0.35	0.76	0.95	0.77	0.97	0.74
Small CEDN	0.35	0.81	0.96	0.80	**0.98**	0.81

detection than in CHL front detection. Small CEDN surpasses HED in specificity, meaning Small CEDN is better at identifying all "not ocean front" pixels as "not ocean front." This aligns with our qualitative analysis: Small CEDN outputs have thinner and sharper fronts than HED outputs, resulting in more "not ocean front" pixels.

Note the F_1 scores for SST front detection average in the high 0.7s, comparable to the F_1 scores for edge detection (both HED and CEDN score 0.79 on the BSDS500 dataset). Although the theoretical maximum for an F_1 score is 1.0, the experimental maximum for an F_1 score in our application is around 0.8, which represents human levels of accuracy [31].

4.1.2 Fine-Tuning with Human Annotated Data

Now we take our four models trained on the Belkin-O'Reilly ground truth dataset and fine-tune them on the human annotated ground truth dataset. Figure 4-5 provides four examples of model input/output, displaying model performance across a variety of frontal types and intensities. The far left column is the Level 0 input data. Note the Green channel is also a model input, but is excluded from this diagram for simplicity (it is visually similar to the Coastal Aerosol channel). The next column is the human annotated ground truth data. The final four columns are unthresholded outputs from our four models: HED, Small HED, CEDN, and Small CEDN.

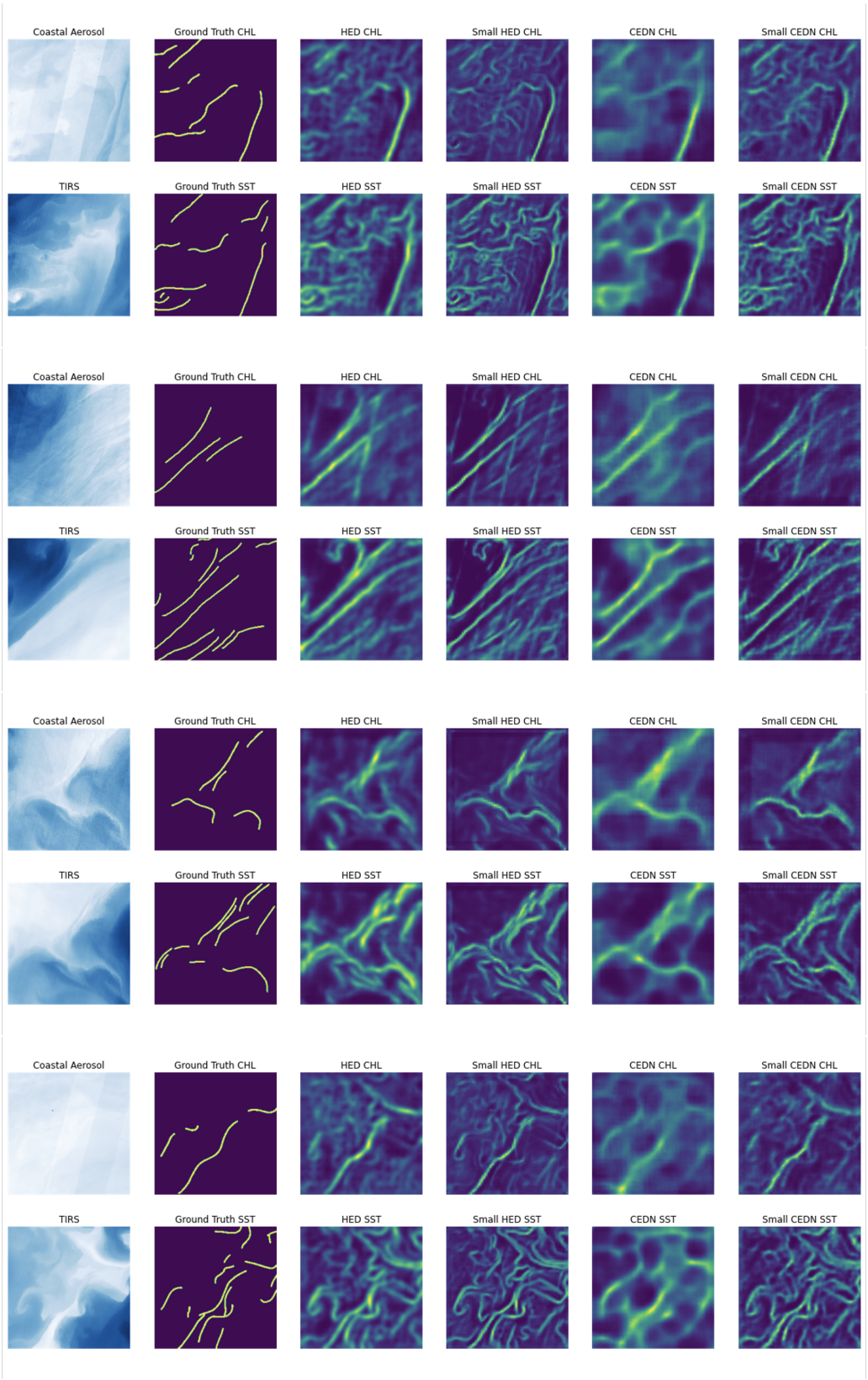

Figure 4-5: Sample HED, Small HED, CEDN, and Small CEDN model predictions on images from the human annnotated ground truth dataset.

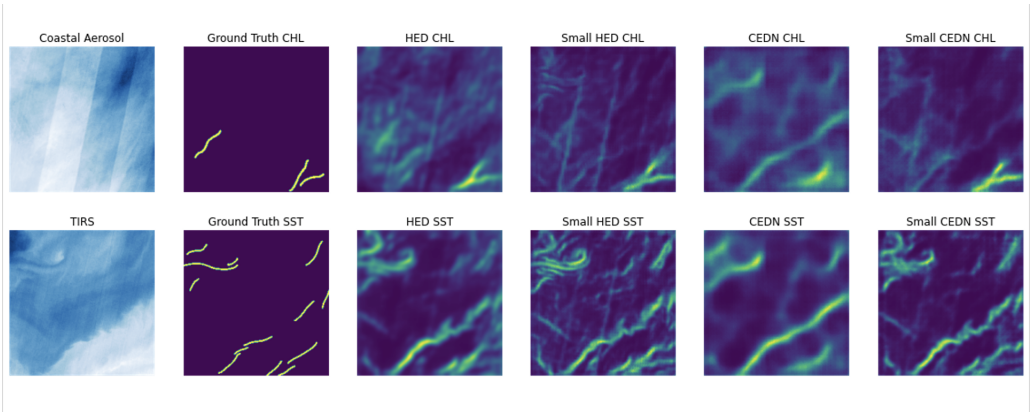

Figure 4-6: Sample HED, Small HED, CEDN, and Small CEDN model predictions on an image with a poor human annotated ground truth.

Qualitative Analysis

All four models still do a reasonable job of looking visually similar to the ground truth.

Small HED continues to capture the mild striping caused by inaccurate Landsat sensor calibration [37], but now the striping is also occasionally visible in the Small CEDN and HED outputs (see the second input/output CHL prediction).

Occasionally, SST front predictions "bleed" into CHL front predictions. This is best captured by CEDN in the second input/output, where the CHL prediction has a front in the bottom right corner that does not show up in the ground truth or other model CHL predictions, but does show up in the SST prediction, discussed further in Section 4.1.3.

All of the outputs contain more fine detail now, which makes intuitive sense as the human annotated ground truth data captures fronts of lower intensity compared to the BOA ground truth data. This fine detail could be useful or detrimental depending on the scientific application. Qualitatively Small HED still captures the most frontal detail, followed by Small CEDN, HED, and finally CEDN.

It is important to remember that our human annotated ground truth data is still not perfect. The intent of creating human annotated ground truth data was to produce images with long, smooth ocean fronts that can be captured by the human

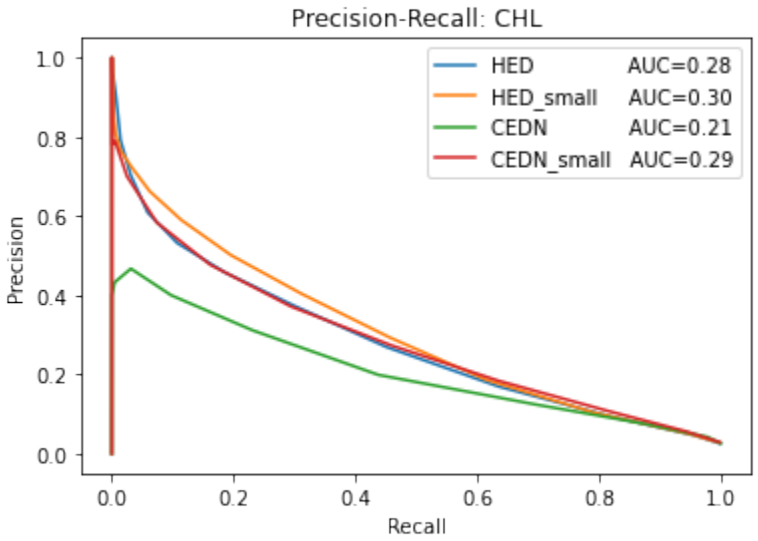

Figure 4-7: CHL precision-recall curve for models fine-tuned with human annotated ground truth data.

eye but not by BOA. While the human annotated ground truth fronts are longer and smoother than the BOA ground truth fronts, there is still room for improvement. See Figure 4-6, where there is a long, smooth-ish ocean front in the bottom right corner of the TIRS input and all model predictions, but a series of short, jagged fronts in the human annotated ground truth. This problem could be mitigated by increasing the number of human annotators per image from n=1 to n=3 (the number of annotators per image in the BSDS500 dataset) and averaging the annotations [31], but that is outside the scope of this thesis.

Figure 4-6 also provides a strong example of CEDN SST front predictions "bleeding" into CHL front predictions. The SST fronts in the top left and the bottom right of the CEDN SST prediction also appear in the CEDN CHL prediction, yet they do not appear in the CHL input, ground truth, or any other model's CHL prediction.

Quantitative Metrics for CHL

The precision-recall curve for CHL is shown in Figure 4-7 and describes model performance across all thresholds. Small HED has the best performance across the board (and highest AUC), followed quickly by Small CEDN and HED, and eventually by

Table 4.3: CHL metrics for models fine-tuned with human annotated ground truth data. The bolding highlights the highest score for each metric.

Model	Best Threshold	F_1 score	Accuracy	Precision	Specificity	Recall
HED	0.25	0.34	0.96	0.27	0.97	0.45
Small HED	0.25	**0.36**	**0.96**	**0.29**	**0.98**	0.46
CEDN	0.25	0.27	0.95	0.20	0.96	0.44
Small CEDN	0.25	0.34	0.96	0.27	0.97	**0.46**

CEDN. The sharp drop-off in precision shows the models are predicting many pixels are ocean fronts, when these pixels are not represented as ocean fronts in the ground truth data.

The best threshold for each model and the corresponding F_1 score and binary metrics are detailed in Table 4.3. The best threshold is consistent at 0.25. Small HED performs well across the board, but is slightly eclipsed by Small CEDN in recall, meaning Small CEDN is better at identifying all "ocean front" pixels as "ocean front." This matches our intuitive understanding–the ocean fronts predicted by Small CEDN are slightly blurrier than the ocean fronts predicted by Small HED (and therefore more pixels are likely to be labeled as "ocean front").

The metrics in Figure 4-7 and Table 4.3 are the lowest metrics in this body of work, meaning detecting chlorophyll fronts after human annotated fine-tuning is the task that the models perform worst at. However, it is important to put these numbers into perspective.

The "baseline" precision-recall curve for this task is a horizontal line equal to the number of ocean front pixels over the total number of pixels, which in our case is 0.02. This results in a "baseline" AUC of 0.02.

Similarly, the "baseline" binary metrics can be calculated by predicting every pixel as "not ocean front" (F_1 score=0.0, accuracy=0.98, precision=0.0, specificity=1.0, recall=1.0), or every pixel as ocean front (F_1 score=0.04, accuracy=0.02, precision=0.02, specificity=0.0, recall=0.0).

In comparison to these baselines, our metrics–particularly AUC, F_1 score, and

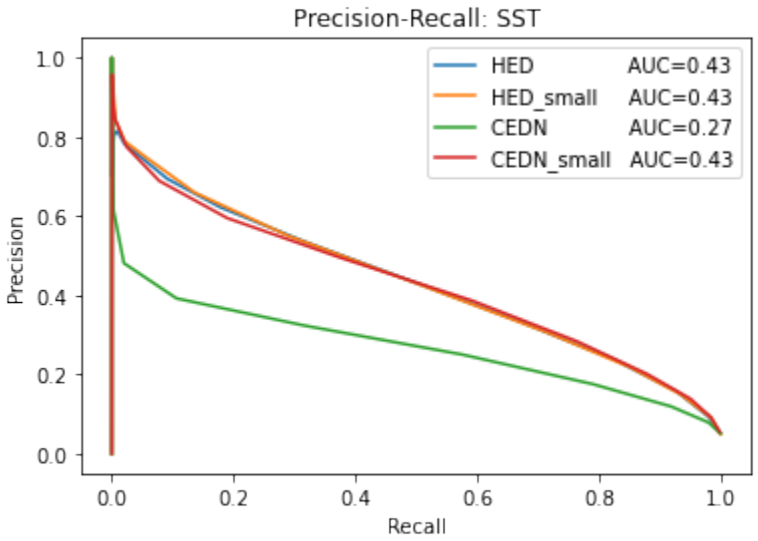

Figure 4-8: SST precision-recall curve for models fine-tuned with human annotated ground truth data.

Table 4.4: SST metrics for models fine-tuned with human annotated ground truth data. The bolding highlights the highest score for each metric.

Model	Best Threshold	F_1 score	Accuracy	Precision	Specificity	Recall
HED	0.3	0.46	**0.96**	**0.44**	**0.98**	0.49
Small HED	0.3	0.46	0.96	0.42	0.98	0.52
CEDN	0.25	0.35	0.93	0.25	0.94	0.58
Small CEDN	0.3	**0.47**	0.96	0.39	0.97	**0.59**

precision–are good.

Quantitative Metrics for SST

The precision-recall curve for SST is shown in Figure 4-8 and describes model performance across all thresholds. HED, Small HED, and Small CEDN perform pretty much equally across the board (and in terms of AUC), followed by CEDN. The sharp drop-off in precision again shows that the models are predicting many "ocean front" pixels, when these pixels are not represented as "ocean front" in the ground truth data.

The best threshold for each model and the corresponding F_1 score and binary metrics are detailed in Table 4.4. The best threshold is 0.3 for all models except CEDN, where it is 0.25. HED and Small CEDN are the highest performers, with HED having a particular high precision (HED is better at only identifying "ocean front" pixels as "ocean front") and Small CEDN having a particularly high recall (Small CEDN is better at identifying all "ocean front" pixels as "ocean front"). This matches our qualitative understanding: the human annotated ground truth captures long smooth fronts of varying intensity, and HED accurately identifies the intense bits of these fronts while Small CEDN identifies much more (too much more) of the detailed bits of these fronts.

With this final set of metrics, we can draw some general conclusions about the comparative performance of our four model architectures across our two testing datasets:

- Models are generally better at capturing SST fronts than CHL fronts (discussed further in Section 4.1.3)
- Models are generally better at producing predictions that align with BOA than predictions that align with human annotation
- The models that capture intermediate levels of frontal detail (HED, Small CEDN) are the most robust performers across all predictions

4.1.3 Comparing CHL and SST Predictions

Overall, the models are quantitatively better at detecting SST fronts than CHL fronts. There are a couple of reasons why this could be the case:

1. In our ground truth datasets where only 6% of pixels are ocean fronts, only 16% of these ocean front pixels are CHL ocean fronts. This could be because our CHL ocean front threshold value is too high, or just reflect the fact that there are fewer CHL fronts than SST fronts in the oceanic regions our training data comes from. Either way, the models have very few examples of what constitutes a CHL ocean front, so they do not learn how to detect them very well.

2. The existing processing pipeline from input Coastal Aerosol/Green data to output CHL fronts is more complex than the existing processing pipeline from input TIRS data to output SST fronts. For example, Landsat Level 2 processing includes correcting for the scattering and absorbing effects of aerosols, which affect the Coastal Aerosol band more than the TIRS band [58]. Furthermore, the NASA algorithm for computing chlorophyll concentration uses a fourth-order polynomial and three Landsat bands [18], more elaborate than the NASA algorithm for computing sea surface temperature which uses a linear equation and one Landsat band [26]. This additional complexity could make it harder for the models to learn to detect CHL ocean fronts.

3. There is more developed literature around SST measurements and SST fronts than CHL measurements and CHL fronts [6, 52], so the processes surrounding front detection could work better for SST than CHL. While the unique features of CHL fronts are specifically targeted by the contextual filter in BOA [7], there could be other parts of the processing pipeline–such as the satellite instruments–that are better optimized for SST [20].

It is likely a combination of all of these factors and more that cause the discrepancy between SST front prediction quality and CHL front prediction quality.

Quantifying Similarity between CHL and SST Predictions

As noted in Section 4.1.2, there is a phenomenon in which SST fronts occasionally "bleed" into CHL fronts, most noticeably in fine-tuned CEDN predictions. We quantify this discovery by plotting the relationship between model input and output similarity. More formally, we compute the similarity between the Coastal Aerosol and TIRS band inputs using structural similarity (SSIM), a complex perception-based metric used for measuring the correlation between two images [54]. We then compute the similarity between unthresholded CHL and SST ocean front outputs for each fine-tuned model using this same metric.

The results of this computation on the test dataset are shown in Figure 4-9.

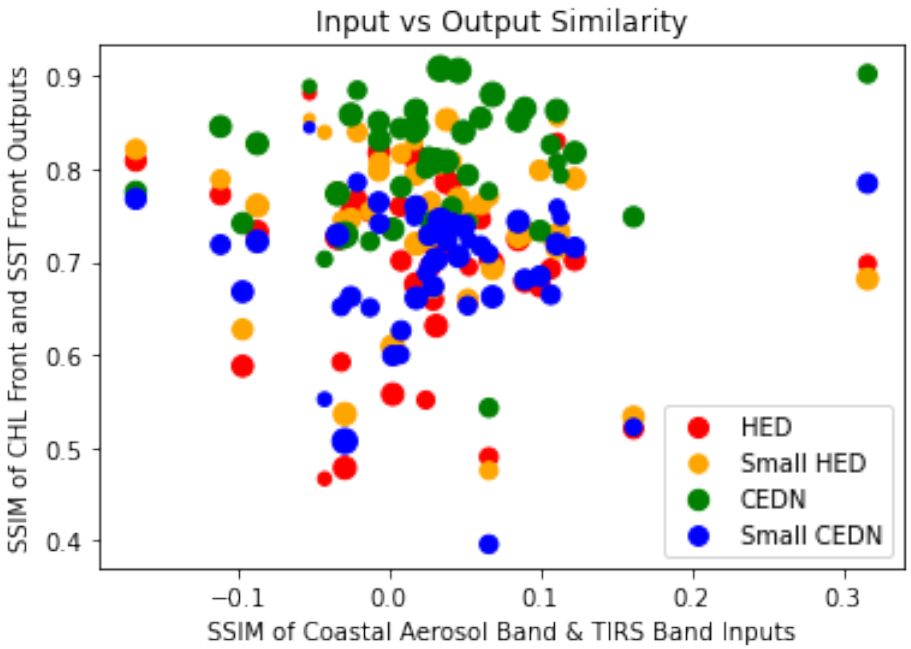

Figure 4-9: Structural similarity between fine-tuned model inputs and outputs. The color of each point corresponds to the model used to make the prediction, while the size of each point corresponds to the number of ocean front pixels in the prediction.

First, it is interesting to analyze the SSIM between the Coastal Aerosol and TIRS inputs. Positive SSIM means the bands are correlated (where temperature increases, chlorophyll concentration increases) while negative SSIM means the bands are anti-correlated (where temperature increases, chlorophyll concentration decreases). Zero SSIM means there is no correlation between the bands. Qualitatively we can see there are numerous inputs where the temperature and chlorophyll concentration are related (directly or inversely) and numerous inputs where the temperature and chlorophyll concentration are not related. Quantitatively, the average SSIM between the input bands is 0.0, and the standard deviation of SSIM between the input bands is 0.1.

In comparing input SSIM to output SSIM, we expect to see a roughly parabolic shape. Where the input bands are highly correlated (SSIM very positive or very negative), the output bands should be highly correlated (meaning the CHL and SST ocean front pixels are aligned). Where the input bands are not correlated (SSIM around zero), the output bands should be less correlated (but not zero; remember the

majority of our predicted pixels are still "not ocean front").

However, there are no parabolas that fit the data well ($R^2 < 0.05$ for each model). Qualitatively each set of model outputs looks like a loose grouping. Quantitatively, CEDN outputs have the highest average SSIM (0.81), followed by Small HED (0.74), then HED (0.70), then Small CEDN (0.69). This aligns with our earlier observation that CEDN CHL predictions noticeably "bleed" into SST predictions, creating a spurious correlation between CHL fronts and SST fronts and resulting in the highest average SSIM.

To avoid this "bleeding", we could train two separate ML models: one to detect CHL ocean fronts and one to detect SST ocean fronts. However, this would double the on-orbit computational resources needed to detect ocean fronts (discussed in Section 4.2), and is outside the scope of this thesis.

Note that we use the human annotated ground truth testing dataset for this analysis, but the same patterns exist (though they are less obvious) in the BOA ground truth testing dataset: the SSIM between the input bands ranges from -0.2 to 0.3, qualitatively each set of model outputs looks like a loose grouping, and the SSIM between the output bands is highest for CEDN.

4.2 Comparing Resource Utilization

We also analyze the performance of our four model architectures by computing the speed of inference, as well as the storage requirements of inference. Finally, we compare these metrics to the BeaverCube-2 CubeSat constraints.

4.2.1 Speed of Inference

We calculate the speed of inference for each model using Google Colab. Google Colab offers three types of computational resources at runtime: Central Processing Units (CPUs), Graphics Processing Units (GPUs), and Tensor Processing Units (TPUs). The default processors are CPUs (Intel(R) Xeon(R)), which are designed to be efficient for general-purpose programming. The hardware accelerator GPUs (Nvidia

Table 4.5: Model inference speeds using CPU, GPU, and TPU backends.

Model	CPU (sec)	GPU (sec)	TPU (sec)
HED	0.11	0.006	0.11
Small HED	0.05	0.002	0.07
CEDN	0.08	0.003	0.07
Small CEDN	0.05	0.001	0.05

K80) are designed to be efficient for computer graphics and AI computations (such as ML model inference), while the hardware accelerator TPUs (Google Cloud) are designed to be efficient for training ML models [53].

The inference speeds for each model are shown in Table 4.5. These times represent the average time for one inference, computed over 2000 inferences. Note that GPU speeds are on average 30 times faster than CPU speeds, while TPU speeds are about the same as CPU speeds. The strong performance of GPU makes sense, as GPUs are better suited for ML model inference.

Small CEDN performs the fastest inferences, followed by Small HED, then CEDN, then HED. This roughly matches the number of parameters in each model, but the HED and Small HED models perform slower than expected. This could be explained by the concatenation layer in the HED and Small HED architectures, a non-standard ML model layer for which the processing units may not be optimized [53].

4.2.2 Storage Requirements

We calculate the storage requirements for each model by converting our TensorFlow models to TensorFlow Lite models. TensorFlow Lite models are quantized TensorFlow models, optimized for efficient inference in SWaP-constrained embedded systems [44].

The space required for storing models and performing inference with models is detailed in Table 4.6. The memory needed for storage corresponds to when the model is not performing inference, while the memory needed for inference corresponds to when the model is performing inference.

The memory required for storage aligns with the number of parameters in each

Table 4.6: Storage metrics for TensorFlow Lite models.

Model	Memory - Storage (MB)	Memory - Inference (MB)
HED	64.7	352.63
Small HED	7.78	119.57
CEDN	160	354.38
Small CEDN	11.1	70.13

model, but the memory required for inference is more interesting. Small CEDN requires the least space, followed by Small HED, then HED and CEDN. This could perhaps also be explained by the non-standard concatenation layer in HED and Small HED, for which TensorFlow Lite may not be optimized [44].

4.2.3 BeaverCube-2 SWaP Constraints

BeaverCube-2 will cycle between 5 modes: Nominal, Eclipse, Communication, Science, and Navigation. The significant payload mode is the Science mode, in which images are taken and processed by the image processing pipeline detailed in Section 2.4.3. During Science mode, the AI Accelerator System on a Chip can use up to 4 Gb of flash memory to perform a maximum 30 second run of the image processing pipeline [3].

Although we have not deployed our models onto the AI Accelerator System on a Chip and the image processing pipeline is not completely built, the general runtimes and memory requirements of our ocean front detection models are a good indicator that the image processing pipeline will likely not exceed BeaverCube-2 resource constraints.

THIS PAGE INTENTIONALLY LEFT BLANK

Chapter 5

Summary and Future Work

5.1 Summary

In this thesis, we present machine learning models for detecting temperature and chlorophyll ocean fronts from satellite imagery. These models can be deployed on a CubeSat to solve the existing spatial, temporal, and computational challenges of detecting ocean fronts.

We discuss the importance of detecting ocean fronts and emerging tools for doing so more effectively. We examine the traditional ocean front detection pipeline, and use this existing pipeline to create a large set of satellite training data. We explore existing ML models and datasets for edge detection, adapting architectures and techniques to fit our task of ocean front detection. We train our models and analyze them across a broad range of metrics to determine the best fit for deployment on the CubeSat BeaverCube-2.

Regarding results, the ML models HED and Small CEDN make the highest quality predictions: achieving accuracies of at least 96% for detecting CHL and SST fronts on both BOA data and human annotated data. The ML models Small HED and Small CEDN use the least resources: utilizing less than 120 MB to make predictions in less than 0.002 seconds on a Google Colab GPU. This inference speed contrasts sharply with the traditional pipeline speed–which takes 16 days to process imagery from Level 0 to Level 2 and 22 seconds to detect ocean fronts using BOA on a Google

Colab GPU.

Overall, Small CEDN seems to have the most promise for CubeSat deployment. The mini-encoder-decoder architecture makes relatively accurate predictions while consuming relatively few resources, a good fit for a SWaP-constrained CubeSat.

Stepping back, it is important to remember that detecting ocean fronts on-orbit is a means to an end; a numeric way to queue images for downlink. These relatively simple on-orbit ML models serve as a technology demonstration and pave the way for future, more complex, on-orbit ML work.

5.2 Future Work

We present three avenues for future work in this area: analyzing the training data to further understand the relationship between CHL and SST ocean fronts, modifying the training data and models to differentiate between land and ocean, and fine-tuning the trained models on-orbit.

5.2.1 Relationship between CHL and SST Fronts

While we begin to explore the relationship between CHL and SST fronts in Section 4.1.3, there is future work that must be done to fully characterize the spatial and temporal relationship between the front types. There has been little work done in this area before, because ocean-observing satellites often do not observe the bands necessary to generate high-quality CHL and SST data at the same spatial and temporal resolutions [1, 27].

The training dataset developed in this work provides an opportunity to more fully explore the relationship between CHL and SST fronts, including quantifying the CHL/SST offset in different oceanic regions during different seasons.

5.2.2 Differentiating between Land and Ocean

Our models are designed to detect ocean fronts in oceanic images, but their performance on coastal images (in which there is a mix of land and water visible) has not been fully explored. There is often interesting ocean front activity in coastal regions [5, 20], so there is a use case for models that can detect ocean fronts in images that are not fully of the ocean.

BOA detects edges in images regardless of if these edges come from ocean fronts, coastlines, or features on land. Expanding our models to only detect the oceanic edges would involve creating a new human annotated training dataset–where edges that BOA detects on land are manually removed–and re-training the models. While time consuming, the ability of the models to differentiate between land and ocean would increase the number of potential model applications.

5.2.3 On-Orbit Fine-Tuning

Our models are trained with Landsat 8 data, but will be deployed on BeaverCube-2. If the BeaverCube-2 cameras are calibrated differently from the Landsat 8 cameras, the models could perform worse than expected [37, 50]. We predict this will not be a large concern because our models depend on image-wide gradients, instead of individual pixel values.

However, if this does present a problem, we could explore on-orbit fine-tuning. This process would involve computing approximate ground truths on-orbit (likely using the Sobel operator [24]) and fine-tuning the models on-orbit. The resources needed for on-orbit training are considerably greater than the resources needed for on-orbit inference, so this fine-tuning would have to be resource-conscious.

THIS PAGE INTENTIONALLY LEFT BLANK

Appendix A

Downloading Data from Google Earth Engine

Below is our code for downloading Landsat 8 data from Google Earth Engine, implemented in JavaScript.

```
//================================================
//Title:   Creating and Downloading Landsat Scenes
//Author:  Violet Felt
//Date:    6 May 2022
//================================================

var geometry = //YOUR ROI HERE
var gsd = 250; //meters per pixel

//create Level 2 dataset with necessary parameters
var proc_collection = ee.ImageCollection('LANDSAT/LC08/C02/T1_L2')
    .filterBounds(geometry)
    .filterDate('2017-01-01', '2022-01-01')
    .select(['SR_B1', 'SR_B2', 'SR_B3', 'SR_B4', 'ST_B10'])
    .filter('CLOUD_COVER < .05');
```

```
//create Level 0 dataset with necessary parameters
var raw_collection = ee.ImageCollection('LANDSAT/LC08/C02/T1')
    .filterBounds(geometry)
    .filterDate('2017-01-01', '2022-01-01')
    .select(['B1', 'B2', 'B3', 'B4', 'B10'])
    .filter('CLOUD_COVER < .05');

//find overlap between datasets
var filter = ee.Filter.equals({
  leftField: 'system:index',
  rightField: 'system:index'
});
var overlap = ee.Join.inner('L0', 'L2').apply(proc_collection,
    raw_collection, filter);
var joined = overlap.map(function(feature) {
  return ee.Image.cat(feature.get('L0'), feature.get('L2'));
});

//export each image
var n = joined.size().getInfo();
var colList = joined.toList(n);
for(var i = 0; i < n; i++){
    var img = ee.Image(colList.get(i));
    var name = img.get('system:index').getInfo().substring(0,20)
    Export.image.toDrive({
      image: img,
      scale: gsd,
      description: name,
      folder: 'ocean_images',
      fileFormat: 'GeoTIFF',
      });
```

```
}
```

Landsat scenes are labeled with the convention LXSS_LLLL_PPPRRR_YYYYMMDD [41] where:

- L = Landsat
- X = Sensor ("C"=OLI/TIRS combined)
- SS = Satellite ("08"=Landsat 8)
- PPP = Worldwide reference system path
- RRR = Woldwide reference system row
- YYYYMMDD = Acquisition year, month, day

The final set of 225 training scenes is listed below:

LC08_150046_20170112	LC08_115080_20200523	LC08_151045_20211216
LC08_152044_20210326	LC08_160078_20181207	LC08_148048_20200224
LC08_152044_20180302	LC08_151045_20170409	LC08_141049_20210225
LC08_116077_20210805	LC08_152044_20180318	LC08_116075_20190816
LC08_115074_20201014	LC08_152044_20171212	LC08_150046_20180216
LC08_116078_20210618	LC08_154043_20170414	LC08_150046_20210107
LC08_150046_20190102	LC08_116078_20190426	LC08_116078_20190715
LC08_115074_20200912	LC08_116077_20190512	LC08_150046_20210328
LC08_115074_20200710	LC08_149046_20171105	LC08_158078_20180920
LC08_159078_20210701	LC08_150046_20190203	LC08_151045_20210303
LC08_148048_20190410	LC08_116075_20170709	LC08_115074_20200608
LC08_115080_20170429	LC08_151045_20201229	LC08_115074_20170803
LC08_115074_20210424	LC08_151045_20210130	LC08_150046_20210208
LC08_147050_20211220	LC08_139047_20190222	LC08_151045_20180412
LC08_148048_20180101	LC08_154043_20211103	LC08_150046_20200222

LC08_148048_20180218	LC08_115074_20200726	LC08_203037_20170405
LC08_148048_20170303	LC08_116078_20190512	LC08_203035_20170912
LC08_151045_20191008	LC08_148048_20190120	LC08_011032_20211109
LC08_151045_20180311	LC08_150046_20201206	LC08_001028_20210425
LC08_148048_20181101	LC08_152044_20171025	LC08_016038_20180309
LC08_150046_20170128	LC08_116078_20190120	LC08_205052_20180116
LC08_152044_20170331	LC08_115074_20180907	LC08_008030_20180824
LC08_154043_20201031	LC08_116075_20190715	LC08_205032_20190324
LC08_115074_20170920	LC08_152044_20210206	LC08_182073_20210515
LC08_116075_20190917	LC08_115074_20191028	LC08_205033_20190324
LC08_116077_20201005	LC08_115080_20170224	LC08_205052_20170302
LC08_115074_20200827	LC08_115080_20200608	LC08_205052_20210225
LC08_115074_20170531	LC08_116078_20170607	LC08_003028_20210829
LC08_116078_20191104	LC08_115074_20190926	LC08_205031_20201121
LC08_116077_20180813	LC08_116078_20210805	LC08_011032_20200224
LC08_115080_20180518	LC08_151045_20190330	LC08_006029_20190525
LC08_116077_20190715	LC08_115074_20190403	LC08_205031_20210921
LC08_116078_20210415	LC08_150046_20210224	LC08_205052_20171231
LC08_150046_20180131	LC08_151045_20180327	LC08_005029_20171019
LC08_116078_20170404	LC08_116075_20170404	LC08_205032_20211007
LC08_116078_20210906	LC08_116077_20210618	LC08_176083_20200619
LC08_140048_20180314	LC08_140048_20210306	LC08_205032_20210921
LC08_151045_20201111	LC08_151045_20171103	LC08_205052_20170113
LC08_148048_20190104	LC08_147050_20180227	LC08_013041_20210413
LC08_160078_20200416	LC08_148048_20190309	LC08_005029_20200520
LC08_115074_20180923	LC08_150046_20170301	LC08_011032_20210226
LC08_116077_20190426	LC08_159078_20200612	LC08_205052_20191205
LC08_140048_20180226	LC08_116075_20200802	LC08_001028_20210714
LC08_154043_20180316	LC08_011032_20180423	LC08_005030_20171019
LC08_116077_20210906	LC08_178080_20210604	LC08_179079_20180705

LC08_205052_20170129	LC08_013033_20170418	LC08_040038_20210221
LC08_204034_20171106	LC08_196057_20200107	LC08_044035_20181007
LC08_178080_20210401	LC08_013033_20190830	LC08_038041_20210327
LC08_182073_20180608	LC08_015037_20171025	LC08_001084_20180127
LC08_179079_20170718	LC08_205052_20190119	LC08_043036_20171130
LC08_205033_20201121	LC08_012032_20201129	LC08_044035_20190924
LC08_013033_20200222	LC08_179079_20210729	LC08_001084_20180417
LC08_001028_20211103	LC08_008030_20190405	LC08_038043_20200104
LC08_179079_20210510	LC08_203035_20190310	LC08_044035_20211015
LC08_016039_20180309	LC08_179079_20181025	LC08_036043_20190324
LC08_203035_20190513	LC08_205033_20171028	LC08_001084_20181026
LC08_205052_20181218	LC08_006029_20211005	LC08_042037_20191012
LC08_221084_20210225	LC08_012032_20200318	LC08_112040_20170219
LC08_181075_20200724	LC08_179079_20190825	LC08_043036_20211125
LC08_203035_20170405	LC08_001084_20180212	LC08_001084_20210220
LC08_009030_20170913	LC08_042037_20181228	LC08_036043_20190119
LC08_011032_20200311	LC08_040038_20201203	LC08_042037_20201014
LC08_181075_20210524	LC08_042037_20201201	LC08_046033_20180207
LC08_181075_20180703	LC08_040038_20171024	LC08_046033_20201026
LC08_205032_20201121	LC08_040038_20201219	LC08_042037_20201115
LC08_181075_20200910	LC08_042037_20191231	LC08_044035_20180209
LC08_204034_20171005	LC08_001084_20200406	LC08_044035_20181108
LC08_176083_20200822	LC08_001084_20191130	LC08_044035_20200302
LC08_205031_20171028	LC08_040038_20180128	LC08_042037_20171006
LC08_179079_20210814	LC08_106037_20170430	LC08_043036_20171013
LC08_012032_20171020	LC08_047031_20181012	LC08_046033_20171205
LC08_205032_20171028	LC08_038041_20200104	LC08_040038_20181112
LC08_203035_20171201	LC08_045034_20210920	LC08_039040_20210113
LC08_181075_20190620	LC08_039040_20210302	LC08_047031_20191031
LC08_205033_20211007	LC08_001084_20180908	LC08_005083_20181107

THIS PAGE INTENTIONALLY LEFT BLANK

Appendix B

Belkin O'Reilly Algorithm Implementation

Below is our code for the Belkin O'Reilly Algorithm, implemented in Python. This is based on the pseudocode provided in the Belkin O'Reilly paper [7].

```
#================================================
#Title:   Belkin O'Reilly Algorithm
#Author:  Violet Felt
#Date:    6 May 2022
#================================================

#helper function: returns the median value of data
def median(data):
    return np.median(np.ndarray.flatten(data))

#helper function: returns true if the center value of data is the maximum
    value across all 1D slices of size filter_size, false otherwise
def peak_max(data, filter_size):
    indexer = filter_size // 2
    WE = np.argmax(data[indexer])
```

```
    NS = np.argmax(np.transpose(data)[indexer])
    NWSE = np.argmax(data.diagonal())
    NESW = np.argmax(data[:,::-1].diagonal())
    return {indexer} == {WE,NS,NWSE,NESW}

#helper function: returns true if the center value of data is the minimum
    value across all 1D slices of size filter_size, false otherwise
def peak_min(data, filter_size):
    indexer = filter_size // 2
    WE = np.argmin(data[indexer])
    NS = np.argmin(np.transpose(data)[indexer])
    NWSE = np.argmin(data.diagonal())
    NESW = np.argmin(data[:,::-1].diagonal())
    return {indexer} == {WE,NS,NWSE,NESW}

#helper function: returns a mask of size img where min/max peaks in a
    filter_size window are labeled with 1s and non-peaks are labeled with 0s
def peak_masks(img, filter_size):
    peaks = np.zeros(img.shape)
    indexer = filter_size // 2
    for i in range(img.shape[0]):
        for j in range(img.shape[1]):
            test = img[i-indexer:i+indexer+1,j-indexer:j+indexer+1]
            if test.shape == (filter_size,filter_size):
                if peak_max(test,filter_size) or peak_min(test,filter_size):
                    peaks[i][j] = 1
    return peaks

#contextual median filter: returns a filtered image where pixels that are
    not peak pixels in a 5-pixel-wide window but are peak pixels in a
    3-pixel-wide window are replaced with the median value of that
    3-pixel-wide window
```

```
def contextual_median_filter(img):
    peak_5 = peak_masks(img,5)
    peak_3 = peak_masks(img,3)
    new_img = np.zeros(img.shape)
    for i in range(img.shape[0]):
        for j in range(img.shape[1]):
            if not peak_5[i][j] and peak_3[i][j]:
                new_img[i][j] = median(img[i-1:i+2,j-1:j+2])
            else:
                new_img[i][j] = img[i][j]
    return new_img

#boa: returns the ocean fronts in an image by recursively filtering an
    image with the contextual median filter until convergence, then applying
    the Sobel operator
def boa(img):
  old_step = img
  new_step = contextual_median_filter(img)
  while not np.array_equal(old_step,new_step):
      old_step = new_step
      new_step = contextual_median_filter(old_step)
  filtered_img = np.float32(new_step)
  gx = cv.Sobel(filtered_img, cv.CV_32F, 1, 0, ksize=3)
  gy = cv.Sobel(filtered_img, cv.CV_32F, 0, 1, ksize=3)
  sobel_final = cv.addWeighted(gx, 0.5, gy, 0.5, 0)
  return sobel_final
```

THIS PAGE INTENTIONALLY LEFT BLANK

Appendix C

Machine Learning Model Architectures

Below is our code for the four ML model architectures, implemented in TensorFlow.

```
#================================================
#Title:   ML Model Architectures
#Author:  Violet Felt
#Date:    6 May 2022
#================================================

def HED():
  vgg = VGG16(weights='imagenet', include_top=False)
  input = Input(shape=(224,224,3))
  x = vgg.get_layer('block1_conv1')(input)
  x = vgg.get_layer('block1_conv2')(x)
  side1 = Conv2DTranspose(filters=16, kernel_size=3, strides=1,
      padding='SAME', activation='relu')(x)
  side1 = Conv2D(filters=4, kernel_size=3, padding='SAME',
      activation='relu')(side1)
  x = vgg.get_layer('block1_pool')(x)
```

```
x = vgg.get_layer('block2_conv1')(x)
x = vgg.get_layer('block2_conv2')(x)
side2 = Conv2DTranspose(filters=32, kernel_size=3, strides=2,
    padding='SAME', activation='relu')(x)
side2 = Conv2D(filters=8, kernel_size=3, padding='SAME',
    activation='relu')(side2)
x = vgg.get_layer('block2_pool')(x)
x = vgg.get_layer('block3_conv1')(x)
x = vgg.get_layer('block3_conv2')(x)
x = vgg.get_layer('block3_conv3')(x)
side3 = Conv2DTranspose(filters=64, kernel_size=3, strides=4,
    padding='SAME', activation='relu')(x)
side3 = Conv2D(filters=16, kernel_size=3, padding='SAME',
    activation='relu')(side3)
x = vgg.get_layer('block3_pool')(x)
x = vgg.get_layer('block4_conv1')(x)
x = vgg.get_layer('block4_conv2')(x)
x = vgg.get_layer('block4_conv3')(x)
side4 = Conv2DTranspose(filters=128, kernel_size=3, strides=8,
    padding='SAME', activation='relu')(x)
side4 = Conv2D(filters=32, kernel_size=3, padding='SAME',
    activation='relu')(side4)
x = vgg.get_layer('block4_pool')(x)
x = vgg.get_layer('block5_conv1')(x)
x = vgg.get_layer('block5_conv2')(x)
x = vgg.get_layer('block5_conv3')(x)
side5 = Conv2DTranspose(filters=128, kernel_size=3, strides=16,
    padding='SAME', activation='relu')(x)
side5 = Conv2D(filters=32, kernel_size=3, padding='SAME',
    activation='relu')(side5)
side_outputs = [side1, side2, side3, side4, side5]
output = Conv2D(filters=2, kernel_size=3, padding='SAME',
```

```
        activation='sigmoid')(tf.concat(side_outputs, axis=3))
    model = Model(inputs=input, outputs=output)
    return model

def HED_small():
    vgg = VGG16(weights='imagenet', include_top=False)
    input = Input(shape=(224,224,3))
    x = vgg.get_layer('block1_conv1')(input)
    x = vgg.get_layer('block1_conv2')(x)
    side1 = Conv2DTranspose(filters=16, kernel_size=3, strides=1,
        padding='SAME', activation='relu')(x)
    side1 = Conv2D(filters=4, kernel_size=3, padding='SAME',
        activation='relu')(side1)
    x = vgg.get_layer('block1_pool')(x)
    x = vgg.get_layer('block2_conv1')(x)
    x = vgg.get_layer('block2_conv2')(x)
    side2 = Conv2DTranspose(filters=32, kernel_size=3, strides=2,
        padding='SAME', activation='relu')(x)
    side2 = Conv2D(filters=8, kernel_size=3, padding='SAME',
        activation='relu')(side2)
    x = vgg.get_layer('block2_pool')(x)
    x = vgg.get_layer('block3_conv1')(x)
    x = vgg.get_layer('block3_conv2')(x)
    x = vgg.get_layer('block3_conv3')(x)
    side3 = Conv2DTranspose(filters=64, kernel_size=3, strides=4,
        padding='SAME', activation='relu')(x)
    side3 = Conv2D(filters=16, kernel_size=3, padding='SAME',
        activation='relu')(side3)
    side_outputs = [side1, side2, side3]
    output = Conv2D(filters=2, kernel_size=3, padding='SAME',
        activation='sigmoid')(tf.concat(side_outputs, axis=3))
    model = Model(inputs=input, outputs=output)
```

```
  return model

def CEDN():
  vgg = VGG16(weights='imagenet', include_top=False)
  input = Input(shape=(224,224,3))
  x = vgg.get_layer('block1_conv1')(input)
  x = vgg.get_layer('block1_conv2')(x)
  x = vgg.get_layer('block1_pool')(x)
  x = vgg.get_layer('block2_conv1')(x)
  x = vgg.get_layer('block2_conv2')(x)
  x = vgg.get_layer('block2_pool')(x)
  x = vgg.get_layer('block3_conv1')(x)
  x = vgg.get_layer('block3_conv2')(x)
  x = vgg.get_layer('block3_conv3')(x)
  x = vgg.get_layer('block3_pool')(x)
  x = vgg.get_layer('block4_conv1')(x)
  x = vgg.get_layer('block4_conv2')(x)
  x = vgg.get_layer('block4_conv3')(x)
  x = vgg.get_layer('block4_pool')(x)
  x = vgg.get_layer('block5_conv1')(x)
  x = vgg.get_layer('block5_conv2')(x)
  x = vgg.get_layer('block5_conv3')(x)
  x = vgg.get_layer('block5_pool')(x)
  x = Conv2D(filters=4096, kernel_size=1, padding='SAME')(x)
  x = Conv2DTranspose(filters=512, kernel_size=1, padding='SAME',
      activation='relu')(x)
  x = Conv2DTranspose(filters=256, kernel_size=5, strides=2,
      padding='SAME', activation='relu')(x)
  x = Conv2DTranspose(filters=128, kernel_size=5, strides=2,
      padding='SAME', activation='relu')(x)
  x = Conv2DTranspose(filters=64, kernel_size=5, strides=2, padding='SAME',
      activation='relu')(x)
```

```
  x = Conv2DTranspose(filters=32, kernel_size=5, strides=2, padding='SAME',
      activation='relu')(x)
  output = Conv2DTranspose(filters=2, kernel_size=5, strides=2,
      padding='SAME', activation='sigmoid')(x)
  model = Model(inputs=input, outputs=output)
  return model

def CEDN_small():
  vgg = VGG16(weights='imagenet', include_top=False)
  input = Input(shape=(224,224,3))
  x = vgg.get_layer('block1_conv1')(input)
  x = vgg.get_layer('block1_conv2')(x)
  x = vgg.get_layer('block1_pool')(x)
  x = vgg.get_layer('block2_conv1')(x)
  x = vgg.get_layer('block2_conv2')(x)
  x = vgg.get_layer('block2_pool')(x)
  x = vgg.get_layer('block3_conv1')(x)
  x = vgg.get_layer('block3_conv2')(x)
  x = vgg.get_layer('block3_conv3')(x)
  x = vgg.get_layer('block3_pool')(x)
  x = Conv2DTranspose(filters=128, kernel_size=5, strides=2,
      padding='SAME', activation='relu')(x)
  x = Conv2DTranspose(filters=64, kernel_size=5, strides=2, padding='SAME',
      activation='relu')(x)
  output = Conv2DTranspose(filters=2, kernel_size=5, strides=2,
      padding='SAME', activation='sigmoid')(x)
  model = Model(inputs=input, outputs=output)
  return model
```

THIS PAGE INTENTIONALLY LEFT BLANK

Bibliography

[1] Visible infrared imaging radiometer suite (VIIRS) sensor data record user's guide. *NOAA Technical Reports*, 2017.

[2] Atmospheric correction of optical imagery. *Advanced Remote Sensing*, 2020.

[3] System-level benefits of the versal platform. *Xilinx*, 2022.

[4] Martin Abadi et al. TensorFlow: Large-scale machine learning on heterogeneous systems. 2015.

[5] Mohd Abbas et al. Satellite estimation of chlorophyll-a using moderate resolution imaging spectroradiometer (MODIS) sensor in shallow coastal water bodies: Validation and improvement. *Water*, 2019.

[6] Igor Belkin. Remote sensing of ocean fronts in marine ecology and fisheries. *Remote Sensing*, 2021.

[7] Igor Belkin and John O'Reilly. An algorithm for oceanic front detection in chlorophyll and SST satellite imagery. *Journal of Marine Systems*, 2009.

[8] Garcia-Soto Carlos et al. An overview of ocean climate change indicators: Sea Surface Temperature, ocean heat content, ocean pH, dissolved oxygen concentration, arctic sea ice extent, thickness and volume, sea level and strength of the AMOC (atlantic meridional overturning circulation). *Frontiers in Marine Science*, 2021.

[9] Jean-François Cayula and Peter Cornillon. Edge detection algorithm for SST images. *Journal of Atmospheric and Oceanic Technology*, 1992.

[10] Committee on Achieving Science Goals with CubeSats et al. *Achieving Science with CubeSats: Thinking Inside the Box*. National Academies Press, 2016.

[11] Mary Dahl and Benjamin Martell. BeaverCube 2 coastline, salinity, and temperature measurements. 2021.

[12] Rafael Pires de Lima and Kurt Marfurt. Convolutional neural network for remote-sensing scene classification: Transfer learning analysis. *Remote Sensing*, 2019.

[13] John Dwyer. Landsat collection 1 level 1 product definition. 2019.

[14] Yabo Fu et al. Deep learning in medical image registration: a review. *Physics in Medicine Biology*, 2020.

[15] Biyun Guo, M.V. Subrahmanyam, and C. Li. Waves on Louisiana continental shelf influenced by atmospheric fronts. *Scientific Reports*, 2020.

[16] C. Heinze et al. The ocean carbon sink - impacts, vulnerabilities and challenges. *Earth System Dynamics*, 2015.

[17] Ove Hoegh-Guldberg et al. The ocean as a solution to climate change: Five opportunities for action. *World Resources Institute*, 2019.

[18] Chuanmin Hu, Zhongping Lee, and Bryan Franz. Chlorophyll-a algorithms for oligotrophic oceans: A novel approach based on three-band reflectance difference. *Journal of Geophysical Research: Oceans*, 2012.

[19] Shruti Jadon. A survey of loss functions for semantic segmentation. *2020 IEEE Conference on Computational Intelligence in Bioinformatics and Computational Biology (CIBCB)*, 2020.

[20] Jae-Cheol Jang and Kyung-Ae Park. High-resolution sea surface temperature retrieval from Landsat 8 OLI/TIRS data at coastal regions. *Remote Sensing*, 2019.

[21] Calli Jenkerson. Landsat 8 collection 1 land surface reflectance code (LaSRC) product guide. 2020.

[22] Shreeyam Kacker et al. On-orbit rule-based and deep learning image segmentation strategies. *AIAA SciTech*, 2022.

[23] Shreeyam Kacker and Alex Meredith. BeaverCube2 cloud segmentation. 2021.

[24] Nikos Kanopoulos, N. Vasanthavada, and R.L. Baker. Design of an image edge detection filter using the Sobel operator. *IEEE Journal of Solid-State Circuits*, 1988.

[25] Andrew Kennedy. Planning and scheduling for earth-observing small satellite constellations, 2018.

[26] K. Kilpatrick, G. Podestá, and R. Evans. Overview of the NOAA/NASA advanced very high resolution radiometer Pathfinder algorithm for sea surface temperature and associated matchup database. *Journal of Geophysical Research: Oceans*, 2001.

[27] Donald Lauer, Stanley Morain, and Vincent Salomonson. The Landsat program: Its origins, evolution, and impacts. *Photogrammetric Engineering Remote Sensing*, 1997.

[28] Qingyang Li, Guoqiang Zhong, and Cui Xie. Weak edge identification nets for ocean front detection. *arXiv*, 2019.

[29] Tsung-Yi Lin et al. Focal loss for dense object detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.

[30] Yun Liu et al. Richer convolutional features for edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2019.

[31] D. Martin et al. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. *Eighth IEEE International Conference on Computer Vision*, 2001.

[32] Yackar Mauzole. Dynamical typology of sea surface temperature fronts based on satellite observations, 2017.

[33] William McIlhagga. Estimates of edge detection filters in human vision. *Vision Research*, 2018.

[34] Peter Miller. Detection and visualisation of oceanic fronts from satellite data, with applications for fisheries, marine megafauna and marine protected areas. *Handbook of Satellite Remote Sensing Image Interpretation: Marine Applications*.

[35] Peter Miller. Composite front maps for improved visibility of dynamic sea-surface features on cloudy SeaWiFS and AVHRR data. *Journal of Marine Systems*, 2009.

[36] Sorour Mohajerani and Parvaneh Saeedi. Cloud-Net: An end-to-end cloud detection algorithm for Landsat 8 imagery. 2019.

[37] Ron Morfitt et al. Landsat-8 operational land imager (OLI) radiometric performance on-orbit. *Remote Sensing*, 2015.

[38] NASA. Sea Surface Temperature and Chlorophyll, 2020. Available at `https://earthobservatory.nasa.gov/global-maps/MYD28M/MY1DMM_CHLORA`.

[39] John O'Reilly et al. Ocean color chlorophyll algorithms for SeaWiFS. *Journal of Geophysical Research: Oceans*, 1998.

[40] Albert Reuther et al. Interactive supercomputing on 40,000 cores for machine learning and data analysis. *IEEE High Performance Extreme Computing Conference (HPEC)*, 2018.

[41] David Roy et al. Landsat-8: Science and product vision for terrestrial global change research. *Remote Sensing of Environment*, 2014.

[42] Luka Rumora, Mario Miler, and Damir Medak. Impact of various atmospheric corrections on Sentinel-2 land cover classification accuracy using machine learning classifiers. *ISPRS International Journal of Geo-Information*, 2020.

[43] Daniel Selva and David Krejci. A survey and assessment of the capabilities of CubeSats for earth observation. *Acta Astronautica*, 2012.

[44] Li Shuangfeng. TensorFlow Lite: On-device machine learning framework. *Journal of Computer Research and Development*, 2020.

[45] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, 2015.

[46] H. Solanki, R. Dwivedi, and S. Nayak. Synergistic analysis of SeaWiFS chlorophyll concentration and NOAA-AVHRR SST features for exploring marine living resources. *International Journal of Remote Sensing*, 2001.

[47] Xavier Soria, Edgar Riba, and Angel Sappa. Dense extreme inception network: Towards a robust CNN model for edge detection. *IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2020.

[48] P. Stegmann and D. Ullman. Variability in chlorophyll and sea surface temperature fronts in the long island sound outflow region from satellite observations. *Journal of Geophysical Research: Oceans*, 2004.

[49] André Stumpf, David Michéa, and Jean-Philippe Malet. Improved co-registration of Sentinel-2 and Landsat-8 imagery for earth surface motion measurements. *Remote Sensing*, 2018.

[50] Albert Thieu et al. BeaverCube2 imaging payload: Camera assembly analysis and application. 2021.

[51] David Ullman and Peter Cornillon. Satellite-derived sea surface temperature fronts on the continental shelf off the northeast U.S. coast. *Journal of Geophysical Research: Oceans*, 1999.

[52] David Ullman and Peter Cornillon. Evaluation of front detection methods for satellite-derived SST data using in situ observations. *Journal of Atmospheric and Oceanic Technology*, 2000.

[53] Yu Wang, Gu-Yeon Wei, and David Brooks. Benchmarking TPU, GPU, and CPU platforms for deep learning. *CoRR*, 2019.

[54] Zhou Wang et al. Image quality assessment: From error visibility to structural similarity. *IEEE Transactions on Image Processing*, 2004.

[55] Saining Xie and Zhuowen Tu. Holistically-nested edge detection. *IEEE International Conference on Computer Vision (ICCV)*, 2015.

[56] Guangjun Xu et al. Application of three deep learning schemes into oceanic eddy detection. *Frontiers in Marine Science*, 2021.

[57] Jimei Yang et al. Object contour detection with a fully convolutional encoder-decoder network. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[58] Nicholas Young et al. A survival guide to Landsat preprocessing. *Ecology*, 2017.